

Highlights

A Comparative Study of PINN and CPINN as Control-oriented Models for PEM Electrolyzers

Milad Sabamehr, Carles Batlle, Maria Serra

- PINN and CPINN are compared as control-oriented models of a PEM electrolyzer ECM.
- CPINN generalizes better than PINN to unseen current profiles and initial conditions.
- CPINN remains accurate beyond the training time horizon; PINN drifts and degrades.
- Adding a data-loss term improves robustness of both models under parameter mismatch.
- CPINN is more robust than PINN to noisy current inputs.

A Comparative Study of PINN and CPINN as Control-oriented Models for PEM Electrolyzers

Milad Sabamehr^{a,*}, Carles Batlle^b and Maria Serra^a

^a*Institut de Robòtica i Informàtica Industrial (IRI), CSIC-UPC, Llorens i Artigas 4-6, Barcelona, 08028, Spain*

^b*Departament de Matemàtiques, Institut d'Organització i Control, EPSEVG, Universitat Politècnica de Catalunya (UPC), Vilanova i la Geltrú, 08800, Spain*

ARTICLE INFO

Keywords:

Physics-Informed Neural Networks
Conditional Physics-Informed Neural Networks
Proton Exchange Membrane Electrolyzer
Equivalent Circuit Model
Hydrogen Production
Dynamic System Modeling

ABSTRACT

The growing demand of green hydrogen produced from renewable energy has increased the interest in Proton Exchange Membrane (PEM) electrolyzers, which are able to adapt to the renewable power dynamics. Accurate dynamic models of PEM electrolyzers operating under varying conditions have become necessary for the improvement of the technology. On the contrary, simplified models are necessary in order to design and deploy the control algorithms that allow the PEM electrolyzers to react to changing conditions and requirements in real time. In this sense, models based on physical laws can be substituted by Equivalent Circuit Models (ECM) that offer a computationally efficient representation of the electrolyzers. On the other hand, data-driven models based on Neural Networks (NN) are able to incorporate nonlinear or unknown dynamics, although they often struggle to generalize outside the training domain. Physics-Informed Neural Networks (PINN) propose a hybrid solution by embedding governing physical equations into the learning process. However, their performance in long-horizon prediction remains limited. This work presents a comparative study of conventional PINN and Conditional PINN (CPINN) for modeling a second-order ECM of a PEM electrolyzer. Although both approaches are trained using physics-based losses derived from the system's differential equations, the CPINN framework incorporates conditioning on the operating current and the initial capacitors voltage to learn a family of dynamic solutions. The trained models are evaluated by test data in the training domain as well as in extended-time scenarios beyond the training interval. Results based on several evaluation metrics (MSE, RMSE, and MAE, MBE, and R^2) demonstrate that CPINNs accuracy significantly improves that of PINNs, which exhibit larger and increasing deviation when extrapolating beyond the training time domain. These findings highlight the effectiveness of conditional physics-informed learning for dynamical modeling of electrolyzers and motivate future integration of CPINN models with model predictive control strategies for real-time optimal control of PEM electrolyzers.

1. Introduction

The necessary transition toward sustainable energy systems has accelerated the global interest in green hydrogen, a critical vector for the storage of renewable energy that provides flexibility to the energy networks. Among other candidates for green hydrogen production, PEM electrolyzers have emerged as a leading candidate in different aspects such as high efficiency, compact design, and rapid dynamic response (Gosu et al., 2024; Hidouri et al., 2024).

For the improvement of the PEM electrolysis technology, proper modeling is necessary, which will drive to optimized performance and system-level integration with renewable sources. Under dynamic operation (e.g. fast current variations given by the renewable sources), where transient fluid dynamics and thermal phenomena can strongly affect voltage response and efficiency, obtaining proper models of PEM electrolyzers becomes more challenging (Sayed-Ahmed et al., 2024; Asiaban et al., 2024). Electrical equivalent circuit models (ECMs) have been widely considered because they capture the dominant dynamics while remaining tractable for identification and real-time use (Ratib et al., 2024; Guilbert and Vitale, 2019). In Dang et al. (2022) different equivalent circuit models were used to describe and analyze the transient behaviors of PEM electrolyzers under different operating conditions, providing insights into their control and durability under fluctuating operating conditions. In particular, second-order ECMs (e.g., two RC branches plus ohmic and reversible terms) have proven proper reproduction of the characteristic transient polarization behavior of PEM electrolyzers subjected to time-varying currents (Ratib et al., 2024; Guilbert and Vitale,

*Corresponding author

 sabamehrmilad@gmail.com (M. Sabamehr)

ORCID(s):

2019). To improve the hydrogen production efficiency by identifying the optimal operating frequency for pulsed electrolysis, an ECM is proposed by Kim et al. (2022), which provides a simplified and effective way to represent the electrochemical and physical processes occurring within the electrolyzer. However, despite the usefulness of ECMs, they require accurate parameterization, and classical identification techniques can become sensitive to measurement noise and lead to error, resulting in inaccurate parameter values and, consequently, less reliable models (Ratib et al., 2024). These limitations have motivated the use of hybrid strategies that combine the physical insight given by the ECM with the real behavior given by data-driven models.

Zhang and Chan (2022) focused on real-time operational data to refine the dynamical model of an electrolyzer cell. They captured its dynamical behavior under different operational conditions, implementing Long Short-Term Memory (LSTM) and Artificial Neural Network (ANN) approaches for predicting dynamic behavior of the electrolyzer. In Raman et al. (2025), data-driven models like Support Vector Regression (SVR), Extreme Gradient Boosting (XGB) and ANN were used to predict the performance and analyze the sensitivity of transport properties in a PEM electrolyzer. The results showed that ANN and XGB outperform SVR in predicting current density and cell efficiency. While these data-driven models can achieve high accuracy within the training domain, they usually suffer from poor extrapolation capability and lack physical interpretability.

To overcome these limitations, the work of Raissi et al. (2019) introduced Physics Informed Neural Networks (PINN) that are capable of solving forward and inverse problems with limited data by embedding the physical equation directly into the learning process. The strategy aims at the improvement of the generalization capacity of the neural networks (NN) and the reduction of their necessity of experimental data. In this PINN framework, the model is additionally guided by the boundary. A boundary loss enforces consistency between the network predictions and the prescribed boundary conditions by penalizing deviations at the boundary points of the domain in a similar way as the data loss quantifies the discrepancy and enforces consistency between the predicted outputs and the available measured (or reference) data at observed points. These penalty terms are incorporated into a global loss function that also includes the physics-based residual loss. The optimization algorithm iteratively updates the network parameters to minimize the aggregated loss and achieve a solution that simultaneously satisfies the governing equations, the boundary conditions and the observed data.

PINNs have been applied in various sectors, including healthcare, biomedical engineering, industrial and energy systems (Raissi et al., 2019; Cuomo et al., 2022). With respect to the application of PINNs to electric and power systems, several works can be found in the literature. The use of PINNs to solve the dynamics of a power system, specifically the swing equation for a single machine infinite bus system, was discussed by Misyris et al. (2020). Results showed the prediction of the dynamic states of the power system and the estimation of uncertain parameters of the system, such as inertia and damping parameters, with high accuracy. In Stiasny et al. (2024) a method called PINNSim was developed as a simulator for power system dynamics based on PINNs. This method utilizes PINNs to learn the dynamics of individual power system components, showing considerable scalability and flexibility when compared to traditional numerical integration methods. In Hurt and Toliyat (2025) a PINN approach was used to estimate the parameters of an electrical circuit, demonstrating also high precision in parameter estimation. As examples of application of PINNs to the specific field of electric circuit models one can cite Shahbaz et al. (2025), where a novel version of PINN was developed for efficient and interpretable electrical circuit analysis. In the study of Taj (2024), PINNs demonstrated high precision in current prediction for simple circuits and also in estimating the parameters of simple circuits, while Dey and Mallik (2025) introduced a new approach to optimize the efficiency of dual-active bridge DC-DC converters by PINNs to estimate key circuit parameters, like series inductance and parasitic resistance.

Focusing on the application of PINNs to PEM electrolyzers, Zerrougui et al. (2025) was able to predict temperature fluctuations, showing robust performance when compared to traditional recurrent neural networks such as LSTM, and improving prediction accuracy, especially under data-scarce and dynamic conditions. In (Ko et al., 2025) prediction of remaining useful life of proton exchange membrane fuel cells was undertaken by incorporating governing equations that describe membrane and catalyst degradation in a PINN framework, showing significant improved performance with respect to the previous strategies.

Related to the application of PINN for the control of dynamic systems, Arnold and King (2021) developed a Model Predictive Control (MPC) scheme based on PINN. A PINN-based state-space model was used to predict the states time evolution and optimize the control inputs for real-time reference tracking, showing improved effectiveness for state estimation in uncertain conditions when compared to traditional numerical solutions. In a parallel study on PINN in control, Gokhale et al. (2022) proposed two novel PINN architectures, PhysNet and PhysReg MLP (Physics

Regularized Multi-Layer Perceptron), applied them to control-oriented thermal modeling of buildings, and created an efficient controller that optimizes energy consumption with improved prediction accuracy for long prediction horizons.

Conditional Physics Informed Neural Network (CPINN), which adds conditions to the PINN model through the inclusion of new input parameters that influence the system behavior, were introduced in Kovacs et al. (2022). A special kind of CPINN, called Physics Informed Neural Network for Control (PINNC), was considered in Antonelo et al. (2024). By constructing a PINN that is able to predict the output over short time periods corresponding to given initial conditions and constant control input, and using the trained system in a self-loop time roll out configuration, it was shown that it is possible to extend the prediction to long-range time intervals for arbitrary time-varying inputs and without prediction degradation. Furthermore, the use of PINNC as the embedded model for Model Predictive Control (MPC) was also demonstrated.

In this study, the generalization capacity and other features of PINN and CPINN applied to an ECM of a PEM electrolyzer are compared. The ECM model used in this work is described in Mao et al. (2024). Since the PEM electrolyzer is driven by the electrical current, the neural network must take the current values as inputs, in addition to the time values for which we want to predict the state of the system. The inclusion of control inputs is naturally enforced in the CPINN framework, but in the case of PINN it must be explicitly added to the architecture, departing from the usual formulation of PINNs for non-control purposes.

A desirable characteristic of PINN applied to control systems is that they should be able to cope with unknown parameters in the differential equations that describe the physical system. Since a definite set of parameters must be used for the training, the results of the trained system will in general differ from the ones obtained using a more accurate model or measured in an experimental setup. This problem can be minimized if one considers a hybrid training scheme, where the knowledge of the true parameter values is indirectly incorporated through a data loss term, which compares the output of the neural network with the true results and tries to minimize the total difference, together with the losses provided by the residual of the physics-based differential equations.

The paper is structured as follows: Section 2 provides the PEM electrolyzer model through an equivalent electrical circuit and discusses the governing equations and the physical interpretation of the adopted model. Section 3 describes the theoretical foundations of PINN and CPINN, and describes their application to the equivalent circuit model of a PEM electrolyzer. Results are presented in Section 4, using suitable selected metrics, where the prediction accuracy and generalization capacity of PINN and CPINN are compared. Section 4.3 considers the case of hybrid training for unknown parameters with the addition of data loss, both for PINN and for CPINN. Section 4.4 analyses the effect of noise both for the PINN and the CPINN cases. Finally, Section 5 concludes the paper with key findings and highlights the potential future works based on this study.

2. Second-Order Equivalent Circuit Model of the PEM Electrolyzer

The second-order ECM shown in Figure 1 to model a PEM electrolyzer was considered in Mao et al. (2024). This model has been shown to accurately capture the transient voltage response of PEM electrolyzers under time-varying current excitation, and it is the one used in this work.

The selected equivalent circuit consists of different components, including a reversible voltage source E_{rev} , an ohmic resistance R_{ohm} , and two parallel branches (R_1, C_1) and (R_2, C_2) . The two RC branches represent PEM processes evolving with different time dynamics and R_{ohm} represents the ohmic voltages losses in the membrane and electrical contacts. The state variables of the model are defined as the voltages across the two RC branches, $V_1(t)$ and $V_2(t)$, the input is the applied current $I(t)$, and the output is the terminal cell voltage $V(t)$. The parameters of the equivalent circuit are directly adopted from the literature (Mao et al., 2024) and are treated as known constant values throughout this work, except in Section 4.3.

The differential equations describing the circuit in Figure 1 are

$$R_1 C_1 \frac{dV_1}{dt} - R_1 I(t) + V_1 = 0, \quad (1)$$

$$R_2 C_2 \frac{dV_2}{dt} - R_2 I(t) + V_2 = 0, \quad (2)$$

while the total voltage is given by

$$V(t) = E_{rev} + R_{ohm} I(t) + V_1(t) + V_2(t). \quad (3)$$

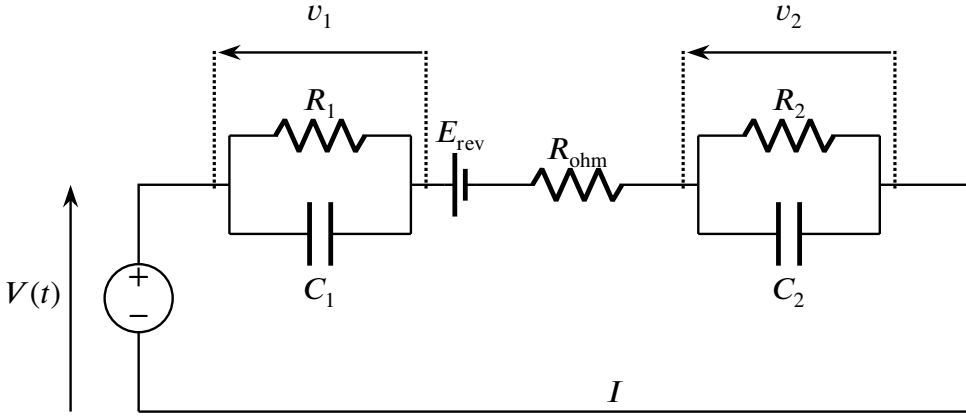


Figure 1: Second-order equivalent circuit model of the PEM electrolyzer.

Table 1

Equivalent circuit model parameters of the PEM electrolyzer, taken from (Mao et al., 2024).

Parameter	Symbol	Value	Unit
Reversible voltage	E_{rev}	1.24	V
Ohmic resistance	R_{ohm}	0.002275	Ω
Resistance (RC branch 1)	R_1	0.001488	Ω
Capacitance (RC branch 1)	C_1	407.4	F
Resistance (RC branch 2)	R_2	0.001319	Ω
Capacitance (RC branch 2)	C_2	45.94	F
Time constant (RC branch 1)	$\tau_1 = R_1 C_1$	0.6062112	s
Time constant (RC branch 2)	$\tau_2 = R_2 C_2$	0.06059486	s

The power supplied to the electrolyzer is $I(t)V(t)$. However, not all this power can be converted into chemical energy of hydrogen because of the different existing losses.

The parameters used in this paper, taken from Mao et al. (2024), are given in Table 1. The two different time constants reflect the existence of fast and slow dynamics, which enable the model to correctly reproduce the voltage time response under different operating condition, particularly during rapid current variations.

3. PINN and CPINN models

3.1. Physics Informed Neural Networks

PINNs are neural networks that, in the most general form, integrate experimental data with known differential equations to enhance the performance of the neural network. A hybrid loss function composed of data, boundary and initial conditions, and physics terms is minimized during the training process (Zerrougui et al., 2025; Karniadakis et al., 2021).

The PINN neural network consists of multiple layers, where each layer performs an affine transformation followed by a nonlinear activation function. In each layer, the input vector

$$\mathbf{x} = [x_1, x_2, \dots, x_i]^\top$$

is mapped to the output vector

$$\mathbf{y} = [y_1, y_2, \dots, y_j]^\top$$

according to

$$y_j = r \left(\sum_i w_{i,j} x_i + b_j \right), \quad (4)$$

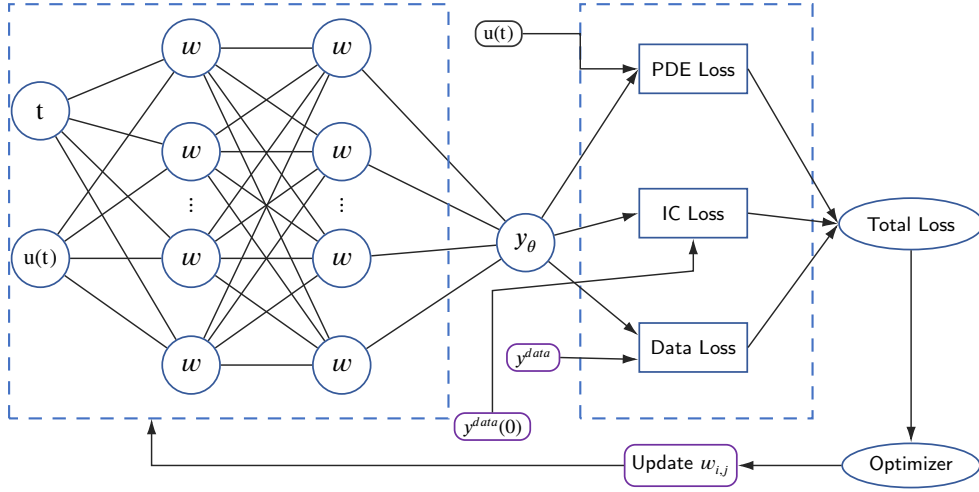


Figure 2: Schematic illustration of a physics-informed neural network for the modeling of a dynamic system with a control input variable. The inputs of the neural network are time, t , and the system control input, $u(t)$, and the output of the neural network is the predicted system response $y_\theta(t) \in \mathbb{R}^n$; the single output node represents the output layer of n neurons, one per predicted variable. Physical knowledge is embedded into the learning process through the residual associated to the governing differential equations that describe the system dynamics, PDE Loss or physics loss. The input $u(t)$ is also fed into the PDE loss function. The term $y^{\text{data}}(0)$ denotes the known initial value of the system state, which is compared with the network prediction $y_\theta(0)$ to compute the IC loss. Usually, if available, a data loss term is included to deal with the errors of the PDE model. These three losses are combined into a total loss function, which is minimized using a gradient-based optimizer that iteratively updates the weights $w_{i,j}$ and b_j to improve the network prediction accuracy.

where $w_{i,j}$ and b_j denote the trainable weights and biases respectively, and $r(\cdot)$ represents the activation function introducing nonlinearity into the network. When the modelled system has a forcing function or control input $u(t)$, the neural network will have as inputs both time and the control input. Note that the network operates pointwise: for each time instant t_j , it receives the scalar time t_j and the control input $u(t_j)$, and returns the predicted state vector $y_\theta(t_j) \in \mathbb{R}^n$, where n is the number of predicted variables. Figure 2 displays a scheme of a PINN for this case.

The expressions of the different loss functions of the PINN as implemented in this work are detailed in the following paragraphs. Since the PINNs in this work are aimed to be control-oriented models, system input control variables $u(t)$ are included.

Physics Loss ($\mathcal{L}_{\text{phys}}$) Following Raissi et al. (2019), a PINN is intended to approximate the solution to an Ordinary Differential Equation (ODE) for $y(t)$

$$y_t + \mathcal{N}(y, u) = 0, \quad (5a)$$

$$y(0) = y_0 \quad (5b)$$

where

$$y_t = \frac{dy}{dt} \quad (6)$$

and where $\mathcal{N}[\cdot]$ denotes a nonlinear or linear function representing the underlying physical laws. The objective is to construct an approximation of $y(t)$, $y_\theta(t)$, that approximately satisfies (5), including the prescribed initial conditions.

In the PINN framework, the neural network approximation $y_\theta(t)$ is substituted into the governing operator to form the residual. The physics loss is defined as the mean squared residual evaluated at a set of N_r collocation points $\{t_j\}_{j=1}^{N_r}$ sampled within the domain $[0, T]$:

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_r} \sum_{j=1}^{N_r} \left| \frac{dy_{\theta}}{dt}(t_j) + \mathcal{N}[y_{\theta}(t_j), u(t_j)] \right|^2, \quad (7)$$

where

- N_r denotes the number of collocation points.
- t_j denotes the j -th collocation point in the time interval $[0, T]$.
- $y_{\theta}(t_j)$ is the output of the neural network parameterized by θ evaluated at that time t_j .

Initial Condition Loss ($\mathcal{L}_{ic}$) The minimization of the Initial Condition Loss in (8) enforces the known initial state of the dynamical system penalizing any mismatch between the neural network prediction and the known initial values. For ODEs, the solution trajectory must start from the prescribed initial condition.

$$\mathcal{L}_{ic} = \left| y_{\theta}(t=0) - y^{\text{data}}(0) \right|^2 \quad (8)$$

Data Loss ($\mathcal{L}_{\text{data}}$) The data loss term penalizes the discrepancy between the neural network predictions and the available reference (ground-truth) measurements. Let $\{t_i\}_{i=1}^{N_d}$ denote a set of N_d observed data points within the time domain $[0, T]$. Data loss is defined as the mean squared error between the predicted and the true values, as developed in (9).

$$\mathcal{L}_{\text{data}} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left| y_{\theta}(t_i) - y^{\text{data}}(t_i) \right|^2, \quad (9)$$

where

- N_d denotes the number of observed data points.
- t_i denotes the i -th observation time point in the time interval $[0, T]$.
- $y^{\text{data}}(t_i)$ is the ground truth value at time t_i (from experiments or simulations).
- $y_{\theta}(t_i)$ denotes the neural network prediction parameterized by θ evaluated at time t_i .

The total loss function is defined as

$$\mathcal{L}_{\text{total}} = \lambda_{\text{phys}} \mathcal{L}_{\text{phys}} + \lambda_{ic} \mathcal{L}_{ic} + \lambda_{\text{data}} \mathcal{L}_{\text{data}}, \quad (10)$$

where λ_{phys} , λ_{ic} , and λ_{data} are weighting coefficients that control the relative contributions of the physics, initial-condition and data-fidelity terms, respectively. Then, in PINN, minimizing $\mathcal{L}_{\text{data}}$ ensures that the learned solution remains consistent with the available measurements, while the physics-loss terms defined in (7) enforce compliance with the governing equations. The combined minimization of data and physics losses, together with the initial condition losses, enables improved accuracy, robustness, and generalization under unseen operating conditions (Raissi et al., 2019).

The parameters θ of the trained network are obtained by solving

$$\theta^* = \arg \min_{\theta} \mathcal{L}_{\text{total}} \quad (11)$$

3.2. PINN model of a PEM electrolyzer equivalent circuit

For the modeling of the PEM electrolyzer ECM, the PINN learns the mapping from time and current to the internal state voltages and the terminal cell voltage. For this model, the PINN predicted output is $y_\theta(t_i) = (V_{\theta,1}(t_i), V_{\theta,2}(t_i), V_\theta(t_i))$.

The network is implemented as a fully connected feedforward neural network. Hyperbolic tangent activation functions are used for smooth differentiability, which is essential for Automatic Differentiation (AD).

The residuals associated with the differential equations are defined as

$$RES_1(t) = R_1 C_1 \frac{dV_{\theta,1}(t)}{dt} - R_1 I(t) + V_{\theta,1}(t), \quad (12)$$

$$RES_2(t) = R_2 C_2 \frac{dV_{\theta,2}(t)}{dt} - R_2 I(t) + V_{\theta,2}(t), \quad (13)$$

$$RES_3(t) = V_\theta(t) - E_{\text{rev}} - R_{\text{ohm}} I(t) - V_{\theta,1}(t) - V_{\theta,2}(t). \quad (14)$$

Residuals RES_1 and RES_2 enforce the dynamics, while RES_3 enforces voltage consistency. Automatic differentiation (AD), to obtain the time derivatives.

According to equation (7), the physics loss is

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left[(RES_1(t_i))^2 + (RES_2(t_i))^2 + (RES_3(t_i))^2 \right]. \quad (15)$$

The initial conditions of the state variables are enforced by minimizing the initial condition loss in (16), in accordance to (8),

$$\mathcal{L}_{\text{ic}} = \frac{1}{2} \left[(V_{\theta,1}(0) - V_1(0))^2 + (V_{\theta,2}(0) - V_2(0))^2 \right]. \quad (16)$$

In this work we do not consider data loss, except in Section 4.3. The reason is that we aim to examine the extent to which data-free networks can perform, assuming that including data, both PINN and CPINN will improve similarly. Therefore, the total loss function of the PINN model in this work will be

$$\mathcal{L}_{\text{total}} = \lambda_{\text{phys}} \mathcal{L}_{\text{phys}} + \lambda_{\text{ic}} \mathcal{L}_{\text{ic}}, \quad (17)$$

with $\lambda_{\text{phys}} = \lambda_{\text{ic}} = 1$.

Training, Validation, and Optimization Strategy To ensure robust generalization, three types of dataset are generated and divided into training, validation, and test sets, each associated with distinct current profiles over the same time interval. The training current includes step changes, ramps, and sinusoidal functions of different frequencies to excite a wide range of system dynamics. The validation current profile is similar but not equal to the training one, ensuring that validation performance reflects generalization rather than memorization. The test dataset is fully independent and is used exclusively for final evaluation.

An early stopping mechanism is adopted to prevent overfitting and improve generalization. At each training epoch, the physics-informed loss defined in (17) is evaluated on the validation dataset. Training is terminated if the validation loss does not improve by at least $\Delta = 10^{-7}$ over a patience window of 2000 consecutive epochs. The trainable neural-network parameters θ corresponding to the minimum validation loss are stored and restored after training, ensuring that the final model represents the best generalization performance observed during training.

A two-stage optimization strategy is employed. In the first stage, the Adam optimizer is used for 6000 epochs with a learning rate of 10^{-3} to efficiently explore the parameter space. In the second stage, the L-BFGS optimizer (Andrew and Gao, 2007) is applied for further refinement with 2000 epochs and learning rate of 1, from epoch 6000 to 8000 epochs, leveraging approximate second-order information to achieve high-precision convergence. This hybrid optimization strategy combines the robustness of adaptive first-order methods with the accuracy of quasi-Newton optimization, which is particularly effective for physics-informed neural networks.

Table 2

Neural network architecture used in the PINN model

Parameter	Value
Input dimension	2
Number of hidden layers	3
Neurons per hidden layer	128
Activation function	$\tanh(\cdot)$
Output dimension	3

This training/validation/test construction, optimization schedule, and early-stopping criterion are shared by the CPINN model described in Section 3.3; the CPINN-specific differences are limited to how the conditioning parameters are sampled, described in Section 3.4.2 and 3.4.3.

In order to justify the selection of the parameters of the NN architecture given in Table 2, a sensitivity analysis map was performed, and the results are shown in Figure 3. As expected, the validation loss decreases when the number of layers/neurons per layer increases. However, increasing the complexity of the NN leads to a regime of diminishing returns. Furthermore, increasing the total number of neurons means higher training times and also higher complexity of the trained system which is to be deployed. The selected 3 layers/128 neurons per layer configuration is a compromise between validation loss (i.e. generalization capacity), training time and complexity of the trained system.

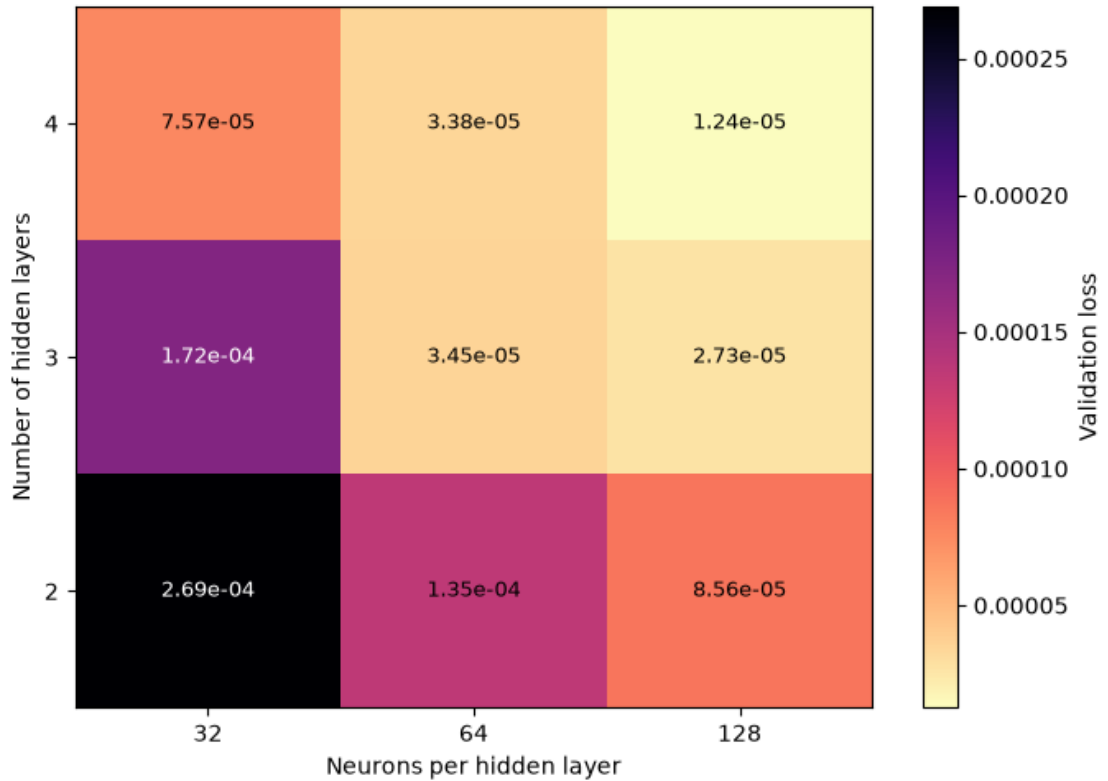


Figure 3: Sensitivity of PINN validation loss to network architecture. Heatmap of validation loss as a function of the number of hidden layers (2–4) and the number of neurons per hidden layer (32, 64, 128). Lower values (lighter cells) indicate better performance; the configuration with 4 hidden layers and 128 neurons per layer achieves the lowest validation loss (1.24×10^{-5}).

3.3. Conditional Physics-Informed Neural Network

The conditional physics-informed neural networks (CPINN) are PINN trained by minimizing a composite objective function that enforces consistency with the governing physical laws, some input conditions, and, when available, observational data. When CPINNs are intended to be used in control, the system control inputs and the system initial conditions are treated as CPINN inputs (Antonelo et al., 2024). The variable $y_\theta(t)$ denotes the neural network approximation of the system time evolution, obtained with inputs (t, u, y_0) , where $u \in \mathbb{R}$ represents the (constant) control input and $y_0 = y(0) \in \mathbb{R}^p$ is the initial state of the system ($p = 2$ for the electrolyzer model, with $y_0 = (V_{10}, V_{20})$).

In contrast to a conventional PINN, which learns the solution for a single input/initial-condition configuration, the CPINN is trained to learn a family of solutions over a prescribed parameter domain. However, unlike the PINN case with varying control inputs, a constant control input is used in CPINN. This is suitable for the intended use of the CPINN as a discretized prediction model integrated into optimal controllers as Model Predictive Controllers. By conditioning the network on the operating system inputs and initial conditions, a single trained model can predict trajectories for many operating points without retraining. For a control-oriented CPINN model, the inputs of the CPINN are the control input of the system, u , the initial condition of the system states, $y(0)$, and time, t , while $y_\theta(t)$ is the output of the neural network (see Figure 4).

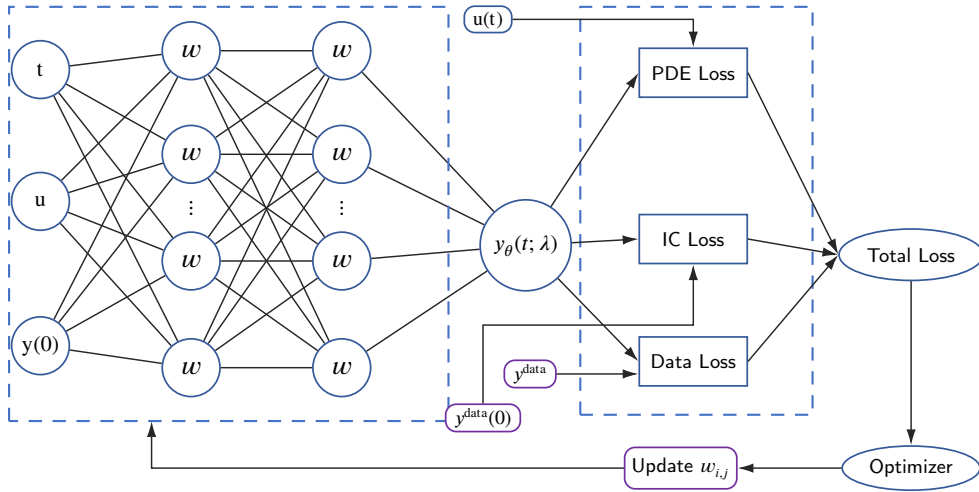


Figure 4: The CPINN network has the initial state $y(0)$ of the dynamic system and u as inputs, as well as time t . $y_\theta(t; \lambda)$ is the neural network prediction of the system at time $t \in [0, T]$ for the conditioning vector $\lambda = (u, y(0))$.

The expressions of the different loss functions of the CPINN as implemented in this work are detailed in the following paragraphs.

Physics loss In the conditional physics-informed neural network (CPINN) framework, the governing differential equations are enforced by minimizing the residual of the physical operator over both the temporal domain and a set of Sobol-sampled conditioning parameters.

Let $\lambda_i \in \Lambda$, $i = 1, \dots, N_p$, denote the set of conditioning variables sampled using a quasi-random Sobol sequence, and let $\{t_j\}_{j=1}^{N_c}$ represent the collocation points in the temporal domain. The neural network approximation is denoted by $y_\theta(t; \lambda)$.

The physics loss is defined as:

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_p N_c} \sum_{i=1}^{N_p} \sum_{j=1}^{N_c} \left\| \frac{dy_\theta}{dt}(t_j; \lambda_i) + \mathcal{N}[y_\theta(t_j; \lambda_i)] \right\|_2^2 \quad (18)$$

where $\mathcal{N}[\cdot]$ denotes the nonlinear differential operator associated with the governing equations of the system.

Initial condition loss In addition to enforcing the governing differential equations, the CPINN framework imposes consistency with prescribed initial conditions for each sampled conditioning parameter.

Recall that $\{\lambda_i\}_{i=1}^{N_p} \subset \Lambda$ denote the set of Sobol-sampled conditioning variables, and let the initial state corresponding to λ_i be given by $y_0(\lambda_i)$.

The initial condition loss is defined as

$$\mathcal{L}_{ic} = \frac{1}{N_p} \sum_{i=1}^{N_p} \|y_\theta(0; \lambda_i) - y_0(\lambda_i)\|_2^2, \quad (19)$$

where $y_\theta(0; \lambda_i)$ represents the network prediction evaluated at time $t = 0$ for the conditioning parameter λ_i , and $y_0(\lambda_i)$ denotes the corresponding prescribed initial state.

Data loss When observational measurements of the system output are available, a data fidelity term is introduced to penalize the discrepancy between the network prediction and the observed data.

For the same conditioning variables λ_i , $i = 1, \dots, N_p$, let $\{t_k\}_{k=1}^{N_d}$ represent the measurement time instants, and let the observed data be denoted by $y^{\text{data}}(t_k; \lambda_i)$. Then, the data loss is defined as

$$\mathcal{L}_{\text{data}} = \frac{1}{N_p N_d} \sum_{i=1}^{N_p} \sum_{k=1}^{N_d} \|y_\theta(t_k; \lambda_i) - y^{\text{data}}(t_k; \lambda_i)\|_2^2. \quad (20)$$

Similar to the PINN case, the total loss function in CPINN is defined as in (10) and the network parameters θ are found with the optimization in (11).

Sequential Long-Horizon Prediction The CPINN is trained over a finite temporal domain $t \in [0, T]$ under the condition of a constant control input variable. To enable prediction over a longer horizon $[0, t_{\text{final}}]$ without extrapolating beyond the training time domain, a sequential simulation strategy is employed.

The global time horizon t_{final} is partitioned into K sub-intervals of length T_{sim} , $T_{\text{sim}} \leq T$. Each time sub-interval is:

$$[t_k, t_{k+1}] = [kT_{\text{sim}}, (k+1)T_{\text{sim}}], \quad k = 0, \dots, K-1, \quad (21)$$

Within each sub-interval, inputs and conditioning parameters are assumed to be constant and are approximated by representative values.

For each sub-interval, the CPINN predicts a local trajectory

$$y_{\theta^*}^{(k)}(\tau; u_k, y_0^{(k)}), \quad \tau \in [0, T_{\text{sim}}], \quad (22)$$

where $y_0^{(k)}$ denotes the state at the beginning of the interval. State continuity is enforced by updating the initial condition for the next interval as

$$y_0^{(k+1)} = y_{\theta^*}^{(k)}(T_{\text{sim}}) \quad (23)$$

The global trajectory is obtained by concatenating the local predictions and mapping the local time coordinate to the global time variable via $t = t_k + \tau$. This approach enables stable long-horizon prediction while ensuring that the neural network is only evaluated within its trained temporal domain.

3.4. CPINN model of a PEM electrolyzer equivalent circuit

In this section, a Conditional Physics-Informed Neural Network (CPINN) is used to model the PEM electrolyzer ECM presented in section 2. In the context of control-oriented models, as explained in section 3.3, the CPINN will be conditioned on the control input of the system and its initial conditions, so that it can predict trajectories for many operating conditions without retraining.

The CPINN predicted output, is $y_\theta(t_i) = (V_{\theta,1}(t_i), V_{\theta,2}(t_i), V_\theta(t_i))$, as for the PINN model in Section 3.2. The goal of the CPINN is to approximate a parametric mapping from time and conditioning variables to the circuit and total voltages.

For the electrolyzer model, we define the input-conditioning vector

$$\lambda := (I, V_{10}, V_{20}), \quad (24)$$

where V_{10} and V_{20} are the state voltages initial conditions

$$V_{10} = V_1(0), \quad V_{20} = V_2(0), \quad (25)$$

defined within the admissible domain

$$\Omega := \left\{ (t, I, V_{10}, V_{20}) \mid t \in [0, T], I \in [I_{\min}, I_{\max}], \right. \\ \left. V_{10} \in [V_{10}^{\min}, V_{10}^{\max}], V_{20} \in [V_{20}^{\min}, V_{20}^{\max}] \right\}. \quad (26)$$

The trained surrogate can then be queried for values t , I , V_{10} and V_{20} in this admissible domain.

A natural choice for T is $T = \max(R_1 C_1, R_2 C_2)$, which covers the slowest dynamics. For the initial-condition, physically consistent bounds are

$$V_{10}^{\max} = R_1 I_{\max}, \quad V_{20}^{\max} = R_2 I_{\max}, \quad (27)$$

with $V_{10}^{\min} = V_{20}^{\min} = 0$ (or other bounds if required by data/operation).

The CPINN is a fully-connected network with parameters θ that implements the mapping

$$(V_{\theta,1}(t), V_{\theta,2}(t), V_{\theta}(t)) = y_{\theta}(t, I, V_{10}, V_{20}). \quad (28)$$

According to (18), the physics loss is

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left(RES_1(t_i; \lambda_i)^2 + RES_2(t_i; \lambda_i)^2 + RES_3(t_i; \lambda_i)^2 \right), \quad (29)$$

where $N_r = N_p N_c$ is the total number of collocation points $\{(t_i, \lambda_i)\}_{i=1}^{N_r} = \{(t_i, I_i, V_{10,i}, V_{20,i})\}_{i=1}^{N_r}$, obtained by combining N_p Sobol-sampled conditioning tuples $\lambda = (I, V_{10}, V_{20})$ with N_c time points each, as detailed in Section 3.4.2.

To enforce (25) across the conditioning domain, we sample N_p conditioning tuples $\{(I_j, V_{10,j}, V_{20,j})\}_{j=1}^{N_p}$ and compute the initial condition loss as:

$$\mathcal{L}_{\text{ic}} = \frac{1}{N_p} \sum_{j=1}^{N_p} \left[(V_{\theta,1}(0, I_j, V_{10,j}, V_{20,j}) - V_{10,j})^2 \right. \\ \left. + (V_{\theta,2}(0, I_j, V_{10,j}, V_{20,j}) - V_{20,j})^2 \right]. \quad (30)$$

As in the PINN case, the total loss is $\mathcal{L}_{\text{total}} = \lambda_{\text{phys}} \mathcal{L}_{\text{phys}} + \lambda_{\text{ic}} \mathcal{L}_{\text{ic}}$, with $\lambda_{\text{phys}} = \lambda_{\text{ic}} = 1$ the weighting factors controlling the relative contributions of the physics residual and initial-condition losses, respectively.

3.4.1. Input normalization

To improve numerical conditioning, each input is scaled to $[-1, 1]$. For any scalar $x \in [x_{\min}, x_{\max}]$, the normalized value is

$$x^n = 2 \frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1. \quad (31)$$

We apply (31) to t , I , V_{10} , and V_{20} using the bounds in (26)–(27). In this work, the current range is $I_{\min} = 0$ A and $I_{\max} = 190$ A, which, according to (27) and the parameters in Table 1, gives $V_{10}^{\max} = R_1 I_{\max} = 0.28272$ V and $V_{20}^{\max} = R_2 I_{\max} = 0.25061$ V, with $V_{10}^{\min} = V_{20}^{\min} = 0$ V. The training time horizon is $T = \max(R_1 C_1, R_2 C_2) = R_1 C_1 \approx 0.606$ s. This ensures comparable magnitudes across inputs and improves gradient flow during optimization.

3.4.2. Sampling strategy: Sobol conditioning and time collocation

Uniform coverage of the conditioning domain is critical for generalization. We sample the 3D conditioning space (I, V_{10}, V_{20}) using Sobol low-discrepancy sequences. Let $\{\xi_j\}_{j=1}^{N_p} \subset [0, 1]^3$ be Sobol points with $\xi_j = (\xi_{j,1}, \xi_{j,2}, \xi_{j,3})$. These are mapped to physical ranges via

$$I_j = I_{\min} + \xi_{j,1} (I_{\max} - I_{\min}), \quad (32)$$

$$V_{10,j} = V_{10}^{\min} + \xi_{j,2} (V_{10}^{\max} - V_{10}^{\min}), \quad (33)$$

$$V_{20,j} = V_{20}^{\min} + \xi_{j,3} (V_{20}^{\max} - V_{20}^{\min}). \quad (34)$$

Time collocation sampling For each conditioning tuple, we sample N_c time points $t \sim \mathcal{U}(0, T)$ (or Sobol in 1D). The total number of collocation points per batch is

$$N_r = N_p N_c. \quad (35)$$

The physics residuals (12), (13) and (14) are evaluated at these points.

3.4.3. Optimization, validation, and early stopping

The CPINN uses the same two-stage Adam/L-BFGS optimization schedule and early-stopping criterion described in Section 3.2 (patience $P = 2000$ epochs, minimum improvement $\Delta = 10^{-7}$; settings summarized in Table 7). The only methodological difference is that, for CPINN, the validation loss in (36) is evaluated on an independent Sobol-sampled conditioning set $\{\lambda_j^{\text{val}}\}$ and time collocation points, rather than on a single fixed current trajectory as in the PINN case. This ensures that early stopping reflects generalization across the full conditioning domain (I, V_{10}, V_{20}) and not only over time for a single operating condition. The independence of the training and validation Sobol sequences used for this purpose is detailed in Section 3.4.5.

$$\mathcal{L}_{\text{val}} = \lambda_{\text{phys}} \mathcal{L}_{\text{phys, val}} + \lambda_{\text{ic}} \mathcal{L}_{\text{ic, val}}. \quad (36)$$

is evaluated on an independent Sobol-sampled conditioning set $\{\lambda_j^{\text{val}}\}$ and time collocation points, rather than on a single fixed current trajectory as in the PINN case. This ensures that early stopping reflects generalization across the full conditioning domain (I, V_{10}, V_{20}) and not just over time for one operating condition. The independence of the training and validation sobol sequences used for this purpose is detailed in section 3.4.5.

As in the PINN case, moderate changes to P and Δ mainly affect the stopping epoch and selected checkpoint rather than the relative performance trend between the two models; CPINN consistently achieves lower validation loss and more stable long-horizon behavior regardless of the specific early-stopping hyperparameters used.

To select the NN architecture, a sensitivity analysis map is performed and shown in Figure 5. As in the PINN case, and for the same reasons, the selected architecture consists of 3 layers and 128 neurons per layer, which also turns out to make the comparison between PINN and CPINN more meaningful. The whole set of parameters is described in Table 3.

Table 3

Neural network architecture used in the CPINN model

Parameter	Value
Input dimension	4
Number of hidden layers	3
Neurons per hidden layer	128
Activation function	$\tanh(\cdot)$
Output dimension	3

3.4.4. Definition of a batch in CPINN training

In standard supervised learning, a *batch* typically refers to a subset of labeled input–output pairs drawn from a finite dataset. In the proposed CPINN setting, there is no fixed dataset of labeled trajectories; instead, each batch is

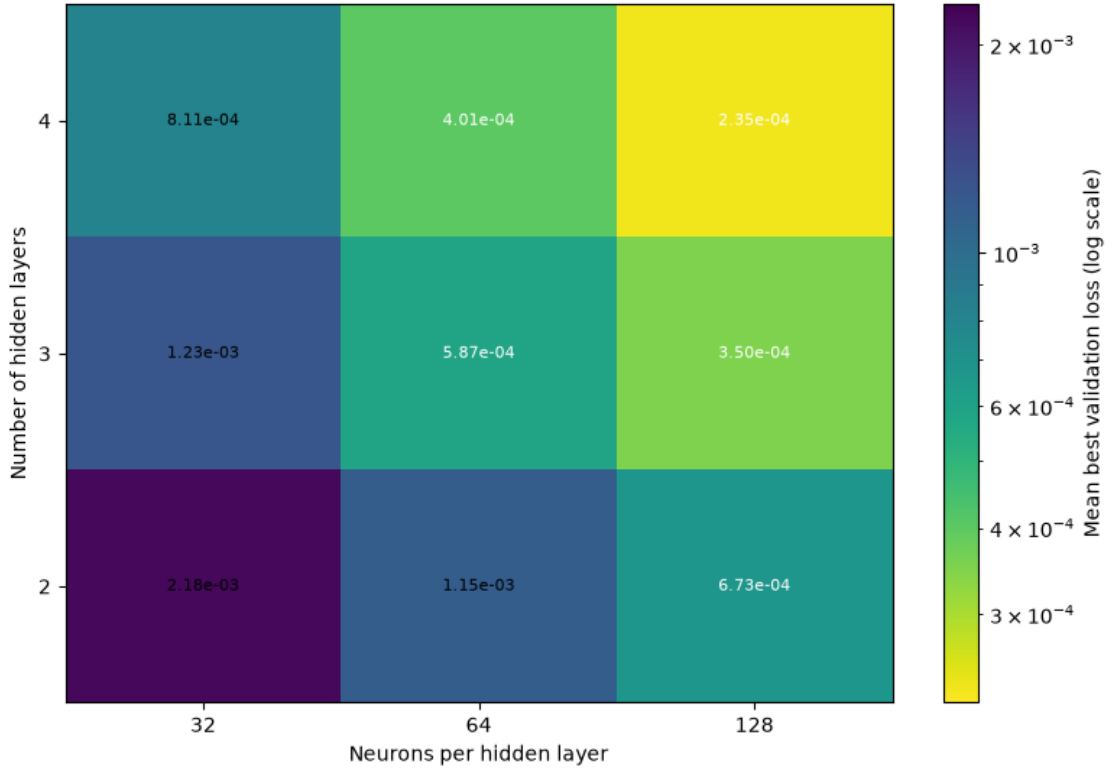


Figure 5: Sensitivity of CPINN validation loss to network architecture. Heatmap of the mean best validation loss (log scale) as a function of the number of hidden layers (2–4) and the number of neurons per hidden layer (32, 64, 128). Lower values (yellow) indicate better performance; the configuration with 4 hidden layers and 128 neurons per layer achieves the lowest mean best validation loss (2.35×10^{-4}).

synthetically constructed by sampling from the continuous spatio-parameter domain. Concretely, a batch is formed in two steps:

1. **Conditioning sampling:** draw N_p tuples $\lambda_j = (I_j, V_{10,j}, V_{20,j})$ using a Sobol low-discrepancy sequence over the parameter domain.
2. **Time collocation sampling:** for each conditioning tuple λ_j , draw N_c collocation time points $\{t_i\}_{i=1}^{N_c} \subset [0, T]$ (e.g., uniform or Sobol in 1D).

The Cartesian product of these two sets yields the batch collocation set

$$\mathcal{B} = \{(t_i, \lambda_j)\}_{i=1, \dots, N_c}^{j=1, \dots, N_p}, \quad |\mathcal{B}| = N_r = N_p N_c. \quad (37)$$

At each point in $\mathcal{B}$, the network outputs are evaluated, automatic differentiation is used to compute $dV_{\theta,1}/dt$ and $dV_{\theta,2}/dt$ and the physics residuals are computed to form $\mathcal{L}_{\text{phys}}$ in (29). In addition, the initial-condition loss $\mathcal{L}_{\text{ic}}$ is computed using the same N_p conditioning tuples evaluated at $t = 0$. Therefore, in CPINN training, the batch size is not the number of labeled samples but the number of collocation points generated per optimization step, i.e., $N_r = N_p N_c$. Increasing N_p improves coverage of the conditioning space (I, V_{10}, V_{20}) , while increasing N_c improves enforcement of the ODE constraints along the time axis. Both affect training cost because computational and memory complexity scale approximately with N_r .

3.4.5. Independence between training and validation sampling

To assess generalization and avoid overly optimistic validation metrics, the training and validation collocation sets must be statistically independent. In our implementation, this is enforced by using two independent Sobol samplers:

one for training and one for validation. Specifically, we generate

$$\{\lambda_j^{\text{train}}\}_{j=1}^{N_p} \sim \text{Sobol}([0, 1]^3), \quad \{\lambda_j^{\text{val}}\}_{j=1}^{N_{p,\text{val}}} \sim \text{Sobol}([0, 1]^3), \quad (38)$$

where the two sequences are initialized independently (e.g., different Sobol engines, different seeds/scrambling states, or disjoint subsequences). Each set is then mapped to the physical ranges of (I, V_{10}, V_{20}) using the same affine transformation described in Section 3.4.2.

This separation prevents sampling leakage, *i.e.* the scenario in which validation points coincide with (or are highly correlated with) training points. Leakage would artificially reduce $\mathcal{L}_{\text{val}}$ and weaken the reliability of early stopping. By constructing the validation set using an independent sobol sequence (and independently sampled time collocation points), $\mathcal{L}_{\text{val}}$ provides a more faithful estimate of the CPINN's ability to interpolate across unseen conditioning tuples and time locations within the admissible domain.

3.4.6. Prediction for long horizons via sequential simulation

The CPINN is trained on a finite horizon $t \in [0, T]$. To predict system behavior over a longer horizon $[0, t_{\text{final}}]$ under a time-varying current $I(t)$, we employ a sequential (piecewise) simulation strategy. The key idea is to exploit conditioning on initial states: each local prediction starts from the final state of the previous interval.

Piecewise constant discretizations of current: We discretize $I(t)$ into piecewise constant segments of length T_{sim} :

$$I_k := I\left(\frac{t_k + t_{k+1}}{2}\right), \quad [t_k, t_{k+1}] = [kT_{\text{sim}}, (k+1)T_{\text{sim}}], \quad (39)$$

for $k = 0, \dots, K-1$, where $K = \lceil t_{\text{final}}/T_{\text{sim}} \rceil$. Typically $T_{\text{sim}} < T$ to remain within the training window. Within each sub-interval, the input current is approximated as constant. In this work, $T_{\text{sim}} = \min(R_1 C_1, R_2 C_2) = R_2 C_2 \approx 0.0606$ s. This choice satisfies $T_{\text{sim}} < T$, so that the network is always evaluated within its training window, and is small enough for the piecewise constant approximation to capture the variations of $I(t)$, since it matches the fastest time constant of the circuit.

Sequential chaining using state continuity: Let $(V_{10}^{(k)}, V_{20}^{(k)})$ denote the initial states for interval k . For each interval, the CPINN predicts local trajectories

$$(V_{\theta,1}^{(k)}(\tau), V_{\theta,2}^{(k)}(\tau), V_{\theta}^{(k)}(\tau)) = y_{\theta^*}(\tau, I_k, V_{10}^{(k)}, V_{20}^{(k)}), \quad \tau \in [0, T_{\text{sim}}], \quad (40)$$

where θ^* denotes the trained parameters and $\tau = t - t_k$ denotes the local time coordinate within the k -th sub-interval. This time-shifting ensures that the network is always evaluated within its trained temporal domain $[0, T]$, thereby avoiding extrapolation beyond the training range. The terminal values are used to update the initial conditions for the next interval:

$$V_{10}^{(k+1)} = V_{\theta,1}^{(k)}(T_{\text{sim}}), \quad V_{20}^{(k+1)} = V_{\theta,2}^{(k)}(T_{\text{sim}}). \quad (41)$$

This ensures continuity of capacitor voltages across interval boundaries, which is physically required because capacitor voltages cannot change instantaneously. The same trained network y_{θ^*} is reused in each sub-interval without retraining and no additional training is performed during sequential simulation.

Global trajectory assembly: Each interval produces samples on a local grid $\tau \in [0, T_{\text{sim}}]$. The global time is recovered by $t = \tau + t_k$. Concatenating interval outputs yield $V_{\theta}(t)$, $V_{\theta,1}(t)$, and $V_{\theta,2}(t)$ over $[0, t_{\text{final}}]$.

4. Comparison of PINN and CPINN prediction performance

In this section, the discussed methods are compared to see how well do the models perform when the operating conditions change and also which model remains reliable when predicting farther into the future. In both cases the data loss term is not considered because the goal of the study is to compare the pure PINN aspects of both formulations.

4.1. Same range of time and current

The input current used in training for the PINN case is shown in Figure 6. This profile was selected because it was shown to be rich enough to train the system with sufficient generalization capacity. Similar signals work as well. Table 4 contains all the information about the datasets used in the training and tests. In the training phase, time and current vectors of 12000 points are used. The current profile ranges from 0 A to 190 A and the time vector ranges from 0 s to 90 s. Current and time vectors of 3000 points in the same ranges but with different current profiles are used as validation dataset to find the best model without overfitting. Finally, 2000 points are chosen in the same time range in order to test and analyze the generalization capacity of the trained model.

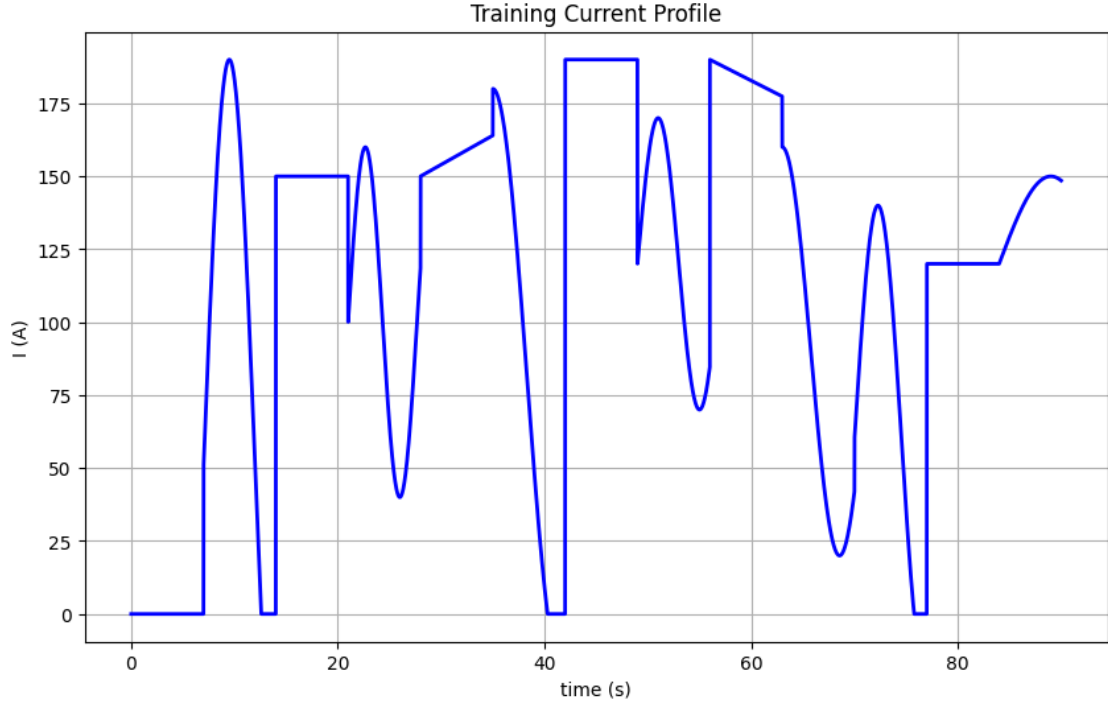


Figure 6: Current profile used in training for the PINN case

The neural network architecture of PINN, is shown in Table 5, with 3 hidden layers with 128 neurons each. The training is carried out for a total of 8000 epochs. The input dimension of the neural network is 2, which includes time (t) and current $I(t)$, and the output dimension is 3, corresponding to the total voltage V and branch voltages V_1 and V_2 . The activation function of the neural network is the hyperbolic tangent, which is a smooth activation suitable for automatic differentiation. In the PINN training two approaches implemented as explained in the table 7, maximum epochs $N_{\text{epochs}} = 8000$ that in the stage 1 Adam optimization with learning rate 10^{-3} applied until epoch $S_{\text{epoch}} = 6000$ and then optimization switches to LBFGS optimization with learning rate of 1.0. Training stopped at 5912 epoch with 2.55×10^{-6} validation loss that is the best validation loss and training terminated at epoch 7912 after checking for the lower validation loss than the best validation loss to continue the training. The evolution of the PINN training and validation losses, together with the individual loss components, is shown in 7. With the same optimization approaches shown in Table 7, and architecture shown in the Table 6, implemented in CPINN, the train stopped at epoch 6343 with best validation loss 5.22×10^{-7} and the training ended at epoch 7950. The corresponding CPINN loss curves are shown in figure 8.

After training CPINN and PINN, the objective is to study the generalization of these models by test datasets. In Figure 9, as can be seen by comparing the performance of each model, CPINN, shows better performance in the case of doing prediction and generalization compared to PINN. To compare the prediction accuracy of CPINN and PINN in the context of voltages of each branch, Figure 10, CPINN outperformed PINN and shows a better performance. In addition to figures that are comparing the performance of the models, Table 11, quantitatively compares the models by

evaluation metrics which represents that CPINN with $MSE:6.42 \times 10^{-5}$, $RMSE:8.01 \times 10^{-3}$ and $MAE:4.26 \times 10^{-4}$, $MBE:3.46 \times 10^{-3}$ and $R^2:0.998$ outperforms the PINN. The used current profile shown in Figure 9 (c) has been selected because it displays a variety of steps, ramps and sinusoidal components that are typical for the intended application. Similar results are obtained if other sensible profiles are used.

Table 4

Dataset splits and corresponding current excitation profiles

Dataset	Points	Time range (s)	$I(t)$ profile	I range (A)
Training	12000	0–90	$I_{\text{train}}(t)$	0–190
Validation	3000	0–90	$I_{\text{val}}(t)$	0–160
Test	2000	0–90	$I_{\text{test}}(t)$	Piecewise

Table 5

PINN architecture and input normalization settings. FCNN stands for Fully Connected Neural Network

Component	Setting	Notes
Input dimension	2	Time t and input current $I(t)$
Output dimension	3	Total voltage V , branch voltages V_1 , V_2
Hidden layers	3	Fully connected layers
Layer sizes	[2, 128, 128, 128, 3]	Three hidden layers with 128 neurons each
Activation function	tanh	Smooth activation suitable for automatic differentiation
Time normalization	$t \in [-1, 1]$	Using training time range $[t_{\min} = 0, t_{\max} = 90]$
Current normalization	$I \in [-1, 1]$	Using $I_{\min} = 0$, $I_{\max} = 190$ A

Table 6

CPINN architecture and input normalization settings

Component	Setting	Notes
Model type	FCNN	
Input dimension	4	Time t and input current $I(t)$, branch voltages $V_1(t=0)$ and $V_2(t=0)$
Output dimension	3	Total voltage V , branch voltages V_1 , V_2
Hidden layers	3	Fully connected layers
Layer sizes	[4, 128, 128, 128, 3]	Three hidden layers with 128 neurons each
Activation function	tanh	Smooth activation suitable for automatic differentiation
Time normalization	$t \in [-1, 1]$	Using training time range $[0, T]$, $T = R_1 C_1 \approx 0.606$ s
Current normalization	$I \in [-1, 1]$	Using $I_{\min} = 0$, $I_{\max} = 190$ A
Voltages branch (V1)	$V_1 \in [-1, 1]$	Using $V_{1\min} = 0$, $V_{1\max} = 0.28272$
Voltages branch (V2)	$V_2 \in [-1, 1]$	Using $V_{2\min} = 0$, $V_{2\max} = 0.25061$

Table 7

Optimization strategy and early stopping configuration for PINN and CPINN

Component	Setting	Notes
Maximum epochs	$N_{\text{epochs}} = 8000$	Upper bound on training iterations
Optimizer (stage 1)	Adam	Learning rate 10^{-3}
Optimizer (stage 2)	L-BFGS	Learning rate 1.0, max_iter = 20, history_size = 50
Switch epoch	$S_{\text{epoch}} = 6000$	Transition from Adam to L-BFGS
Batch size	Full batch	All training time points used per iteration
Early stopping criterion	Validation loss	No gradient updates on validation set
Patience	2000 epochs	Training stops after 2000 epochs without improvement
Minimum improvement	10^{-7}	Minimum validation loss decrease to reset patience
Physics loss weight	$\lambda_{\text{physics}} = 1.0$	Weight for physics residual loss
Initial Condition loss weight	$\lambda_{\text{IC}} = 1.0$	Weight for initial condition loss

Table 8

Summary of the PINN validation checkpoint, early stopping criterion, and training time.

Component	Setting	Notes
Best validation epoch	≈ 5912	Lowest validation loss observed
Best validation loss	$\approx 2.55 \times 10^{-6}$	Selected model checkpoint
Stopped epoch	≈ 7912	Training terminated by early stopping
Training time	38s	computation time of PINN

Table 9

Summary of the CPINN validation checkpoint, early stopping criterion, and training time.

Component	Setting	Notes
Best validation epoch	≈ 6343	Lowest validation loss observed
Best validation loss	$\approx 5.22 \times 10^{-7}$	Selected model checkpoint
Stopped epoch	≈ 7950	Training terminated by early stopping
Training time	57s	computation time of CPINN

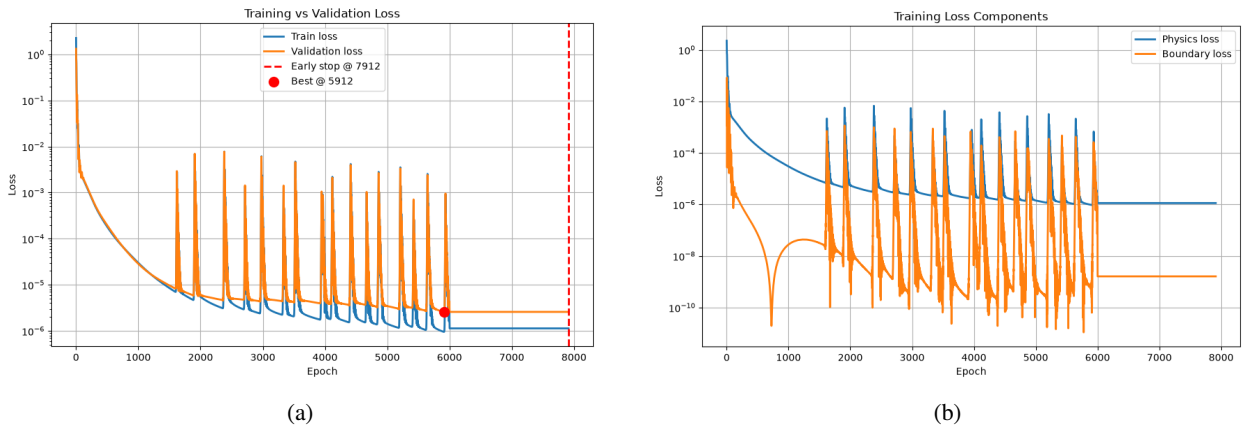


Figure 7: (a) Evolution of the PINN training and validation losses during optimization on a logarithmic scale. Both losses decrease consistently, showing stable convergence of the PINN. The red marker denotes the epoch with the best validation loss, while the dashed line indicates the early-stopping point selected to prevent overfitting and improve generalization. (b) Decomposition of the PINN training loss into physics loss and boundary (initial-condition) loss components. The decrease of both terms demonstrates that the PINN simultaneously minimizes the governing-equation residual while satisfying the imposed constraints, confirming stable physics-informed learning dynamics.

4.2. Testing time outside the training range

To evaluate the temporal generalization capability of the proposed models, both the standard PINN and the CPINN were trained using current within the time interval $t \in [0, 90]$ s. After training, the models were tested on an extended time horizon dataset spanning $t \in [0, 200]$ s, which lies partially outside the training domain. This experiment aims to assess the ability of each model to maintain accurate predictions when simulating longer operating periods than those seen during training.

Temporal extrapolation is a challenging task for neural network models because prediction errors can accumulate over time, leading to divergence from the physical trajectory. In dynamical systems such as equivalent circuit of PEM electrolyzer, accurate long-term prediction requires not only fitting the training data but also learning the underlying physical dynamics that governs the system.

According to the figure, 11, while both of them show better performance during the training time but beyond the training time, after 90s, PINN starts to extrapolate and drift from the reference data (analytical solution), while CPINN stays accurate from the 90 s until the 200 s. To a quantitative comparison in the case of longer time prediction, Table 10 demonstrates that the CPINN achieves lower error metrics (MSE, RMSE, and MAE, MBE and R^2) compared to the standard PINN under extended-time testing conditions. Although both models produce accurate predictions

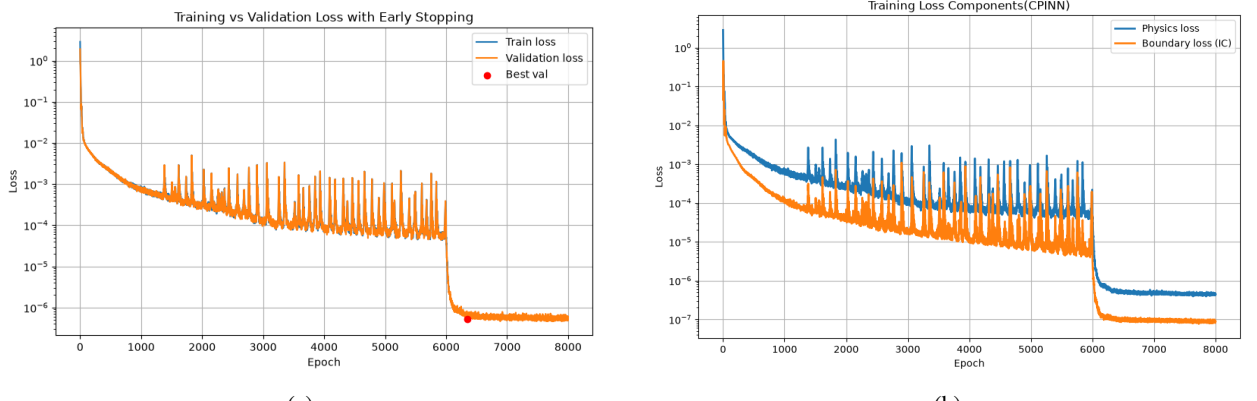


Figure 8: (a) Evolution of the CPINN training and validation losses throughout the optimization process, presented on a logarithmic scale. Both curves exhibit a steady reduction, indicating stable and reliable convergence of the conditional physics-informed model. The red marker highlights the epoch corresponding to the minimum validation loss, while the dashed line represents the early-stopping criterion applied to avoid overfitting and enhance generalization across different conditioning parameters. (b) Breakdown of the CPINN training loss into its main components, the physics-based (ODE) loss and the initial-condition loss. The consistent decrease of these terms shows that the CPINN effectively enforces the governing equations while respecting the imposed constraints, demonstrating robust and stable conditional physics-informed learning behavior.

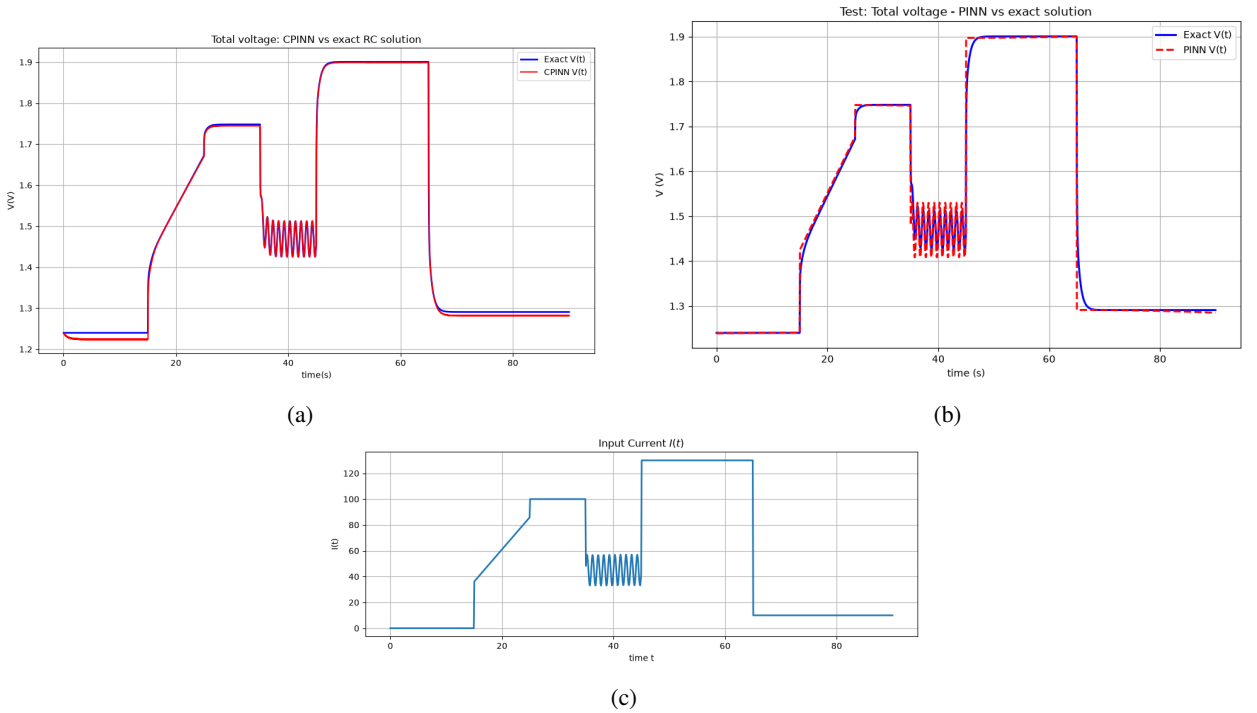


Figure 9: (a) Voltage prediction obtained using the CPINN model, showing strong agreement with the exact RC solution across dynamic transients and oscillatory regions. (b) Voltage prediction obtained using the PINN model under the same test conditions. While the PINN captures the general voltage trend, larger deviations appear during fast transient and oscillatory intervals compared to CPINN. These results highlight the improved prediction accuracy and dynamic response modeling capability of the CPINN framework. (c) Test current profile.

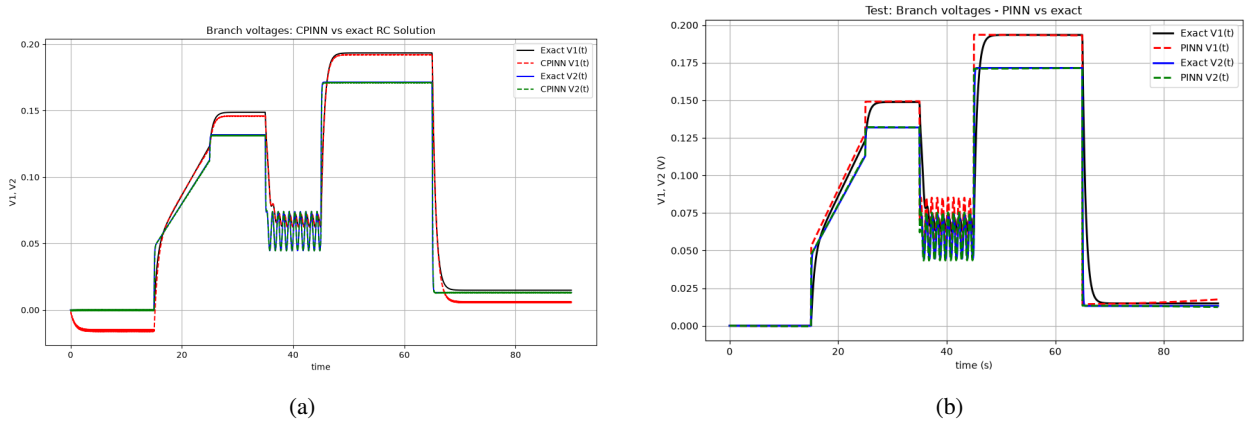


Figure 10: (a) CPINN prediction of the RC-branch voltages $V_1(t)$ and $V_2(t)$ compared with the exact equivalent-circuit solution. The CPINN accurately reproduces both transient behavior and oscillatory dynamics of the branches. (b) PINN prediction of the same branch voltages under identical test conditions. While the overall trends are captured, larger deviations appear during fast transients and oscillatory intervals compared to CPINN.

within the training interval, the PINN exhibits an increased deviation beyond 90 s, indicating a reduced stability when extrapolating in time. In contrast, the CPINN maintains consistent voltage predictions and smoother branch dynamics over the entire 0–200 s interval. The corresponding branch voltages V_1 and V_2 over the extended interval are compared in figure 12.

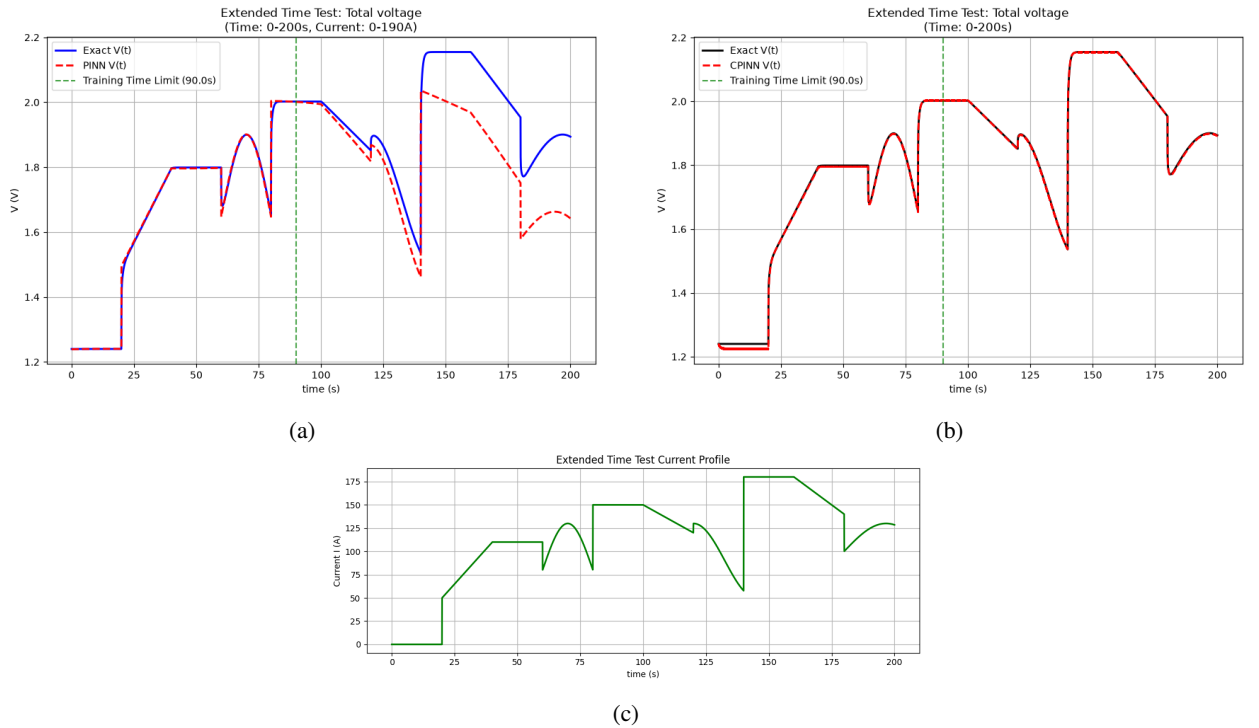


Figure 11: (a) PINN voltage prediction compared with the exact equivalent-circuit solution, where increasing deviation is observed outside the training time region. (b) CPINN voltage prediction under the same conditions, showing improved stability and closer agreement with the reference solution over the extended horizon. The vertical dashed line indicates the upper bound of the training time domain, highlighting the models' out-of-distribution temporal generalization capability. (c) Test current profile used in testing the trained models.

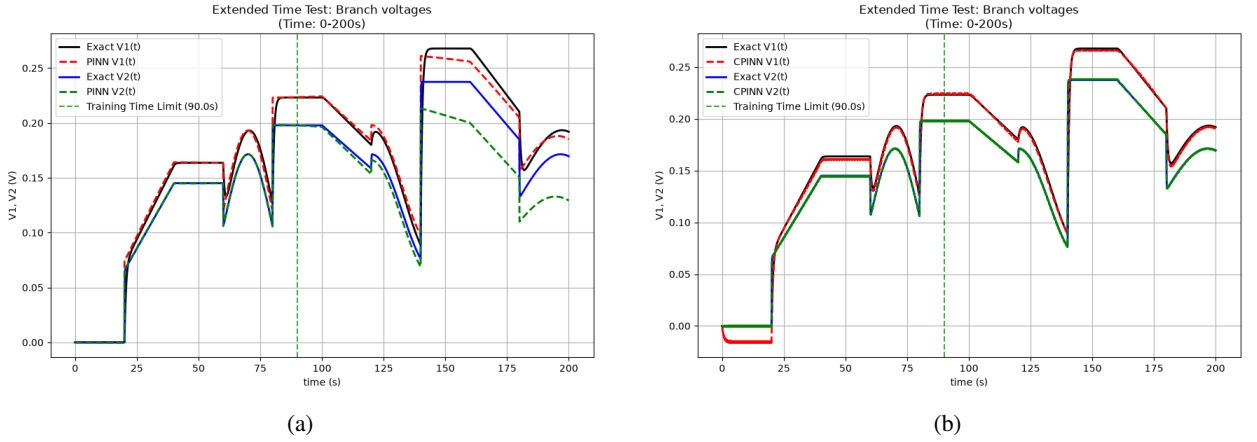


Figure 12: (a) PINN prediction of branch voltages evaluated over the extended interval 0–200s after training within 0–90s, where increasing deviations appear outside the training domain. (b) CPINN prediction under identical conditions, demonstrating improved stability and closer agreement with the reference equivalent-circuit solution during long-horizon extrapolation. The vertical dashed line indicates the upper bound of the training time region.

Table 10

Comparison of PINN and CPINN prediction performance on an extended time horizon test. Both models were trained on $t \in [0, 90]$ s and evaluated on $t \in [0, 200]$ s using the same current range (0–190 A).

		MSE (V ²)	RMSE (V)	MAE (V)	MBE (V)	R^2
PINN	V	1.10×10^{-2}	1.05×10^{-1}	6.48×10^{-2}	-6.24×10^{-2}	0.821
	V_1	9.49×10^{-5}	9.74×10^{-3}	4.73×10^{-3}	1.25×10^{-5}	0.982
	V_2	3.43×10^{-4}	1.85×10^{-2}	1.11×10^{-2}	-1.08×10^{-2}	0.918
CPINN	V	4.55×10^{-5}	6.74×10^{-3}	3.41×10^{-3}	-2.81×10^{-3}	0.999
	V_1	2.64×10^{-5}	5.14×10^{-3}	3.04×10^{-3}	-2.62×10^{-3}	0.995
	V_2	9.16×10^{-7}	9.57×10^{-4}	5.05×10^{-4}	-2.48×10^{-4}	1.000

The reason behind the degradation observed in the PINN model shown in figure (11a) beyond the training interval can be attributed to its trajectory-based learning formulation, which implicitly assumes a fixed initial condition and operating regime. By increasing the prediction time, small residual errors accumulate and propagate through the dynamical system, leading to drift in the prediction. In contrast, the CPINN learns a family of solution trajectories parameterized by the initial state and input current, which improves the stability of the model and mitigates error accumulation in long-time horizon.

Table 11

Quantitative comparison of PINN and CPINN prediction performance on the generalization test dataset shown in Figure 9.

		MSE (V ²)	RMSE (V)	MAE (V)	MBE (V)	R^2
PINN	V	3.09×10^{-4}	1.75×10^{-2}	6.24×10^{-3}	-8.50×10^{-4}	0.995
	V_1	2.20×10^{-4}	1.48×10^{-2}	4.95×10^{-3}	2.65×10^{-4}	0.960
	V_2	1.82×10^{-5}	4.27×10^{-3}	8.86×10^{-4}	-1.20×10^{-4}	0.996
CPINN	V	6.42×10^{-5}	8.01×10^{-3}	4.26×10^{-3}	3.46×10^{-3}	0.998
	V_1	1.28×10^{-5}	3.58×10^{-3}	2.83×10^{-3}	4.15×10^{-4}	0.989
	V_2	5.02×10^{-6}	2.24×10^{-3}	1.38×10^{-3}	-7.24×10^{-4}	0.999

4.3. Robustness Under Parameter Mismatch: Effect of Data Loss

The results presented in Sections 4.1 and 4.2 were obtained under the assumption that the parameters used in the physics residual were exactly equal to those of the reference equivalent circuit model. Although this setting is appropriate for controlled benchmarking, it is idealized. In practice, equivalent circuit model (ECM) parameters are

typically identified from measurements and may therefore be affected by uncertainty, estimation errors, or operating-point dependence. To evaluate the robustness of the proposed framework under more realistic conditions, an additional experiment was performed in a parameter-mismatch setting.

In this experiment, the reference trajectories were generated using the nominal ECM parameters reported in Table 1, whereas the physics residuals used during training were computed with perturbed parameter values. In particular, the residual model employed

$$R_1^{\text{res}} = 0.92 R_1, \quad C_1^{\text{res}} = 1.15 C_1, \quad R_2^{\text{res}} = 1.05 R_2, \quad C_2^{\text{res}} = 0.93 C_2, \quad (42)$$

and

$$E_{\text{rev}}^{\text{res}} = 0.99 E_{\text{rev}}, \quad R_{\text{ohm}}^{\text{res}} = 1.01 R_{\text{ohm}}. \quad (43)$$

This defines a controlled model-mismatch scenario in which the physical model embedded into the loss is no longer exactly consistent with the reference trajectories.

Under this setting, the total voltage reconstructed from the predicted states $V_{\theta,1}$ and $V_{\theta,2}$ was also computed using the perturbed values of E_{rev} and R_{ohm} , consistently with the assumption that these are the parameter values available to the model in practice. The hybrid training objective was defined as

$$\mathcal{L}_{\text{total}} = \lambda_{\text{phys}} \mathcal{L}_{\text{phys}} + \lambda_{\text{IC}} \mathcal{L}_{\text{IC}} + \lambda_{\text{data}} \mathcal{L}_{\text{data}}, \quad (44)$$

where the additional data-fidelity term provides supervised correction when the embedded physics is imperfect. This combination of physics and data based training can also contribute to regularize any noise present in the data, preventing the overfitting of the noisy data that could occur in a pure data-driven situation.

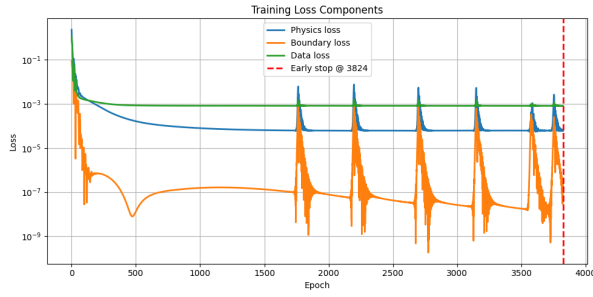
The conditioning tuples (I, V_{10}, V_{20}) were sampled from the admissible domain using a Sobol low-discrepancy sequence. A large pool of conditioning cases was generated in advance, and each batch was formed by randomly selecting a subset of cases from this pool. The same selected tuples were used both for neural-network evaluation and for generating the corresponding reference trajectories with the numerical solver, ensuring consistency between the conditioning inputs and the supervised targets. In addition, the time instants used for the data-loss term were sampled independently of the collocation points used in the physics loss. This reflects the practical situation in which measurement times are generally not tied to the residual-enforcement grid.

Figures 15–16 and Tables 12–13 summarize the obtained results. The loss curves of the hybrid CPINN, shown in Fig. 14, indicates stable optimization and successful convergence of the physics, initial-condition, and data-loss components. Early stopping selected the best CPINN checkpoint at approximately epoch 7560, with training terminated at epoch 7999. For the hybrid PINN(13), the best validation epoch was approximately 1824 and training terminated at epoch 3824. The training time difference between PINN and CPINN has increased compared with the case without data loss. However, this is not important from the point of view of implementation because the training is done off-line and not in real time.

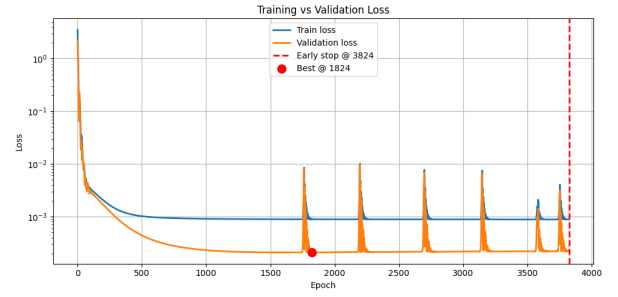
The prediction results show that the inclusion of the data-loss term substantially improves the model behavior under parameter mismatch. In the CPINN case, the predicted total voltage remains in very close agreement with the exact numerical solution over the whole time interval, including transient regions and oscillatory segments. The branch-voltage predictions $V_{\theta,1}$ and $V_{\theta,2}$ also remain close to the reference trajectories, confirming that the hybrid CPINN is able to compensate for the inconsistency introduced by the perturbed residual parameters.

The hybrid PINN also benefits from the addition of the data term, but its deviations remain visibly larger than those of CPINN, particularly in the branch-voltage predictions and in some transient regions of the total voltage response. This difference is also reflected in the quantitative error metrics reported in Table 12. Under the considered parameter-mismatch setting, CPINN achieves lower errors than PINN for all reported outputs. In particular, for the total voltage prediction, CPINN reduces the MSE from 3.01×10^{-4} to 3.29×10^{-5} , the RMSE from 1.73×10^{-2} to 5.73×10^{-3} , and the MAE from 6.84×10^{-3} to 3.98×10^{-3} and the MBE from 1.36×10^{-3} to -3.26×10^{-3} and increased R^2 from 0.995 to 0.999.

These results highlight the practical value of combining physics-based regularization with data supervision. When the embedded physical model is only approximate, the data term acts as a correction mechanism that reduces systematic bias and improves robustness. The benefit of the data-loss term becomes especially evident in this mismatched-parameter regime, which is the practically relevant case for realistic applications. Overall, the results suggest that

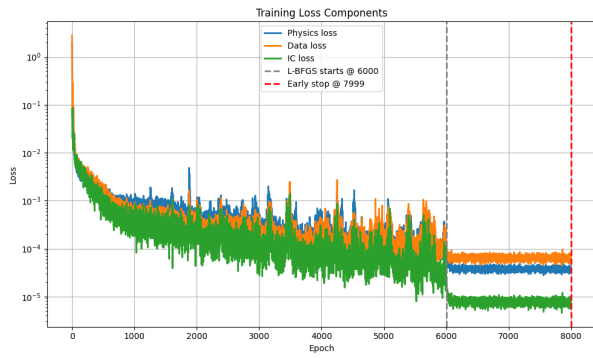


(a) Evolution of the hybrid PINN loss components during training.

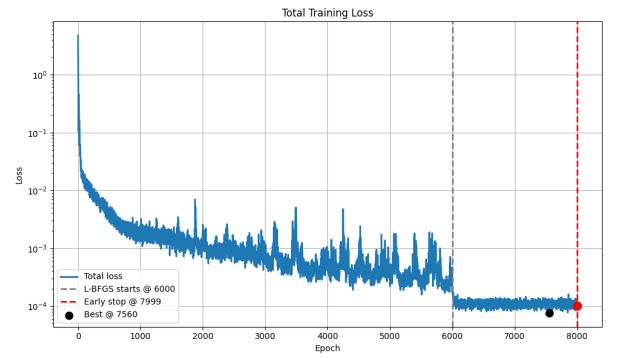


(b) Total training/validation loss and early-stopping behavior for hybrid PINN.

Figure 13: Training behavior of the hybrid PINN under parameter mismatch. The plots show the evolution of the loss terms during optimization and the checkpoint selected by early stopping.

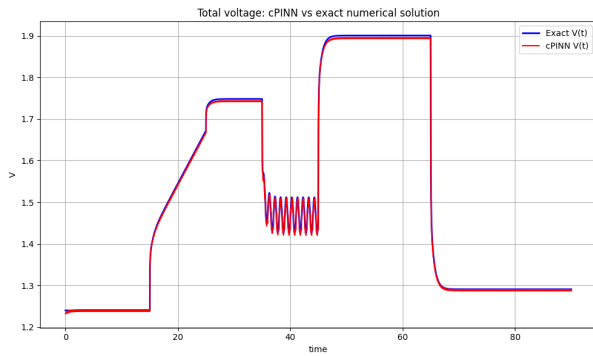


(a) Evolution of the hybrid CPINN loss components during training.

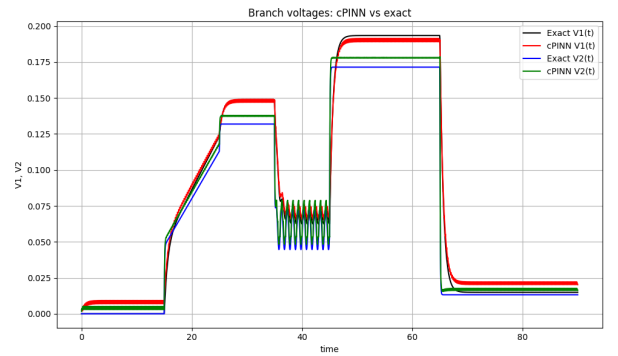


(b) Total training loss and early-stopping behavior for hybrid CPINN.

Figure 14: Training behavior of the hybrid CPINN under parameter mismatch. The plots show stable optimization of the loss components and the selection of the best checkpoint by early stopping.

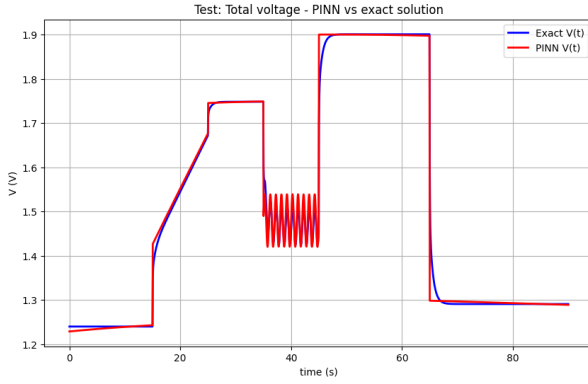


(a) Total voltage prediction of hybrid CPINN versus the exact numerical solution.

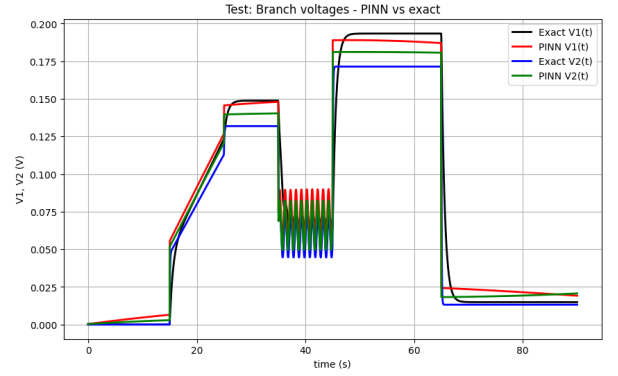


(b) Branch-voltage prediction of hybrid CPINN versus the exact numerical solution.

Figure 15: Prediction performance of the hybrid CPINN under parameter mismatch. The model remains in close agreement with the exact solution for both total voltage and internal branch voltages.



(a) Total voltage prediction of hybrid PINN versus the exact numerical solution.



(b) Branch-voltage prediction of hybrid PINN versus the exact numerical solution.

Figure 16: Prediction performance of the hybrid PINN under parameter mismatch. The data-loss term improves the prediction, although larger deviations remain compared with hybrid CPINN.

Table 12

Prediction performance of the hybrid PINN and hybrid CPINN models under parameter mismatch. The reference trajectories are generated using the nominal ECM parameters, whereas the residual model used during training employs perturbed parameter values.

		MSE (V^2)	RMSE (V)	MAE (V)	MBE (V)	R^2
PINN	V	3.01×10^{-4}	1.73×10^{-2}	6.84×10^{-3}	1.36×10^{-3}	0.995
	V_1	2.08×10^{-4}	1.44×10^{-2}	7.57×10^{-3}	2.09×10^{-3}	0.962
	V_2	6.62×10^{-5}	8.13×10^{-3}	6.62×10^{-3}	6.36×10^{-3}	0.985
CPINN	V	3.29×10^{-5}	5.73×10^{-3}	3.98×10^{-3}	-3.26×10^{-3}	0.999
	V_1	2.67×10^{-5}	5.16×10^{-3}	4.56×10^{-3}	2.99×10^{-3}	0.995
	V_2	2.55×10^{-5}	5.05×10^{-3}	4.87×10^{-3}	4.86×10^{-3}	0.994

Table 13

Optimization strategy and early-stopping configuration for the hybrid PINN and hybrid CPINN models under parameter mismatch.

Component	PINN	CPINN
Maximum epochs	$N_{\text{epochs}} = 8000$	$N_{\text{epochs}} = 8000$
Optimizer (stage 1)	Adam, learning rate 10^{-3}	Adam, learning rate 10^{-3}
Optimizer (stage 2)	L-BFGS, learning rate 1.0	L-BFGS, learning rate 1.0
L-BFGS configuration	max_iter = 20, history_size = 50	max_iter = 20, history_size = 50
Switch epoch	$S_{\text{epoch}} = 6000$	$S_{\text{epoch}} = 6000$
Batch size	Full batch	Full batch
Early stopping criterion	Validation loss	Validation loss
Patience	2000 epochs	2000 epochs
Minimum improvement	10^{-7}	10^{-7}
Physics loss weight	$\lambda_{\text{phys}} = 1.0$	$\lambda_{\text{phys}} = 1.0$
Initial-condition loss weight	$\lambda_{\text{IC}} = 1.0$	$\lambda_{\text{IC}} = 1.0$
Data loss weight	$\lambda_{\text{data}} = 1.0$	$\lambda_{\text{data}} = 1.0$
Best validation epoch	≈ 1824	≈ 7560
Best validation loss	$\approx 2.074681 \times 10^{-4}$	$\approx 7.7406 \times 10^{-5}$
Stopped epoch	≈ 3824	≈ 7999
Training time	32s	224s

Table 14

Output stability of PINN and CPINN under a noisy custom current profile ($\sigma_I = 3.0$ A, horizon 0–90 s, $I \in [0, 190]$ A). Each row gives the corresponding metric of the difference between a model's prediction under noisy current input against its own prediction under the clean current input.

Model	RMSE (V)	MAE (V)	MBE (V)	R^2
PINN	1.28×10^{-2}	1.01×10^{-2}	-3.76×10^{-4}	0.997
CPINN	8.20×10^{-3}	6.39×10^{-3}	-2.21×10^{-3}	0.999

hybrid CPINN formulations are especially promising for PEM electrolyzer modeling when exact knowledge of ECM parameters cannot be assumed.

Under this setting, the total voltage reconstructed from the predicted states $V_{\theta,1}$ and $V_{\theta,2}$ was also computed using the perturbed values of E_{rev} and R_{ohm} , consistently with the assumption that these are the parameter values available to the model in practice. Two training strategies were then compared: a physics-only formulation, based on the physics and initial-condition losses, and a hybrid formulation including the additional data-fidelity term,

The conditioning tuples (I, V_{10}, V_{20}) were sampled from the admissible domain using a Sobol low-discrepancy sequence. A large pool of conditioning cases was generated in advance, and each batch was formed by randomly selecting a subset of cases from this pool. The same selected tuples were used both for neural-network evaluation and for generating the corresponding reference trajectories with the numerical solver, ensuring consistency between the conditioning inputs and the supervised targets. Moreover, the time instants used for the data-loss term were sampled independently of the collocation points used in the physics loss. This reflects the practical situation in which measured data are generally not available at the same points where the residual equations are enforced.

The obtained results show that the inclusion of the data-loss term substantially improves prediction accuracy under parameter mismatch. While the physics-only model is affected by the inconsistency between the residual equations and the reference dynamics, the hybrid formulation is able to compensate for this mismatch and recover trajectories that remain close to the nominal solution. In particular, the hybrid CPINN preserves strong generalization capability across the considered operating conditions despite the use of inaccurate parameter values in the residual model.

These results highlight the practical value of combining physics-based regularization with data supervision. When the embedded physical model is only approximate, the data term acts as a correction mechanism that reduces systematic bias and improves robustness. This suggests that hybrid CPINN formulations are especially promising for realistic PEM electrolyzer applications, where exact knowledge of ECM parameters cannot generally be assumed.

4.4. Robustness Under Noisy Current Measurements

In practical PEM electrolyzer systems, the measured current is inevitably affected by sensor noise, electromagnetic interference, and measurement uncertainties. Since current is one of the inputs to the CPINN, it is important to evaluate whether such disturbances significantly affect the predicted voltage response. To assess the robustness of the proposed model, zero-mean Gaussian noise with a standard deviation of 3 A was added to the input current over the entire operating range of 0–190 A, while the prediction horizon remained 90s. To evaluate the robustness of the proposed models against measurement uncertainty, a noisy current profile was introduced by adding Gaussian noise with a standard deviation of $\sigma_I = 3.0$ A to the measured current input(17). The voltage responses predicted by the PINN and CPINN models were then compared with their corresponding predictions obtained using the clean current profile. Figure 19 presents the voltage prediction of the PINN model under noisy current excitation. As observed, the PINN model remains capable of tracking the overall voltage behavior; however, noticeable deviations appear in transient regions where the current profile changes rapidly. This indicates a higher sensitivity of the standard PINN formulation to current measurement noise.

The quantitative comparison between the PINN and CPINN models is summarized in Table 14. The CPINN achieves lower prediction deviations across all error metrics, reducing the RMSE from 1.28×10^{-2} V for the PINN to 8.20×10^{-3} V and its voltage prediction closely tracks the exact response over the whole horizon (Figure 18). Furthermore, the higher coefficient of determination ($R^2 = 0.999$) demonstrates that the CPINN provides a more stable voltage response under noisy current measurements. These results highlight the benefit of incorporating conditional information in improving the robustness of physics-informed models against input uncertainty.

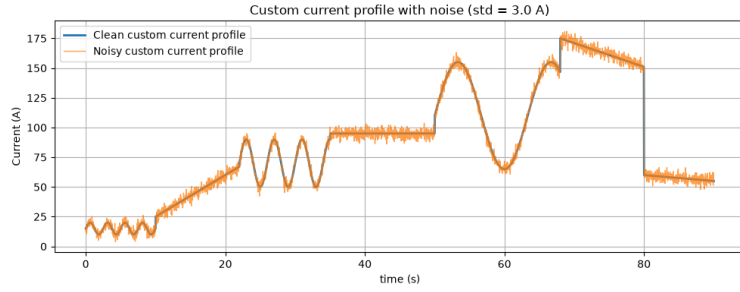


Figure 17: Custom current profile used to evaluate robustness to noisy measurements. The clean profile (blue) is corrupted with additive Gaussian noise ($\sigma_I = 3.0$ A) to produce the noisy profile (orange), which is used as input to the trained PINN and CPINN models.

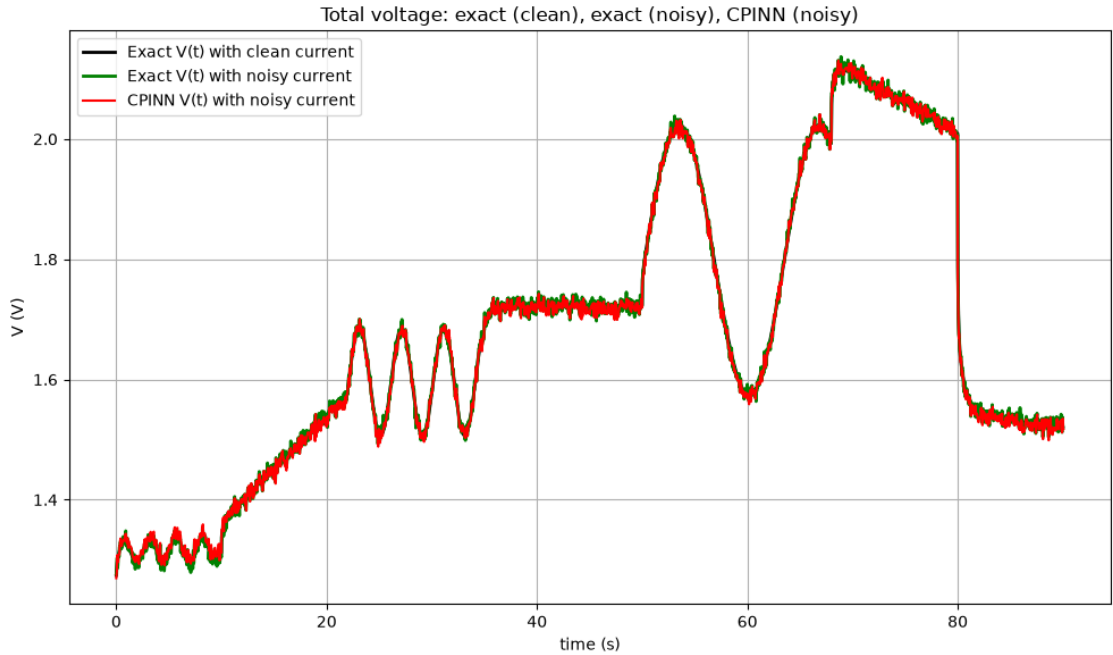


Figure 18: Total voltage prediction of the CPINN model under a noisy current input, compared against the exact voltage response obtained with clean and noisy current. The CPINN prediction closely tracks the exact response across the full time horizon, indicating strong robustness to current measurement noise (RMSE = 8.20×10^{-3} V, $R^2 = 0.998786$).

5. Conclusion and Future Work

This study investigated the capabilities of Physics-Informed Neural Networks (PINN) and Conditional Physics Informed Neural Networks (CPINN) for modeling the second-order equivalent circuit of a PEM electrolyzer under dynamic operating conditions. These frameworks are able to reconstruct the underlying dynamics without the need for extensive labeled data. However, the comparative analysis reveals some differences in their behavior and generalization capacity. Quantitative analysis and evaluations of both frameworks based on the MSE, RMSE and MAE, MBE and R^2 evaluation metrics demonstrated that in the case of voltage prediction tasks, CPINN outperformed the standard PINN. While PINN achieves satisfactory accuracy within the training interval, its performance degrades when extrapolating to longer time horizons. In contrast, CPINN maintains stable predictions and significantly lower error accumulating during extended simulations beyond the training range. These findings highlight the importance of conditional learning strategies in long-term dynamical system modeling and show the limitations of classical PINN in these tasks. The same study has been performed for the case of hybrid training, where a data loss term has been added to the loss to take

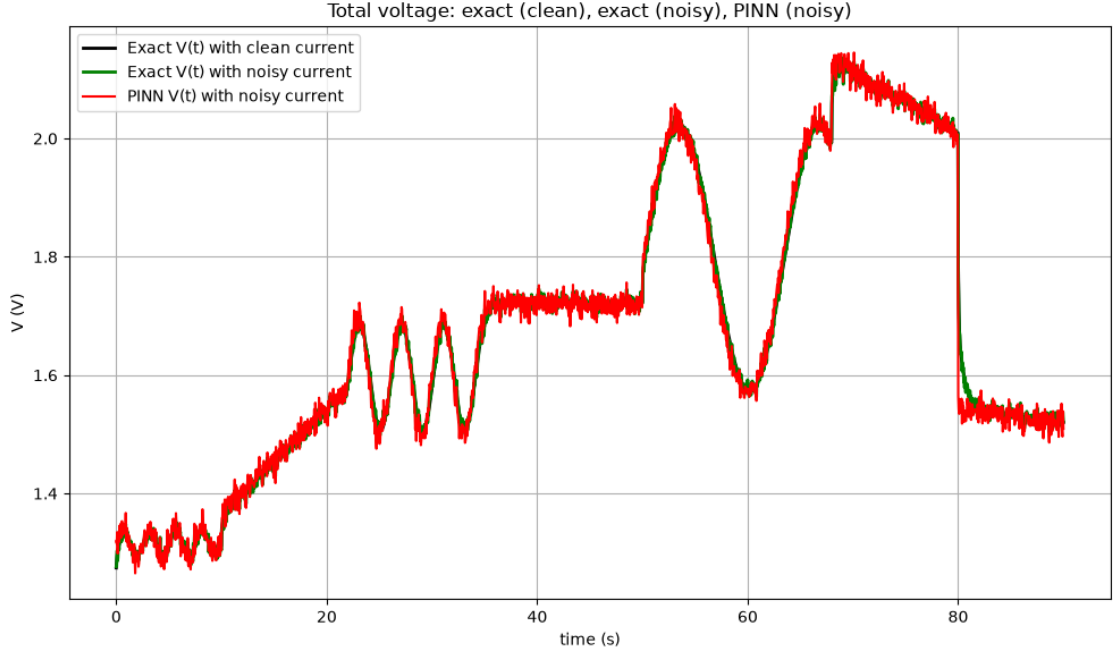


Figure 19: Total voltage prediction of the PINN model under a noisy current input, compared against the exact voltage response obtained with clean and noisy current. Compared to the CPINN, the PINN prediction shows larger deviations from the exact response, particularly around transient regions, reflecting greater sensitivity to current measurement noise (RMSE = 1.28×10^{-2} V, $R^2 = 0.99709$).

into account uncertain knowledge of the physical parameters of the system. Although the data loss term does not compensate completely for the parameter mismatch, in the CPINN case the results are better than in the PINN case. Even if the training time for CPINN is larger than for PINN, this is not relevant from a practical point of view because the training is done off-line and not in real time in the control implementation.

The observed improvements can be attributed to the conditional structure of CPINN, which enables the model to learn a parameterized solution manifold. The findings of this study suggest that conditioning plays an important role in extending physics-informed learning toward applications in control systems.

From a broader perspective, the proposed novel framework contributes to bridging physics-based modeling and data-driven learning for hydrogen energy systems. CPINN with strong generalization capability indicates better ability in the predictions of long-horizon time, digital twins, and advanced control of PEM electrolyzers. These results provide practical insights for selecting suitable physics-informed architectures in industrial contexts.

Future research will focus on the application of these novel methods in higher-fidelity electrochemical models, incorporating experimental and industrial datasets, and investigating real-time deployment for monitoring and control applications. In addition, a natural continuation of this work is the development of a model predictive control (MPC) framework based on the CPINN formulation for real-time monitoring and control of PEM electrolyzers.

A. Error metrics

To objectively compare the performance of PINN and CPINN, there exist a set of complementary error metrics that can compare the NN prediction accuracy with respect to a reference solution (analytical or high-fidelity numerical). These metrics are evaluated on test datasets which are independent from training data and also to assess the trained PINN in- distribution accuracy and out of distribution generalization, which is the main objective of this research. Let $\{t_i\}_{i=1}^N$ denote the discrete time instants of the test dataset, where N is the total number of evaluation samples.

At each time instant t_i , $y(t_i)$ represents the reference solution obtained from the analytical model or a high-fidelity numerical solver, while $\hat{y}(t_i)$ denotes the corresponding prediction produced by the PINN or CPINN model. All evaluation metrics are computed on independent test datasets that are not used during training.

A.1. Mean Squared Error (MSE)

The mean squared error is defined as

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y(t_i) - \hat{y}(t_i))^2. \quad (45)$$

MSE measures the average squared deviation between predicted and reference values and therefore strongly penalizes large errors. Due to the squaring operation: 1. Large deviations dominate the metric 2. Positive and negative errors contribute equally and also the metric is smooth and differentiable (Manoj et al., 2022).

In the context of PINN and CPINN, MSE is particularly informative for detecting error accumulation in long-horizon predictions and models failures under the out of distribution excitation, such as current amplitudes outside the training range.

Lower amount of MSE values for PINN demonstrates the ability of the generalization capacity in the unseen operating conditions.

The root mean squared error is evaluated as

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y(t_i) - \hat{y}(t_i))^2}. \quad (46)$$

RMSE expresses the typical prediction error in the same physical units as the target variable. This make the RMSE directly usable in engineering terms. Without square root, the error is expressed in squared units, which has no direct physical meaning in engineering (Willmott and Matsuura, 2005).

A.2. Mean Absolute Error (MAE)

The mean absolute error is given by

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y(t_i) - \hat{y}(t_i)|, \quad (47)$$

It measures the average absolute deviation between the model predictions and the reference values. Because it does not square the errors, MAE is expressed in the same physical units as the target variable and is less sensitive to large outliers than MSE or RMSE. Therefore, it provides an easily interpretable measure of typical prediction error (Robeson and Willmott, 2023). In electrolyzer modeling and control, predicting voltage with minimal error is imperative to prevent incorrect control actions. In addition, incorrect accuracy is critical for operational and safety reliability.

A.3. Mean Bias Error (MBE)

The mean bias error is defined as

$$\text{MBE} = \frac{1}{N} \sum_{i=1}^N (\hat{y}(t_i) - y(t_i)). \quad (48)$$

MBE measures the average signed difference between the model prediction and the reference solution. Unlike MAE and RMSE, which quantify the magnitude of the prediction error, MBE indicates the direction of the error. A positive MBE means that the model tends to overestimate the reference solution, while a negative MBE indicates systematic underestimation. Therefore, MBE is useful for identifying prediction bias in the voltage response of the PEM electrolyzer. A value close to zero indicates that the positive and negative errors are balanced on average (Legates and McCabe, 1999).

A.4. Coefficient of Determination (R^2)

The coefficient of determination is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^N (y(t_i) - \hat{y}(t_i))^2}{\sum_{i=1}^N (y(t_i) - \bar{y})^2}, \quad (49)$$

where

$$\bar{y} = \frac{1}{N} \sum_{i=1}^N y(t_i) \quad (50)$$

is the mean value of the reference solution.

The R^2 score measures how well the predicted response explains the variance of the reference solution. A value of $R^2 = 1$ indicates perfect agreement between the predicted and reference trajectories, while values closer to zero indicate weaker predictive performance. Negative values can occur when the model prediction is worse than simply using the mean of the reference solution. In this work, R^2 is used as an additional global accuracy indicator to evaluate how well the PINN and CPINN models reproduce the dynamic voltage (Legates and McCabe, 1999; Nash and Sutcliffe, 1970).

References

- Andrew, G., Gao, J., 2007. Scalable training of ℓ_1 -regularized log-linear models, in: Proceedings of the 24th International Conference on Machine Learning (ICML), pp. 33–40.
- Antonelo, E.A., Camponogara, E., Seman, L.O., Jordanou, J.P., de Souza, E.R., Hübner, J.F., 2024. Physics-informed neural nets for control of dynamical systems. *Neurocomputing* 579, 127419.
- Arnold, F., King, R., 2021. State-space modeling for control based on physics-informed neural networks. *Eng. Appl. Artif. Intell.* 101, 104195.
- Asiaban, S., Bozalakov, D., Vandeveld, L., 2024. Development of a dynamic mathematical model of PEM electrolyser for integration into large-scale power systems. *Energy Convers. Manag.* X 23, 100610. <https://doi.org/10.1016/j.ecmx.2024.100610>.
- Cai, S., Wang, Z., Wang, S., Perdikaris, P., Karniadakis, G.E., 2021. Physics-informed neural networks for heat transfer problems. *J. Heat Transfer* 143, 060801.
- Cuomo, S., Di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F., 2022. Scientific machine learning through physics-informed neural networks: Where we are and what's next. *J. Sci. Comput.* 92, 88. <https://doi.org/10.1007/s10915-022-01939-z>.
- Dang, J., Yang, F., Li, Y., Deng, X., Ouyang, M., 2022. Transient behaviors and mathematical model of proton exchange membrane electrolyzer. *J. Power Sources* 542, 231757.
- Dey, S., Mallik, A., 2025. Physics-informed neural network-estimated circuit parameter adaptive modulation of DAB. *IEEE Trans. Power Electron.* 40, 14821–14841.
- Gokhale, G., Claessens, B., Develder, C., 2022. Physics-informed neural networks for control-oriented thermal modeling of buildings. *Appl. Energy* 314, 118852.
- Gosu, D.L., Durvasula, A., Annamalai, J., 2024. Modelling of PEM electrolyzer dynamics for green hydrogen production, in: *Hydrogen Energy System Modelling and Simulation*. Elsevier.
- Guilbert, D., Vitale, G., 2019. Dynamic emulation of a PEM electrolyzer by time constant based exponential model. *Energies* 12, 750. <https://doi.org/10.3390/en12040750>.
- Hidouri, D., Marouani, R., Cherif, A., 2024. Modeling and simulation of a renewable energy PV/PEM with green hydrogen storage. *Eng. Technol. Appl. Sci. Res.* 14, 12543–12548. <https://doi.org/10.48084/etasr.6492>.
- Hodson, T.O., 2022. Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geosci. Model Dev. Discuss.* 2022, 1–10.
- Hurt, S.E., Toliyat, H.A., 2025. Physics informed neural network induction motor equivalent circuit parameter estimation with only electrical measurements, in: Proceedings of the 2025 IEEE International Electric Machines & Drives Conference (IEMDC), Houston, TX, USA, pp. 1081–1086.
- Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. *Nat. Rev. Phys.* 3, 422–440.
- Kim, J., Oh, C., Kim, K., Lee, J., Kim, T., 2022. Parameter identification of electrical equivalent circuits including mass transfer parameters for the selection of the operating frequencies of pulsed PEM water electrolysis. *Energies* 15, 9303. <https://doi.org/10.3390/en15249303>.
- Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Ko, T., Kim, D., Park, J., Lee, S.H., 2025. Physics-informed neural network for long-term prognostics of proton exchange membrane fuel cells. *Appl. Energy* 382, 125318.
- Kovacs, A., Exl, L., Kornell, A., Fischbacher, J., Hovorka, M., Gusenbauer, M., Schrefl, T., 2022. Conditional physics-informed neural networks. *Commun. Nonlinear Sci. Numer. Simul.* 104, 106041.
- Legates, D.R., McCabe, G.J., 1999. Evaluating the use of “goodness-of-fit” measures in hydrologic and hydroclimatic model validation. *Water Resour. Res.* 35, 233–241.
- Manoj, S.O., Ananth, J.P., Rohini, M., Dhanka, B., Pooranam, N., Arumugam, S.R., 2022. FWS-DL: Forecasting wind speed based on deep learning algorithms, in: *Artificial Intelligence for Renewable Energy Systems*. Elsevier, pp. 353–374.
- Mao, X., Tian, Y., Yang, A., Zhang, G., 2024. Identification of equivalent circuit parameters for proton exchange membrane (PEM) electrolyzer engineering models. *IEEE Access* 12, 15509–15524.
- Misyris, G.S., Venzke, A., Chatzivasileiadis, S., 2020. Physics-informed neural networks for power systems, in: 2020 IEEE Power & Energy Society General Meeting (PESGM), Montreal, QC, Canada, pp. 1–5.
- Nash, J.E., Sutcliffe, J.V., 1970. River flow forecasting through conceptual models part I—A discussion of principles. *J. Hydrol.* 10, 282–290.
- Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378, 686–707.

- Raman, K.A., Hammacher, L., Kungl, H., Karl, A., Jodat, E., Eichel, R.A., Karyofylli, V., 2025. Data-driven surrogate modeling for performance prediction and sensitivity analysis of transport properties in proton exchange membrane water electrolyzers. *Appl. Energy* 386, 125529. <https://doi.org/10.1016/j.apenergy.2025.125529>.
- Ratib, M.K., Muttaqi, K.M., Islam, M.R., Sutanto, D., Agalgaonkar, A.P., 2024. Electrical circuit modeling of proton exchange membrane electrolyzer: The state-of-the-art, current challenges, and recommendations. *Int. J. Hydrogen Energy* 49, 625–645. <https://doi.org/10.1016/j.ijhydene.2023.08.319>.
- Robeson, S.M., Willmott, C.J., 2023. Decomposition of the mean absolute error (MAE) into systematic and unsystematic components. *PLoS One* 18, e0279774.
- Sabamehr, M., Batlle, C., Serra-Prat, M., 2026. PINN for parameter identification of equivalent circuit models of PEM electrolyzer. Accepted abstract, European Hydrogen Energy Conference (EHEC).
- Sayed-Ahmed, H., Toldy, Á.I., Santasalo-Aarnio, A., 2024. Dynamic operation of proton exchange membrane electrolyzers—Critical review. *Renew. Sustain. Energy Rev.* 189, 113883. <https://doi.org/10.1016/j.rser.2023.113883>.
- Shahbaz, I., Abdel-Rahman, M.J., Hammad, E., 2025. G-PIFNN: A generalizable physics-informed Fourier neural network framework for electrical circuits. *arXiv preprint arXiv:2512.02712*.
- Stiasny, J., Zhang, B., Chatzivasileiadis, S., 2024. PINNSim: A simulator for power system dynamics based on physics-informed neural networks. *Electr. Power Syst. Res.* 235, 110796. <https://doi.org/10.1016/j.epsr.2024.110796>.
- Taj, R., 2024. Physics-informed neural networks for electrical circuit analysis: Applications in dielectric material modeling. *arXiv preprint arXiv:2411.10483*.
- Willmott, C.J., Matsuura, K., 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Clim. Res.* 30, 79–82.
- Zerrougui, I., Li, Z., Hissel, D., 2025. Physics-informed neural network for modeling and predicting temperature fluctuations in proton exchange membrane electrolysis. *Energy AI* 20, 100474.
- Zhang, X., Chan, S.H., 2022. Data-driven modeling of electrochemical energy systems: A review. *Appl. Energy* 305, 117859.