Simulation of a closed-loop dc-dc converter using a physics-informed neural network-based model

Marc-Antoine Coulombe, Maxime Berger, and Antoine Lesage-Landry

***Abstract*—The growing reliance on power electronics introduces new challenges requiring detailed time-domain analyses with fast and accurate circuit simulation tools. Currently, commercial time-domain simulation software are mainly relying on physics-based methods to simulate power electronics. Recent work showed that data-driven and physics-informed learning methods can increase simulation speed with limited compromise on accuracy, but many challenges remain before deployment in commercial tools can be possible. In this paper, we propose a physics-informed bidirectional long-short term memory neural network (BiLSTM-PINN) model to simulate the time-domain response of a closed-loop dc-dc boost converter for various operating points, parameters, and perturbations. A physics-informed fully-connected neural network (FCNN) and a BiLSTM are also trained to establish a comparison. The three methods are then compared using step-response tests to assess their performance and limitations in terms of accuracy. The results show that the BiLSTM-PINN and BiLSTM models outperform the FCNN model by more than 9 and 4.5 times, respectively, in terms of median RMSE. Their standard deviation values are more than 2.6 and 1.7 smaller than the FCNN's, making them also more consistent. Those results illustrate that the proposed BiLSTM-PINN is a potential alternative to other physics-based or data-driven methods for power electronics simulations.**

***Keywords*—boost converter, machine learning, modelling, neural network, power converters, time-domain simulation.**

I. Introduction

THE shift toward renewable sources of energy has accelerated, bringing a growing reliance on power electronics converters in power systems [1]. Their presence increases power system complexity and introduces new challenges related to stability, reliability, and safety. This requires the use of detailed time-domain analyses utilizing offline and real-time simulation tools to predict their impacts on power systems [2]. Fast and accurate simulation of power electronics remains challenging due to their time-varying structure, nonlinear response, and switching dynamics [3]. Moreover, recent applications require the use of real-time models and simulations, such as model predictive control and digital twins, further increasing the demand for faster circuit simulation [4]–[6]. Commercially available electromagnetic transient (EMT) simulation software predominantly relies on physics-based methods to translate power electronics converters into equations that can be solved with classical numerical techniques [7]. Computation time reduction is generally done by substituting a time-varying with an equivalent fixed topology using switching functions [8] or by using averaging techniques [9], [10]. Recent works have shown that data-driven methods, physics-informed learning-based methods, and optimization techniques, can significantly increase the simulation speed while providing high-accuracy predictions, thus complementing traditional computational techniques [11], [12]. Data-driven and physics-informed learning-based methods are also valued for their ability to model commercial power electronics converters in a blackbox manner, removing the need for prior knowledge of the system [13], [14]. These emerging approaches are still in their early stage, such that their performance requires deeper evaluation to make them suitable for reliable integration into commercial simulation tools.

Among the different machine learning-based modelling approaches used to model power electronics converters, neural network dominates, with primarily two architectures: feedforward neural networks (FNNs) [15]–[17] and recurrent neural networks (RNNs) [6], [11], [13]. In [15], a model with multiple fully-connected neural networks (FCNNs) is developed to model an IGBT's switching transients on a field-programmable gate array (FPGA) for real-time simulation. The authors achieved real-time simulation with a timestep of 5 ns. However, the proposed model requires 150 and 500 neural networks for simulating the turn-on and turn-off transient waveforms, respectively. In [16], the authors combine an FCNN with Bayesian regularization backpropagation to model the transient response and random forests for the steady-state response. Using random forests alongside neural networks increases the model stability and reduces its variance, as the results obtained by decision trees are easier to interpret. An FCNN is used in [17] to predict intermediate latent states between two observable states. Then, a physical model is used to predict the observable states using predictions from the intermediate latent states enabling them to integrate physics into their model. This approach requires solving multiple equations for each prediction using an implicit Runge-Kutta method, which can increase the computation time for complex models. In [13], a long-short term memory (LSTM) neural network is used to model a dc-dc buck converter. Although the LSTM appears to perform

M.-A. Coulombe and M. Berger are with the Department of Mathematics, Informatics, and Engineering, Université du Québec à Rimouski, Rimouski, Québec, Canada (e-mail of corresponding authors: [marc-antoine.coulombe, maxime_berger]@uqar.ca).

A. Lesage-Landry is with the Department of Electrical Engineering, Polytechnique Montréal, GERAD & Mila, Montréal, Québec, Canada (e-mail: antoine.lesage-landry@polymtl.ca).

This work was funded by Mitacs, the Natural Sciences and Engineering Research Council of Canada (NSERC), and OPAL-RT Technologies under Grants ALLRP593314 & IT40898.

Paper submitted to the International Conference on Power Systems Transients (IPST2025) in Guadalajara, Mexico, June 8-12, 2025.

better than other FNN-based architectures for modelling the converter response, limited quantitative assements are provided to compare performance. Which is required for reliable deployment in simulation tools. In [11], the output signal is decomposed into a transient and a periodic component modelled, respectively, by an FNN and an LSTM combined with a convolutional neural network. This approach reduces the complexity of the problem into two sub-problems. Reference [6] uses a nonlinear autoregressive exogenous network, which yields good prediction accuracy. However, this type of architecture is limited in its ability to capture long-term dependencies without using internal memories like LSTMs. To consider a longer sequence of values, a large input size is required, which reduces computational efficiency.

While not directly applied to model power electronics converters, other data-driven approaches are also considered in the power systems literature, e.g., physics-informed neural ordinary differential equations for modelling power transformer's dynamic thermal behaviour [18] or for predicting the nonlinear transient dynamics of a inverter under fault [19] and graph-based learning for high-frequency filter design or microgrids voltage prediction [20]. These approaches are promising alternatives for future studies on power electronics modelling.

Among the aforementioned articles on power electronics simulations, the following limitations have been observed:

- Some proposed solutions are difficult to generalize to multiple types of converters and different topologies;
- The assessment of data-driven models is limited, i.e., with little to no quantitative metrics, thus, preventing their reliable deployment in simulation tools for power system analysis;
- Some proposed solutions require significant computational time, making them difficult to apply in real-time simulations or without specialized hardware.

The main contributions of this paper are (i) designing a physics-informed bidirectional LSTM (BiLSTM-PINN) network architecture and to model the time-domain response of a closed-loop dc-dc boost converter and (ii) thoroughly benchmarking our model using step-response tests. The BiLSTM architecture stands out as one well studied and documented, and easy to implement. It can also run on graphics processing units (GPUs) or FPGAs for computation acceleration [21], [22], making it readily accessible and practical for deployment in simulation software [23], [24]. This architecture also takes into account both past and future values to make predictions, in contrast to RNNs or standard LSTMs, which only consider past values or FCNNs that are limited to present values. This property improves performance by utilizing the initial and final operating point values derived from the dc steady-state model to better reproduce the converter response. The inclusion of a physics-informed loss also helps mitigate prediction errors by penalizing results that do not adhere to physical principles during training. Computational time evaluation is left for future work as many sources of interference such as the programming language (compiled MATLAB® *vs.* interpreted Python), the solver(s) used for the physical model or the hardware used, can make the results unreliable. This further underscores the importance of first defining the model architecture and assessing the model's accuracy, as it is done in this work. However, we consider that function evaluation with neural networks should be faster than solving a complex system of time-varying equations as hinted in [11].

In this paper, we train an FCNN, a BiLSTM, and a BiLSTM-PINN to model the input and output currents of a closed-loop dc-dc boost converter's large-signal averaged model. The validation of the FCNN and BiLSTM models with a detailed switching model is beyond the scope of this paper. By using an averaged model, we aim to reduce the complexity of the neural network, enabling a more generalizable methodology and reduced computation time. Additionally, this model helps us quantify the transient waveforms using common performance metrics. This model is utilized to generate a dataset using MATLAB® Simulink for various operating points, parameters, and perturbations. The dataset is then used to train an FCNN, a BiLSTM and, a BiLSTM-PINN, with hyperparameter tuning performed via `HyperOpt` [25]. Finally, the two architectures are compared using step-response tests to assess model performance.

The paper is organized as follows. Section II presents the dc-dc boost converter topology and the large-signal averaged model used to generate the training and testing datasets. Section III presents the physics-informed learning-based boost converter models. Section IV discusses dataset generation and hyperparameter tuning, and validates the accuracy of our method via time-domain simulations in MATLAB® Simulink.

II. Boost Converter Model

The closed-loop boost converter topology used in this paper is shown in Fig. 1. The switch S is controlled by a pulse-width modulation (PWM) signal, introducing two distinct circuit states: switch S open and closed. The duty cycle $d(t)$ represents the proportion of time with the switch S closed during a complete period T_s, while the complementary duty cycle is given by $d'(t) = 1 - d(t)$. The input current i is the regulated variable. The transfer function G_f models the presence of a filter on the current measurement.

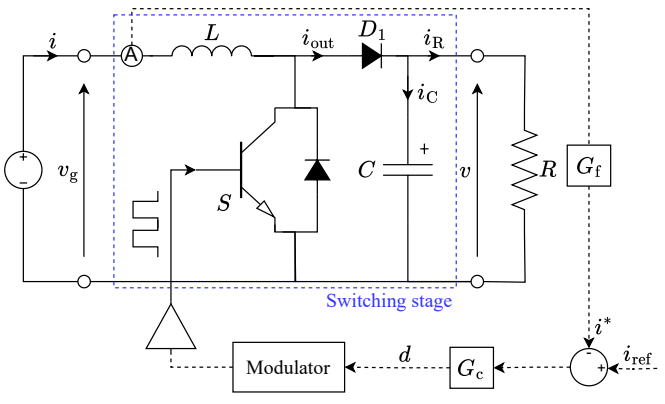


Fig. 1: Circuit topology of the current regulated boost converter.

The circuit in Fig. 1 regulates the input current i to match the reference i_{ref} by calculating the error signal $\epsilon = i_{\text{ref}} - i^*$, where i^* is the filtered input current. The error signal is then fed into a PI controller, with a transfer function defined as:

$$G_{\text{c}}(s) = \frac{sK_{\text{p}} + K_{\text{i}}}{s} \ ,$$

where $K_{\text{p}} \in \mathbb{R}$ and $K_{\text{i}} \in \mathbb{R}$ are determined using the *algebra on the graph* technique [26] and further tuned to achieve the desired dynamic performance, with specified cross-over frequency $\omega_{\phi_{\text{m}}} = 2\pi f_{\phi_{\text{m}}}$ and phase-margin ϕ_{m} as explicited in Appendix. The filter and controller parameters for the converter model under validation are specified in Section IV.

In dc steady-state, the dc conversion ratio $M(D)$, the inductor's dc current I_{L}, the inductor's current ripple Δi_{L} and the capacitance's voltage ripple Δv_{C} are derived as

$$M(D) = \frac{V}{V_{\text{g}}} = \frac{1}{1-D} \tag{1}$$

$$I_{\text{L}} = \frac{V}{(1-D)R} \tag{2}$$

$$\Delta i_{\text{L}} = \frac{DV_{\text{g}}}{2f_{\text{s}}L} \tag{3}$$

$$\Delta v_{\text{C}} = \frac{DV}{2f_{\text{s}}RC}, \tag{4}$$

where V_{g} and V are the dc input and output voltages, D is the duty cycle, and f_{s} is the switching frequency. We use (1)−(4) to describe the boost converter in initial and final steady-states which are used to initialize the BiLSTM models and the simulation parameters for the generation of the dataset in Section III and Section IV, respectively.

We then use the classical dc-averaging technique for dc-dc converter [27] to simplify the switching stage (in blue colour in Fig. 1) leading to the following differential equations:

$$L\frac{\mathrm{d}\langle i\rangle_{T_{\text{s}}}}{\mathrm{d}t} = \langle v_{\text{g}}\rangle_{T_{\text{s}}} - d'\langle v\rangle_{T_{\text{s}}} \tag{5}$$

$$C\frac{\mathrm{d}\langle v\rangle_{T_{\text{s}}}}{\mathrm{d}t} = d'\langle i\rangle_{T_{\text{s}}} - \frac{\langle v\rangle_{T_{\text{s}}}}{R}, \tag{6}$$

where denotes $\langle\rangle_{T_{\text{s}}}$ the average value of a variable over T_{s}. The equivalent circuit derived from (5) and (6) is illustrated in Fig. 2. Our objective is to model the currents $\langle i\rangle_{T_{\text{s}}}$ and $\langle i_{\text{out}}\rangle_{T_{\text{s}}}$ in Fig. 2 with learning-based methods. The generation of the dataset to train learning-based models using this circuit is presented in Section IV.

III. Physics-informed Learning-based Boost Converter Models

We now develop our learning-based model of the boost converter. Let $\mathbf{x} \in \mathbb{R}^{18}$ and $\mathbf{y} \in \mathbb{R}^2$ be, respectively, the input (feature) and output (target) vector. We use the subscript $t \in \mathbb{N}$ to denote the time instance inputs/outputs are registered. We define the feature vector $\mathbf{x}_t$ as

$$\mathbf{x}_t = \text{vec}\begin{bmatrix} t & L & C \\ R & R_{\text{s}} & V_{\text{g}} \\ V(t=0) & \Delta v(t=0) & I_{\text{L}}(t=0) \\ \Delta i_{\text{L}}(t=0) & I_{\text{step}} & t_{\text{step}} \\ K_{\text{p}} & K_{\text{i}} & f_{\text{s}} \\ f_{\text{c}} & i_{\text{ref},t} & d_t \end{bmatrix}, \tag{7}$$

where vec is the vectorization operator. Various constants that characterize the converter are collected in (7), such as the inductance L, capacitance C, load resistance R, inductance resistance R_{s}, input voltage V_{g}, switching frequency f_{s}, and controller parameters K_{i} and K_{p}, as derived in Section II. Next, additional features are used to define the initial state of the system, namely, the output voltage $V(t=0)$, the capacitance's voltage ripple $\Delta v(t=0)$, the inductor's current $I_L(t=0)$, and the inductor's current ripple $\Delta i_{\text{L}}(t=0)$. Then, time-varying features, viz., the desired current $i_{\text{ref},t}$ and the control signal d_t are included. Finally, I_{step} and t_{step} specify the amplitude and timing of the square disturbance on $i_{\text{ref},t}$.

In our setting, the entire system is assumed to be known, enabling the learning-based models to leverage comprehensive information about the circuit, its initial state, and its state at time t for accurate predictions. This assumption is compatible with settings encountered in power system simulation tools. Further explorations could be done to reduce the dimension of the feature vector $\mathbf{x}_t$ while minimizing the impact on accuracy. This is a topic for future work.

Next, we define the target vector at time t as

$$\mathbf{y}_t = \begin{bmatrix}\langle i\rangle_{T_{\text{s}},t} & \langle i_{\text{out}}\rangle_{T_{\text{s}},t}\end{bmatrix}.$$

The goal of the learning-based approach is to predict the input current $\langle i\rangle$ and the output current $\langle i_{\text{out}}\rangle$, derived using the previously defined large-signal averaged model. The decision to focus on current predictions is primarily to ensure signals operate on comparable time scales, hence simplifying the result analysis.

Let $\texttt{FCNN} : \mathbb{R}^{18} \to \mathbb{R}^2$ and $\texttt{BiLSTM} : \mathbb{R}^{18} \to \mathbb{R}^2$ be, respectively, the trained FCNN and BiLSTM with their corresponding collections of weights denoted by $\mathcal{W}_{\text{FCNN}}$ and $\mathcal{W}_{\text{BiLSTM}}$. We first model the converter with an FCNN, a common architecture in the machine learning literature. The input vector $\mathbf{x}_t$ is processed through the network, where a sequence of activation function and linear combination based weights $\mathbf{W} \in \mathcal{W}_{\text{FCNN}}$ compositions produces the prediction $\hat{\mathbf{y}}_{\text{FCNN},t}$ at the output layer. The FCNN is later used for benchmarking. We then utilize a BiLSTM to model the converter dynamics. To generate a prediction $\hat{\mathbf{y}}_{\text{BiLSTM},t}$, the BiLSTM network processes features from $\mathbf{x}_{t-k}$ to $\mathbf{x}_t$ and $\mathbf{x}_{t+k}$ to $\mathbf{x}_t$ through LSTM cells, as illustrated in Fig. 3. Here, $k \in \mathbb{N}$ denotes the sequence length, a hyperparameter that will be discussed in Section IV. The weights $\mathbf{W} \in \mathcal{W}_{\text{BiLSTM}}$ are the trainable parameters within each LSTM cell fully-connected layers. Because the LSTM cell follows a well-documented topology, we refer interested readers to, e.g., [13], [28], for additional details. Finally, given observation $\mathbf{x}_t$, the resulting predictions $\hat{\mathbf{y}}_t$ are:

$$\hat{\mathbf{y}}_{\text{FCNN},t} = \texttt{FCNN}(\mathbf{x}_t; \mathcal{W}_{\text{FCNN}}) \tag{8}$$

$$\hat{\mathbf{y}}_{\text{BiLSTM},t} = \texttt{BiLSTM}(\mathbf{x}_t; \mathcal{W}_{\text{BiLSTM}}). \tag{9}$$

To achieve accurate predictions, the collections of weights $\mathcal{W}_{\text{FCNN}}$ and $\mathcal{W}_{\text{BiLSTM}}$ in (8) and (9) are computed to minimize the discrepancy between the predictions $\hat{\mathbf{y}}$ and the targets $\mathbf{y}$ through a loss function. Let the loss function be defined as

$$\mathcal{L} = \mathcal{L}_{\text{RMSE}} + \lambda\mathcal{L}_{\text{PBE}}, \tag{10}$$

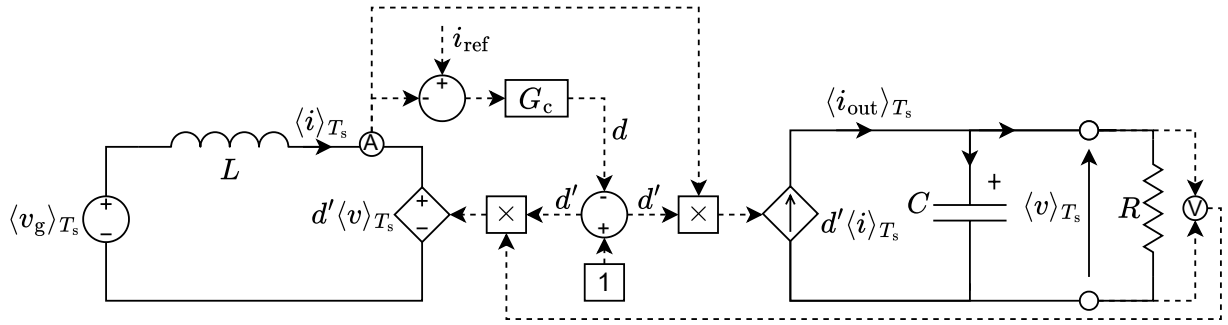


Fig. 2: Closed-loop large-signal averaged model for the boost converter.

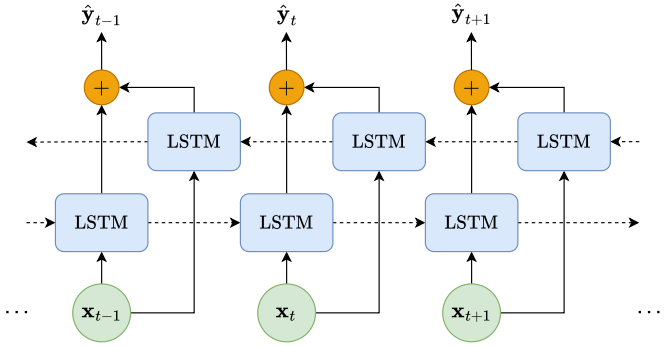


Fig. 3: Schematic of a bidirectional LSTM neural network.

where $\mathcal{L}$ combines the root-mean-squared error $\mathcal{L}_{\text{RMSE}}$ with the circuit power balance error $\mathcal{L}_{\text{PBE}}$ weighted by a constant $\lambda > 0$ which is to be tuned. The loss function is used to compute the gradient, which drives the backpropagation algorithm that trains both networks. By iteratively adjusting the collections of weights $\mathcal{W}_{\text{FCNN}}$ and $\mathcal{W}_{\text{BiLSTM}}$ using (10), the models progressively improve their accuracy. Specifically, the loss function $\mathcal{L}_{\text{RMSE}}$ for a single curve is

$$\mathcal{L}_{\text{RMSE}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\hat{\mathbf{y}}_t - \mathbf{y}_t)^2}\ , \tag{11}$$

where $n \in \mathbb{N}$ is the number of points on a curve related to the simulation duration and the step size. The power balance loss function $\mathcal{L}_{\text{PBE}}$ is then given by:

$$\mathcal{L}_{\text{PBE}} = \frac{1}{n}\sum_{t=1}^{n} |\langle P_{\text{in}} \rangle_{T_{\text{s}},t} - \langle P_{\text{out}} \rangle_{T_{\text{s}},t}|, \tag{12}$$

where the predicted currents $\hat{\mathbf{y}}_t$ are used to calculate, respectively, the predicted power entering and exiting the converter, defined as:

$$\langle P_{\text{in}} \rangle_{T_{\text{s}},t} = d'_t \langle v \rangle_{T_{\text{s}},t} \langle i \rangle_{T_{\text{s}},t} \tag{13}$$

$$\langle P_{\text{out}} \rangle_{T_{\text{s}},t} = \langle v \rangle_{T_{\text{s}},t} \langle i_{\text{out}} \rangle_{T_{\text{s}},t}\ . \tag{14}$$

In other words, we consider that the conservation of power applies while neglecting the converter inner losses. This assumption allows us to integrate physics into the BiLSTM loss function by penalizing predictions yielding a high-power imbalance between the converter's input and output, then leading to the BiLSTM-PINN (and FCNN) model.

IV. Numerical Results

We now present our numerical setting and then evaluate the performance of our learning-based models.

A. Numerical setting and training

To conduct the experiment, we first generate our dataset. This dataset is created by reproducing the lossless switching circuit in Fig. 1 and the large-signal averaged model shown in Fig. 2 in MATLAB® Simulink. The simulation parameters to create the dataset are provided in Table I.

TABLE I: Simulations Paramaters for the Dataset Generation.

Parameter	Interval	Step size
C (μF)	[1000, 7000]	1000
$d(t)$	± 0.9	–
f_{c} (kHz)	5	–
f_{s} (kHz)	70	–
$f_{\phi_{\text{m}}}$ (Hz)	255	–
I_{step} (A)	$\pm I_L(t=0)$	–
L (mH)	[1, 7]	1
R (Ω)	[1, 500]	10
R_{s} (Ω)	30×10^{-3}	–
V (V)	[200, 400]	10
V_{g} (V)	[100, 200]	10
t (s)	[0, 0.03]	250×10^{-7}
t_{step} (s)	[0.011, 0.021]	–
ϕ_{m} (°)	50	–

Key parameters of Fig. 2's circuit are defined, with their study range specified in a [lower bound, upper bound] format along with the step size for varying parameters such as C, L, R, V, and V_{g}. Other parameters, like f_{s} and R_{s}, remain constant for all simulations. During a simulation, a current step of amplitude I_{step} is applied to i_{ref} at time t_{step} based on a uniform distribution. The simulation itself is carried out from $t = 0$ to $t = 0.03$ with a fixed time step of 250×10^{-7}s.

In total, 565,950 operating points are simulated. Among these, cases that do not exhibit continuous conduction mode are excluded because they display response not accounted for by the large-signal averaged model. We also removed cases with an unfeasible PI controller (see Appendix). In addition, any cases where $|d| = 0.9$ are also rejected because the controller is programmed to saturate at this value, and saturation introduces non-linearities outside of the scope of this study. Approximately 10% of the dataset is removed for these reasons. Additionally, the points corresponding to the first 0.01 second of each simulation are discarded to ensure the

model reaches steady-state before introducing a perturbation. Finally, the dataset is split randomly using the holdout method with an 80%−20% ratio.

We identify suitable hyperparameters for our learning-based models using `HyperOpt` [25]. The resulting hyperparameters are presented in Table II together with the interval considered by `HyperOpt`. To define the intervals, preliminary trials were conducted to establish practical ranges, avoiding underfitting or overfitting. These issues were identified through an observable increase in the loss function $\mathcal{L}$. Subsequently, 30 optimization trials were performed using the tree-structured Parzen estimator (TPE) algorithm, with the objective of minimizing $\mathcal{L}$. The resulting values were rounded for simplicity. We chose the `ADAM` optimizer for all trials as it consistently provided better results compared to the alternatives [29]. From the hyperparameters described in Table II, the FCNN, BiLSTM, and BiLSTM-PINN models are trained on 80% of the dataset. Because our model is trained offline, we train multiple models simultaneously and select the one that delivers the best testing RMSE. Although this approach does not fully address sensitivity issues, it helps mitigate their impact by choosing the best-performing empirical model for deployment. Robustness of the model, the stability of outputs across different inputs, still remains to be established similarly to many of the neural network models. This is a topic for future work.

TABLE II: Hyperparameters interval and tuned values

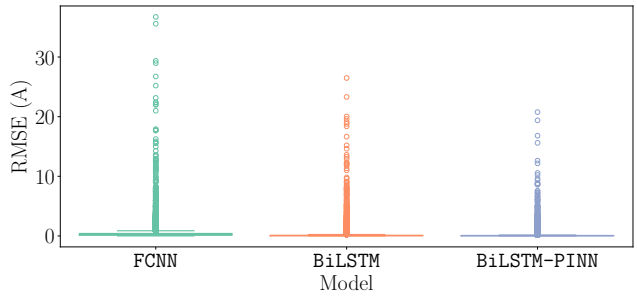
Hyperparameter	Interval	`FCNN`	`BiLSTM(-PINN)`
Batch size	[128, 300]	294	207
Epochs	–	35	35
Learning rate	[1e^{-4}, 1e^{-2}]	6e^{-4}	1e^{-3}
Optimizer	–	`ADAM`	`ADAM`
Neurons/Cells (per hidden layer)	[100, 500]	304	165
Sequence length k	[10, 25]	–	25
Weights decay	[1e^{-5}, 1e^{-2}]	9e^{-3}	7e^{-4}
λ	[0, 1]	0.2	0 (0.2)

B. Test

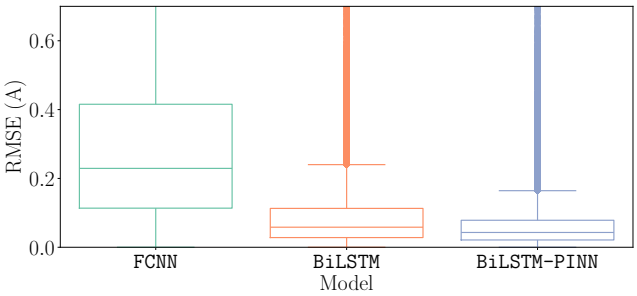
We then use the remaining 20% of the dataset for out-of-sample testing purposes. For each predicted response curves, we calculate the RMSE using (11). The resulting box plot is illustrated at Fig. 4.

Fig. 4 presents the RMSE for the FCNN, BiLSTM, and BiLSTM-PINN models. From Fig. 4a, we can observe that some operating point's RMSE greatly diverges from the median value. This is especially the case for the FCNN. Fig. 4b further zooms on the box plot distribution. We remark that the BiLSTM-PINN outperforms the FCNN and the BiLSTM by achieving a lower median $\mathcal{L}_{\text{RMSE}}$ and distribution dispersion.

Next, we quantified the model prediction performance using the `stepinfo` function from MATLAB®. We refer readers to their documentation for more details on the function and the evaluated metrics [30]. Table III compares all performance metrics for the FCNN and BiLSTM models.



(a) RMSE distribution at the FCNN scale. Circles represent the outliers of each model on the test dataset.



(b) Quartiles comparison for both learning-based models representing quartiles and median values. The lowest and highest delimitations of the box are the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentile, respectively. The vertical line inside the box represents the median and circles (appearing as a bold line) are the RMSE distribution outliers.

Fig. 4: RMSE box plot for the FCNN, BiLSTM, and BiLSTM-PINN models.

Recall that $\mathbf{y}_t$ ($\hat{\mathbf{y}}_t$) consists of the input (predicted) current $\langle i \rangle_{T_s,t}$ ($\langle \hat{i} \rangle_{T_s,t}$) and the (predicted) output $\langle i_{\text{out}} \rangle_{T_s,t}$ ($\langle \hat{i}_{\text{out}} \rangle_{T_s,t}$). Let the total absolute prediction error be

$$e = e_{\text{input}} + e_{\text{output}},$$

where e_{input} and e_{output} are defined as the absolute input and output error, respectively, given by

$$e_{\text{input}} = \left| \langle \hat{i} \rangle_{T_s,t} - \langle i \rangle_{T_s,t} \right|$$
$$e_{\text{output}} = \left| \langle \hat{i}_{\text{out}} \rangle_{T_s,t} - \langle i_{\text{out}} \rangle_{T_s,t} \right|.$$

For each `stepinfo` characteristics, we report the mean, the standard deviation σ_{std}, and the median. Table III compares all performance metrics for the FCNN, BiLSTM, and BiLSTM-PINN models. Table III illustrates the superiority of BiLSTM-PINN over FCNN and BiLSTM on most metrics in terms of mean(e) and median(e). We remark that $\sigma_{\text{std}}(e)$ is often bigger than mean(e), demonstrating that outliers are, skewing the mean, similarly to the RMSE analysis in Fig. 4. In this case, the median might be more informative as an overall evaluation of the performance. Looking at the RMSE medians, we observe that the BiLSTM-PINN has a median error of 0.0256 A (≈ 9 times lower than the FCNN) in comparison to 0.0509 A (≈ 4.5 times lower than the FCNN), and 0.2311 A for the BiLSTM and the FCNN, respectively. We similarly notice that standard deviation values are 0.2782 A, 0.4236 A, and 0.7367 A for the BiLSTM-PINN, BiLSTM, and FCNN. The BiLSTM-PINN

TABLE III: Learning-based model performance metric comparison

Metric	`FCNN` mean$(e) \pm \sigma_{\text{std}}(e)$ median(e)	`BiLSTM` mean$(e) \pm \sigma_{\text{std}}(e)$ median(e)	`BiLSTM-PINN` mean$(e) \pm \sigma_{\text{std}}(e)$ median(e)
$\mathcal{L}_{\text{RMSE}}$ (A)	0.3793 ± 0.7367 0.2311	0.1161 ± 0.4236 0.0509	0.0593 ± 0.2782 0.0256
Rise time (s)	0.0004 ± 0.0009 0.0000e-5	7.7094e-5 ± 0.0004 0.0000e-5	3.5621e-5 ± 0.0003 0.0000e-5
Transient time (s)	0.0036 ± 0.0037 0.0022	0.0034 ± 0.0032 0.0024	0.0022 ± 0.0021 0.0015
Settling time (s)	0.0020 ± 0.0026 0.0010	0.0016 ± 0.0023 0.0007	0.0019 ± 0.0031 0.0003
Settling minimum (A)	0.3131 ± 0.9630 0.1892	0.1205 ± 1.5329 0.0438	0.0685 ± 0.9676 0.0281
Settling maximum (A)	0.5400 ± 1.6805 0.2034	0.0747 ± 0.3949 0.0277	0.1131 ± 0.8193 0.0299
Overshoot (%)	– 7.4707	– 1.8219	– 1.7154
Undershoot (%)	– 0.0000e-5	– 0.0000e-5	– 0.0000e-5
Peak (A)	0.5373 ± 1.6714 0.2034	0.0741 ± 0.3933 0.0277	0.1100 ± 0.7640 0.0299
Peak time (s)	0.0027 ± 0.0031 0.0008	0.0017 ± 0.0029 0.0002	0.0029 ± 0.0037 0.0005

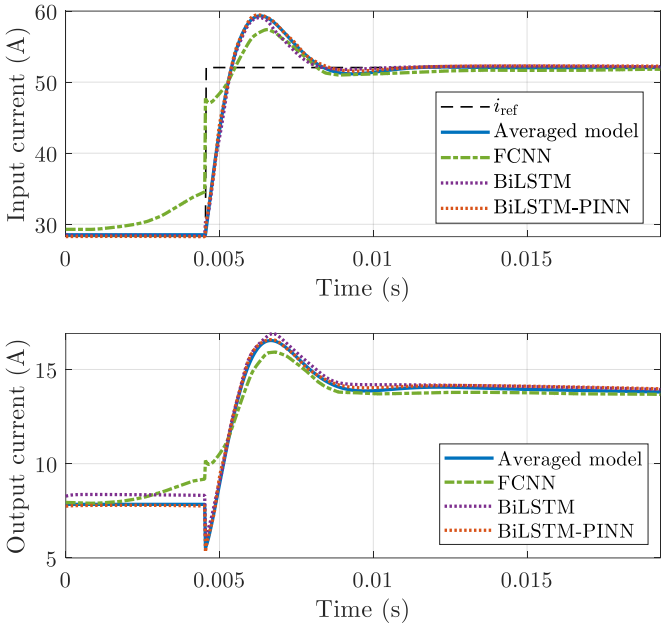
and BiLSTM standard deviation are approximately 2.6 and 1.7 times smaller than the FCNN, making the predictions not only generally more accurate, but also more consistent. Those results are consistent with [13], which showed that recurrent architecture tends to outperform non-recurrent ones. However, unlike [11], our physics-informed FCNN cannot reproduce as accurately the waveform when using $\mathcal{L}_{\text{PBE}}$ to integrate physics to the loss. Over all metrics, the BiLSTM-PINN main limitations are the overshoot and the undershoot. To better understand the problem, we extract four distinct cases from the test dataset, two with average performance and two with a high overshooting or undershooting error and plot their responses. Figs. 5 and 6, respectively, represent the two cases with average performance and with a high overshooting or undershooting error. The BiLSTM-PINN tends to follow accurately the averaged model in both cases while the FCNN exhibits some inaccuracies, e.g., Fig. 6. We notice that overshooting or undershooting error occurs when there is a small perturbations on i_{ref} and the final value is close to zero as observed in Fig. 6. These cases tend to produce outliers in terms of overshooting and undershooting errors given the division to a value close to zero, skewing the distribution and making the mean and standard deviation less representative. We omitted these values from Table III for this reason, keeping only the median values as more insightful in this case. This phenomenon is also amplified by the minimum and maximum values on $I_{\text{L}}(t = 0)$ in our dataset. For perturbations on i_{ref} close to $|I_{\text{L}}(t = 0)|$, a reduced number of operating points exist for training in our dataset due to the use of a uniform distribution, thus increasing the error of our model in these cases. In sum, the results illustrate that our models may achieve good performance inside the parameter intervals defined in Table II, but lose accuracy when operating closer to their minimum and maximum values due to having fewer available operating training points.

V. Conclusions

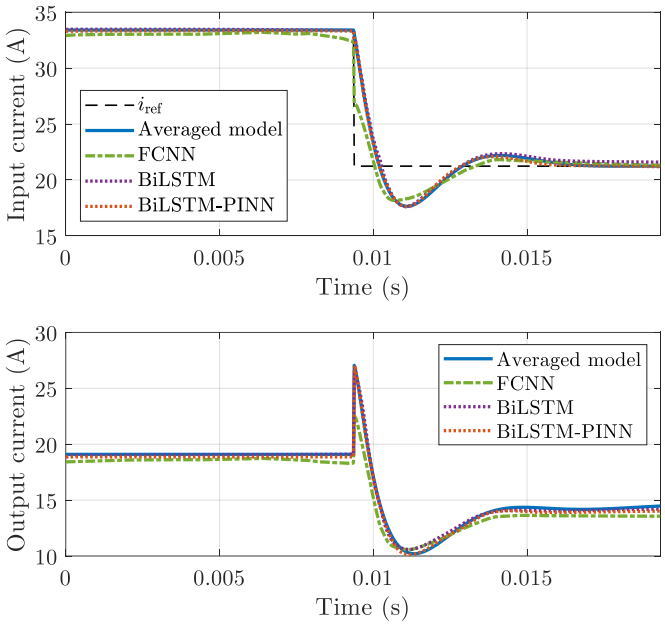
This work proposes a new physics-informed BiLSTM method to model power electronics converters. We evaluate the proposed BiLSTM-PINN by generating a dataset using the steady-state equations of a dc-dc boost converter along with a dc-averaged closed-loop model of the converter in MATLAB® Simulink. We also train a physics-informed FCNN and a data-driven BiLSTM to establish a performance comparison with a conventional non-recurrent neural network architecture. For both architectures, we tune the hyperparameters using `HyperOpt`. We then compare both methods by visualizing the RMSE distribution and quantifying their accuracy for various metrics using the `stepinfo` function from MATLAB®.

Our numerical study illustrates the ability of learning-based models to predict the response of power electronics converters. The BiLSTM-PINN model offers improved performance over the FCNN's and the data-driven BiLSTM and may be a good alternative to classical methods in some applications as a means of simplifying and accelerating real-time simulations. These results can improve the capabilities of commercial EMT simulation software by allowing the simulation of large-scale power systems through simplification of certain complex components, and for blackbox modelling of off-the-shelf converters.

Several limitations must be addressed before deployment in practical settings. Notably, the proposed model is based

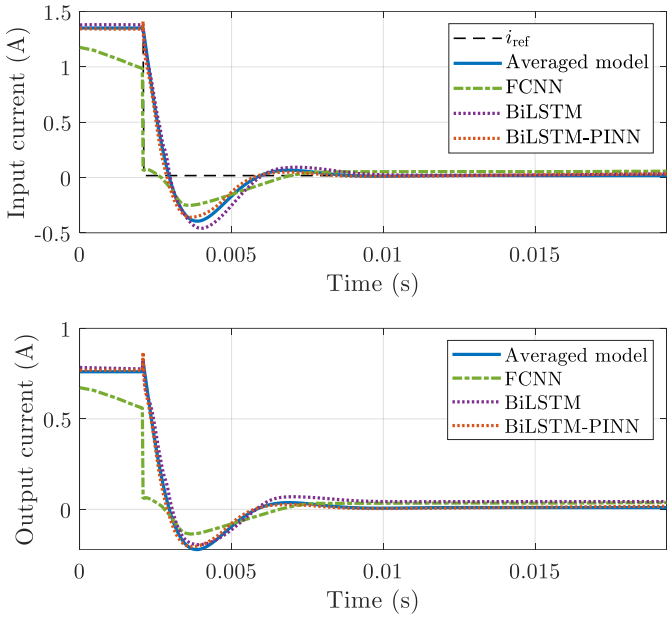


(a) Input and output current signals for $\approx 83\%$ increase in i_{ref}.

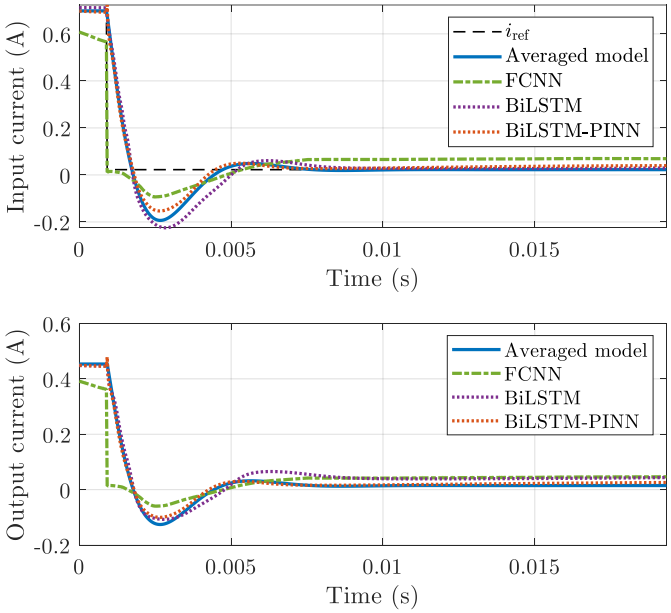


(b) Input and output current signals for $\approx 36\%$ decrease in i_{ref}.

Fig. 5: Predictions for two operating points achieving close to or lower median performance in all metrics evaluated in Table III.



(a) Input and output currents signals for $\approx 99\%$ decrease on i_{ref}.



(b) Input and output currents signals for $\approx 98\%$ decrease on i_{ref}.

Fig. 6: Predictions for two operating points resulting in some of the highest overshooting error (a) or undershooting error (b) among all operating points tested in Table III (outliers).

on an averaged large-signal representation, which overlooks high-frequency behaviour in predicted signals and neglects non-idealities from diode D_1 and switch S. Moreover, the dataset relies on an unsaturated controller, limiting the validity of the model by excluding non-linearities in the control. However, our observations indicate that omitting K_p and K_i from the feature vector $\mathbf{x}_t$ did not significantly impact the performance metrics. This suggests that other controllers could be included into the dataset, as only the control signal remains. While we are confident that this methodology can be applied to other simple dc-dc topologies, such as the buck converter [11], [13], extending it to more complex topologies is a topic of future work.

Future directions will focus on benchmarking the proposed method against physics-based approaches in terms of computational efficiency to quantify the potential gain in simulation speed. Expanding the framework to include other converter operating modes, such as protective states and fault-tolerant operation, is also a key direction to improve model generalization and extend its applications.

REFERENCES

[1] S. Peyghami, F. Blaabjerg, and P. Palensky, "Incorporating power electronic converters reliability into modern power system reliability analysis," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 9, no. 2, pp. 1668–1681, 2021.

[2] S. Peyghami, P. Palensky, and F. Blaabjerg, "An overview on the reliability of modern power electronic based power systems," *IEEE Open Journal of Power Electronics*, vol. 1, pp. 34–50, 2020.

[3] J. A. Martinez-Velasco, *Transient analysis of power systems: solution techniques, tools and applications*. John Wiley & Sons, 2014.

[4] S. Vazquez, J. I. Leon, L. G. Franquelo, J. Rodriguez, H. A. Young, A. Marquez, and P. Zanchetta, "Model predictive control: A review of its applications in power electronics," *IEEE Industrial Electronics Magazine*, vol. 8, no. 1, pp. 16–31, 2014.

[5] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019.

[6] A. Wunderlich and E. Santi, "Digital twin models of power electronic converters using dynamic neural networks," in *2021 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2021, pp. 2369–2376.

[7] M. Berger, I. Kocar, H. Fortin-Blanchette, and C. Lavertu, "Large-signal modeling of three-phase dual active bridge converters for electromagnetic transient analysis in dc grids," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 6, pp. 1684–1696, 2019.

[8] L. Salazar and G. Joos, "Pspice simulation of three-phase inverters by means of switching functions," *IEEE Transactions on Power Electronics*, vol. 9, no. 1, pp. 35–42, 1994.

[9] R. D. Middlebrook and S. Cuk, "A general unified approach to modelling switching-converter power stages," in *1976 IEEE Power Electronics Specialists Conference*, 1976, pp. 18–34.

[10] S. R. Sanders, J. M. Noworolski, X. Z. Liu, and G. C. Verghese, "Generalized averaging method for power conversion circuits," *IEEE Transactions on Power Electronics*, vol. 6, no. 2, pp. 251–259, 1991.

[11] H. Ge, Z. Huang, and Z. Huang, "Modeling methodology based on fast and refined neural networks for non-isolated dc–dc converters with configurable parameter settings," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 2, pp. 617–628, 2023.

[12] G. Rojas-Dueñas, J.-R. Riba, and M. Moreno-Eguilaz, "Nonlinear least squares optimization for parametric identification of DC–DC converters," *IEEE Transactions on Power Electronics*, vol. 36, no. 1, pp. 654–661, 2020.

[13] P. Qashqai, K. Al-Haddad, and R. Zgheib, "Modeling power electronic converters using a method based on long-short term memory (LSTM) networks," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 4697–4702.

[14] G. Rojas-Dueñas, J.-R. Riba, K. Kahalerras, M. Moreno-Eguilaz, A. Kadechkar, and A. Gomez-Pau, "Black-Box Modelling of a DC-DC Buck Converter Based on a Recurrent Neural Network," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, 2020, pp. 456–461.

[15] Q. Li, H. Bai, E. Breaz, R. Roche, and F. Gao, "ANN-aided data-driven IGBT switching transient modeling approach for FPGA-based real-time simulation of power converters," *IEEE Transactions on Transportation Electrification*, vol. 9, no. 1, pp. 1166–1177, 2023.

[16] H. S. Krishnamoorthy and T. Narayanan Aayer, "Machine learning based modeling of power electronic converters," in *2019 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2019, pp. 666–672.

[17] S. Zhao, Y. Peng, Y. Zhang, and H. Wang, "Parameter estimation of power electronic converters with physics-informed machine learning," *IEEE Transactions on Power Electronics*, vol. 37, no. 10, pp. 11 567–11 578, 2022.

[18] F. Bragone, K. Morozovska, P. Hilber, T. Laneryd, and M. Luvisotto, "Physics-informed neural networks for modelling power transformer's dynamic thermal behaviour," *Electric Power Systems Research*, vol. 211, p. 108447, 2022.

[19] R. Nellikkath, I. Murzakhanov, S. Chatzivasileiadis, A. Venzke, and M. K. Bakhshizadeh, "Physics-informed neural networks for phase locked loop transient stability assessment," *Electric Power Systems Research*, vol. 236, p. 110790, 2024.

[20] Y. Li, C. Xue, F. Zargari, and Y. R. Li, "From Graph Theory to Graph Neural Networks (GNNs): The Opportunities of GNNs in Power Electronics," *IEEE Access*, vol. 11, pp. 145 067–145 084, 2023.

[21] D. Danopoulos, I. Stamoulias, G. Lentaris, D. Masouros, I. Kanaropoulos, A. K. Kakolyris, and D. Soudris, "LSTM Acceleration with FPGA and GPU Devices for Edge Computing Applications in B5G MEC," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Cham: Springer International Publishing, 2022, pp. 406–419.

[22] S. D. Nagarale and B. P. Patil, "Accelerating AI-Based Battery Management System's SOC and SOH on FPGA," *Applied Computational Intelligence and Soft Computing*, vol. 2023, no. 1, p. 2060808, 2023.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[24] V. Rybalkin, A. Pappalardo, M. M. Ghaffar, G. Gambardella, N. Wehn, and M. Blott, "FINN-L: Library extensions and design trade-off analysis for variable precision lstm networks on fpgas," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, 2018, pp. 89–897.

[25] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International Conference on Machine Learning*. PMLR, 2013, pp. 115–123.

[26] D. Maksimovic, A. Stankovic, V. Thottuvelil, and G. Verghese, "Modeling and simulation of power electronic converters," *Proceedings of the IEEE*, vol. 89, no. 6, pp. 898–912, 2001.

[27] R. W. Erickson and D. Maksimovic, *Fundamentals of power electronics*. Springer International Publishing, 2020.

[28] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5929–5955, Dec 2020.

[29] D. Kingma and J. L. Ba, "Adam: A method for stochastic gradient descent," in *ICLR: International Conference on Learning Representations*. ICLR US., 2015, pp. 1–15.

[30] The MathWorks Inc., "MATLAB version: 24.1.0.2689473 (r2024a) update 6," Natick, Massachusetts, United States, 2024. [Online]. Available: https://www.mathworks.com

APPENDIX

We determine K_p and K_i using the *algebra on the graph* technique from [26]. To do so, we first set the cross-over frequency $\omega_{\phi_\mathrm{m}} = 2\pi f_{\phi_\mathrm{m}}$ where $f_{\phi_\mathrm{m}} = 255$ Hz (more than 200 times smaller than the switching frequency f_s) and the phase-margin $\phi_\mathrm{m} = 50°$ as a arbitrary compromise between stability margin, overshoot, and settling time. From those desired values, we first calculate the phase and magnitude of $G_\mathrm{c}(s)$ given by $\angle G_\mathrm{c}(\omega_{\phi_\mathrm{m}}) = -(\angle G(\omega_{\phi_\mathrm{m}}) + (180° - \phi_\mathrm{m}))$ and $|G_\mathrm{c}(\omega_{\phi_\mathrm{m}})|_\mathrm{dB} = -|G(\omega_{\phi_\mathrm{m}})|_\mathrm{dB}$. We then find the ratio α and compute K'_p of an intermediate controller with $K'_\mathrm{i} = 1$ as

$$\begin{aligned}\alpha = \frac{K_\mathrm{i}}{K_\mathrm{p}} &= \frac{\omega_{\phi_\mathrm{m}}}{\tan(\angle G_\mathrm{c}(\omega_{\phi_\mathrm{m}}) + 90°)}\\ K'_\mathrm{p} &= \frac{1}{\alpha}.\end{aligned}$$

We finally calculate the magnitude of $G'_\mathrm{c}(s)$ to find K_p and K_i, and verify if a PI-controller with those parameters is feasible:

$$\begin{aligned}|G'_\mathrm{c}(\omega_{\phi_\mathrm{m}})|_\mathrm{dB} &= 20\log\left[\frac{\sqrt{(\omega_{\phi_\mathrm{m}}K'_\mathrm{p})^2 + (K'_\mathrm{i})^2}}{\omega_{\phi_\mathrm{m}}}\right]\\ K_i &= 10^{-|G(\omega_{\phi_\mathrm{m}})|_\mathrm{dB} - |G'_\mathrm{c}(\omega_{\phi_\mathrm{m}})|_\mathrm{dB}/20}\\ K_\mathrm{p} &= \frac{K_\mathrm{i}}{\alpha}.\end{aligned}$$