Tech Science Press

Doi:10.32604/journal.2025.012345

ARTICLE

A Comparative Study of PINN and CPINN for Modeling a Second-Order Equivalent Circuit for PEM Electrolyzers

Milad Sabamehr[1,*], Carles Batlle[2] and Maria Serra[1]

[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens i Artigas 4-6, 08028 Barcelona, Spain
[2]Departament de Matemàtiques, Institut d'Organització i Control, EPSEVG, Universitat Politècnica de Catalunya (UPC), 08800 Vilanova i la Geltrú, Spain
[*]Corresponding Author: Milad Sabamehr. Email: milad.sabamehr@upc.edu

Version March 19, 2026 submitted to Journal Not Specified

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 time-varying conditions have become necessary for the improvement of the technology. 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 PEM electrolyzers. Although both approaches are trained using physics-based losses derived from the system´s differential equations, the CPINN framework incorporates conditioning on operating current and 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) demonstrate that CPINNs accuracy significantly improves that of PINNs, which exhibit larger and increasing deviation when extrapolating beyond the time training domain. These findings highlight the effectiveness of conditional physics-informed learning for dynamical modeling of electrolyzers and motivate future integration of CPINN-based models with model predictive control strategies for real-time optimal control of PEM electrolyzers.

KEYWORDS: Physics-Informed Neural Network; Conditional Physics-Informed Neural Network; Proton Exchange Membrane Electrolyzer Modeling; Equivalent Circuit Model; Hydrogen production; Dynamical System Modeling.

1 Introduction

The necessary transition toward sustainable energy systems has accelerated the global interest in green hydrogen production, a critical vector for the storage of renewable energy that provides flexibility to the energy networks. Among other candidates for hydrogen production, PEM electrolyzers are pivotal for green hydrogen production and have emerged as a leading candidate in different aspects such as high efficiency, compact design, and rapid dynamic response [1,2].

For the improvement of the PEM electrolysis technology, proper modeling is necessary, which will drive to optimized performance, dynamic operation 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 dynamic modeling of PEM electrolyzers becomes more challenging [3,4]. Electrical equivalent circuit models (ECMs) have been widely considered because they capture the dominant dynamics while remaining tractable for identification and real-time use [5,6]. In [14] different equivalent circuit models were used to describe and analyze the transient behaviors of proton exchange membrane 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 [5,6]. To improve the hydrogen production efficiency by identifying the optimal operating frequency for pulsed electrolysis, an ECM is proposed in [13], 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 [5]. 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.

The authors of [7] focused on real-time operational data to refine a dynamical model of the fuel cell to capture 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 [15], 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. The results shown 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, particularly under unseen operating conditions. To overcome these limitations, paper [8] introduced Physics Informed Neural Networks 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 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 [8,9]. With respect to the application of PINNs to electric and

power systems, several works can be found in the literature. [17] focused on using PINNs to solve the dynamics of a power system, specifically the swing equation for a Single Machine Infinite Bus system. 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 [18], a novel 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 [16] 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 [19], where a novel version of PINN was developed for efficient and interpretable electrical circuit analysis. In the study of [21], PINNs demonstrated high precision in current prediction for simple circuits and also in estimating the parameters of simple circuits, while [22] introduced a novel 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, [10] 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 [11] 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, [23] 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 systems when compared to traditional numerical solutions. In a parallel study on PINN in control, [24] 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 longer prediction horizons.

The paper [27] introduced the 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.

A special kind of CPINN, called Physics Informed Neural Network for Control (PINNC), was considered in [25]. 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 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 of PINN and CPINN applied to an ECM of a PEM electrolyzer are compared. Different current profiles and long-range simulation times are analyzed. The ECM model and its corresponding parameters are described in [26].

Since the electrical current that drives the PEM electrolyzer depends on the operating conditions and the control objectives, 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. This 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.

The paper is structured as follows: Section 2 provides the PEM electrolyzer model through an electrical circuit and discusses the governing equations, and 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 modeling. In the section 4, the evaluation metrics used to assess the models performance are introduced. The results are presented in section 5, where the prediction accuracy and generalization capacity of PINN and CPINN are compared. Finally, section 6 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

To capture the dynamic behavior of a PEM electrolyzer, the study [26] introduced the second-order ECM shown in Figure 1. 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.

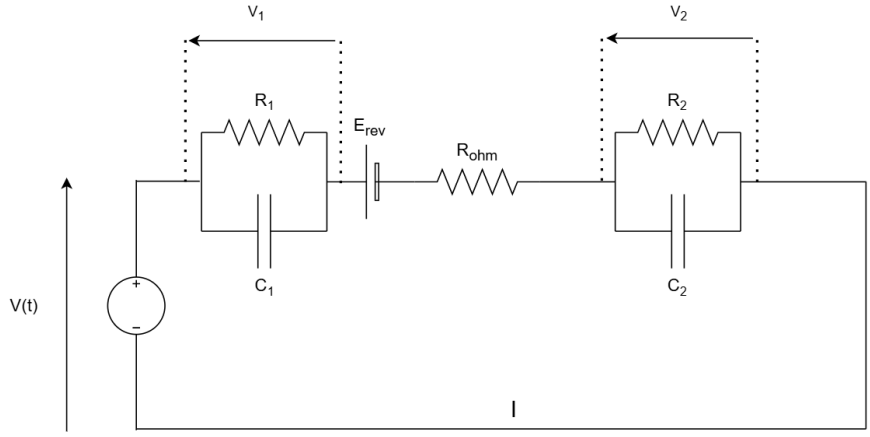


Figure 1: Second-order equivalent circuit model of the PEM electrolyzer

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 an dynamic 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 [26] and are treated as known constant values throughout this work.

The expression of the total voltage is

$$V = E_{\text{rev}} + R_{\text{ohm}} I + V_1 + V_2 \tag{1}$$

The power supplied to the electrolyzer is $I(t)V(t)$. However, because of different losses, the useful power for hydrogen production is not all the supplied power.

2.1 Parameter Values and Time-Scale Analysis

The numerical values of the equivalent circuit parameters are adopted from the identification results reported in [26] and are treated as fixed constants in this study. The parameters are used in this simulation are :

Table 1: Equivalent circuit model parameters of the PEM electrolyzer, taken from [26].

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 presence of two different time constants shows a clear separation of electrolyzer dynamics. The branch with larger time constant shows slower polarization effects, while the other branch shows faster transient response. These different dynamics enables the model to correctly reproduce the voltage time response under different operating condition, particularly during rapid current variations.

3 Methods

3.1 Physics-Informed Neural Networks (PINNs)

The PINNs are neural networks that, in the most general form, integrate experimental data with known differential equations to enhance the performance of neural networks. A hybrid loss function composed of data, boundary(Initial Condition), and physics terms is minimized during the training process [10,12].

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

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

is mapped to the output vector

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

according to

$$y_j = r\left(\sum_i w_{i,j} x_i + b_j\right), \tag{2}$$

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. Since we are considering systems which have an arbitrary control input $u(t)$, we add a second port to the NN. Figure 2 displays a scheme of a PINN for this case.

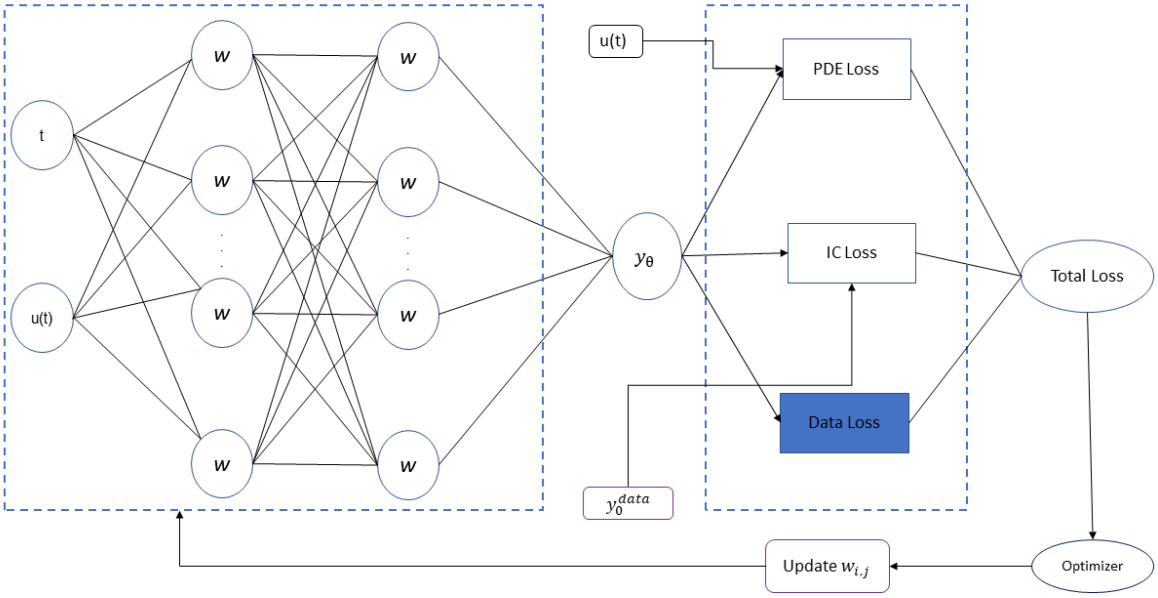


Figure 2: Schematic illustration of a physics-informed neural network. The neural network receives inputs t (time) and $u(t)$ (control input), and produces the predicted system response y_θ, representing the dynamic behavior of the dynamic system. Instead of relying solely on data, in this framework physical knowledge is embedded into the learning process through multiple loss components, including the governing differential equations that describe the dynamics of the system, and the initial condition (IC) loss. The input $u(t)$, representing an external forcing, is also fed into the loss function associated to the dynamics. These losses are summed into a loss function, which is minimized using a gradient-based optimizer to iteratively update the weights $w_{i,j}$ and improve the prediction capacity and accuracy.

3.1.1 Problem Setup

Following [8], the objective of this study is to approximate the solution of a governing ordinary differential equation (ODE), however, in this study which can be expressed in a general operator

$$y_t + \mathcal{N}(y, u) = 0 \tag{3a}$$

$$y_t = \frac{dy}{dt} \tag{3b}$$

$$y(0) = y_0 \tag{3c}$$

where $\mathcal{N}[\cdot]$ denotes a nonlinear or linear function representing the underlying physical laws of the system defined over time interval $t \in [0, T]$. This operator may include spatial derivatives, temporal derivatives, or a combination of both, and can also incorporate algebraic constraints depending on the structure of the physical model.

The objective is to construct an approximation $\hat{y}(t)$ that satisfies the operator equation (3) while respecting any prescribed initial conditions.

Physics Loss ($\mathcal{L}_{phys}$)

The physics loss enforces consistency between the neural network approximation and the governing ordinary differential equation (ODE). Let the governing equation be written in operator form as defined in the equation 3a, where $\mathcal{N}[.]$ denotes the function associated with the physical system and y_t as differential equation of the system.

In the Physics-Informed Neural Network (PINN) framework, the neural network approximation $y_\theta(t_j)$ 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 sampled within the domain $[0, T]$:

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_r} \sum_{j=1}^{N_r} \left| dy_\theta(t_j)/d(t_j) + N\left[y_\theta(t_j), u(t_j)\right] \right|^2, \tag{4}$$

where

- N_r denotes the number of collocation (residual) points.
- $u(t_j)$ j-th collocation point in the time interval $[0, T]$.
- $y_\theta(t_j)$ is the neural network output parameterized by θ evaluated at that time j.
- $\mathcal{N}[y_\theta(t_j)]$ denotes the governing differential operator applied to the network output, yielding the ODE residual.

Intial Condition Loss ($\mathcal{L}_{IC}$)

This term enforces the known initial state of the dynamical system. For ODEs, the solution trajectory must start from the prescribed initial condition at t = 0 . The loss penalizes any mismatch between the neural network prediction and the known initial values:

$$\mathcal{L}_{ic} = \left| y_\theta(t = 0) - y_0^{\text{data}} \right|^2 \tag{5}$$

where $y_\theta(t = 0)$ denotes the neural network output parameterized by θ evaluated at $t = 0$.

Data Loss ($\mathcal{L}_{data}$)

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 between 0 and T.

The data loss is defined as the mean squared error between the predicted and true values:

$$\mathcal{L}_{\text{data}} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left| y_\theta(t_i) - y^{\text{data}}(t_i) \right|^2, \tag{6}$$

where

- N_d denotes the number of observed data points.
- (t_i) i-th observation time point in the time interval $[0, T]$.
- $y^{\text{data}}(t_i)$ 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.

Minimizing $\mathcal{L}_{\text{data}}$ ensures that the learned solution remains consistent with the available measurements, where the physics-loss terms defined as 4 enforce compliance with the governing equations. The combination of data and physics losses enables improved accuracy, robustness, and generalization under unseen operating conditions.

In this work we apply this PINN framework to the ECM of the PEM electrolyzer described in Section 2, but only the physical loss and initial condition loss are included, while data loss is not considered. This makes sense because we assume that the model of the system is exactly known; if this was not the case, the data loss term would have to be introduced in order to identify the parameters [35].

3.1.2 Physics-Informed Neural Network Architecture:

A Physics-Informed Neural Network (PINN) is employed to approximate the solution of the governing equations. The neural network learns a mapping from time and current inputs to the internal state variables and output voltage:

$$(t, I(t)) \longmapsto (V_1(t), V_2(t), V(t)). \tag{7}$$

The network is implemented as a fully connected feedforward neural network with architecture detailed in the table 2. According to Table 2, the input layer, consisting of normalized time and current (dimension 2), predicts total voltage and two RC branch voltages (dimension 3). The hidden layers comprise three layers of 64 neurons each, using hyperbolic tangent activation functions for smooth differentiability, which is essential for automatic differentiation (AD).

The training objective combines physics-based residuals (ODE-based residuals) with the enforcement of the initial conditions. Since in this work we assume that the differential equations describing the system are exactly known, including the values of the physical parameters, data loss is not included.

The residuals associated with the differential equations are defined as

Table 2: Neural network architecture used in the PINN model

Parameter	Value
Input dimension	2
Number of hidden layers	3
Neurons per hidden layer	64
Activation function	$\tanh(\cdot)$
Output dimension	3

$$RES_1(t) = R_1 C_1 \frac{dV_1(t)}{dt} - R_1 I(t) + V_1(t), \tag{8}$$

$$RES_2(t) = R_2 C_2 \frac{dV_2(t)}{dt} - R_2 I(t) + V_2(t), \tag{9}$$

$$RES_3(t) = V(t) - E_{\text{rev}} - R_{\text{ohm}} I(t) - V_1(t) - V_2(t). \tag{10}$$

The physics loss is evaluated at N_r collocation points $\{t_i\}_{i=1}^{N_r}$ and is defined as

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \Big[(RES_1(t_i))^2 + (RES_2(t_i))^2 + (RES_3(t_i))^2 \Big]. \tag{11}$$

The initial conditions of the state variables are enforced by penalizing the deviation between the predicted and prescribed initial voltages at $t = 0$. The initial-condition loss is defined as

$$\mathcal{L}_{\text{IC}} = \frac{1}{2} \Big[\big(\hat{V}_1(t=0) - V_1(t=0)\big)^2 + \big(\hat{V}_2(t=0) - V_2(t=0)\big)^2 \Big]. \tag{12}$$

and the total loss function is given by

$$\mathcal{L} = \lambda_{\text{phys}} \mathcal{L}_{\text{phys}} + \lambda_{\text{bc}} \mathcal{L}_{\text{bc}}, \tag{13}$$

with equal weighting factors $\lambda_{\text{phys}} = \lambda_{\text{bc}} = 1$, However, these values can be adjusted when necessary.

3.1.3 Training and Validation Strategy

To ensure robust generalization, three types of dataset is 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 to excite a wide range of system dynamics. The validation current is designed with switching time, amplitudes, and waveforms , 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 epoch, the physics-informed loss defined in (4) 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 700 consecutive epochs.

The model parameters 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 [33] 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.

3.2 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. In this work, the input conditions are the control input and the initial conditions. Let $\hat{y}_{\boldsymbol{\theta}}(t)$ denote the neural network approximation of the system state, obtained by evaluating the CPINN with inputs $(t, \mathbf{u}(\mathbf{t}), \mathbf{y}_0)$, where $\mathbf{u}(t) \in \mathbb{R}^m$ represents the control input and $\mathbf{y}(0) \in \mathbb{R}^p$ is a vector of initial conditions.

Table 3: Neural network architecture used in the CPINN model

Parameter	**Value**
Input dimension	4
Number of hidden layers	3
Neurons per hidden layer	64
Activation function	$\tanh(\cdot)$
Output dimension	3

The total loss function is defined as

$$\mathcal{L}_{\text{total}} = \lambda_{\text{phys}}\,\mathcal{L}_{\text{phys}} + \lambda_{\text{ic}}\,\mathcal{L}_{\text{ic}} + \lambda_{\text{data}}\,\mathcal{L}_{\text{data}}, \tag{14}$$

where λ_{ic} and λ_{data} are weighting coefficients controlling the relative contribution of initial conditions and data fidelity terms. As in the case of PINN, in this work only physical loss and initial condition is considered and data loss is not implemented.

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

The physics loss is defined as

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_p N_c} \sum_{i=1}^{N_p} \sum_{j=1}^{N_c} \left\| \frac{d\mathbf{y}_\theta}{dt_j}(t_j;\lambda_i) + \mathcal{N}\left[\mathbf{y}_\theta(t_j;\lambda_i)\right] \right\|_2^2 \tag{15}$$

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

3.2.2 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.

Let $\lambda_i \in \Lambda$, $i = 1, \ldots, N_p$, denote the set of Sobol-sampled conditioning variables, and let the initial state corresponding to λ_i be given by $\mathbf{y}_0(\lambda_i)$. The neural network approximation is denoted by $\mathbf{y}_\theta(t;\lambda)$.

The initial-condition loss is defined as

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \|\mathbf{y}_\theta(0;\lambda_i) - \mathbf{y}_0(\lambda_i)\|_2^2, \tag{16}$$

where $\mathbf{y}_\theta(0;\lambda_i)$ represents the network prediction evaluated at the initial time $t = 0$ for the conditioning parameter λ_i, and $\mathbf{y}_0(\lambda_i)$ denotes the corresponding prescribed initial state.

3.2.3 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.

Let $\lambda_i \in \Lambda$, $i = 1, \ldots, N_p$, denote the conditioning variables, and let $\{t_k\}_{k=1}^{N_d}$ represent the measurement time instances. The observed data are denoted by $\mathbf{y}^{\text{data}}(t_k;\lambda_i)$.

The data loss is defined as

$$\mathcal{L}_{\text{data}} = \frac{1}{N_p N_d} \sum_{i=1}^{N_p} \sum_{k=1}^{N_d} \left\| \mathbf{y}_\theta(t_k;\lambda_i) - \mathbf{y}^{\text{data}}(t_k;\lambda_i) \right\|_2^2. \tag{17}$$

3.2.4 Training objective

The network parameters $\boldsymbol{\theta}$ are obtained by solving

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \ \mathcal{L}_{\text{total}} \tag{18}$$

In practice, training is performed using a two-stage optimization strategy combining stochastic gradient-based optimization(Adam) for rapid convergence and a quasi-Newton optimization(LBFGS) for accurate enforcement of the physics constraints.

3.2.5 Sequential Long-Horizon Prediction

The CPINN is trained over a finite temporal domain $t \in [0, T]$. To enable prediction over a longer horizon $[0, t_{\text{final}}]$ without extrapolating beyond the training domain, a sequential simulation strategy is employed.

The global time horizon is partitioned into K sub-intervals,

$$[t_k, t_{k+1}] = [kT_{\text{sim}}, (k+1)T_{\text{sim}}], \quad k = 0, \ldots, K-1, \tag{19}$$

where $T_{\text{sim}} \leq T$. Within each sub-interval, inputs and conditioning parameters are assumed to be constant or slowly varying and are approximated by representative values.

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

$$\hat{\mathbf{y}}^{(k)}(\tau) = \mathcal{N}_{\theta^\star}\left(\tau,\ \mathbf{u}(\mathbf{t})_k,\ \mathbf{y}_0^{(k)}\right), \qquad \tau \in [0, T_{\text{sim}}], \tag{20}$$

where $\mathbf{y}_0^{(k)}$ denotes the initial condition at the beginning of the interval. State continuity is enforced by updating the initial condition for the next interval as

$$\mathbf{y}_0^{(k+1)} = \hat{\mathbf{y}}^{(k)}(T_{\text{sim}}) \tag{21}$$

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.

This proposed Conditional Physics-Informed Neural Network (CPINN) used to model the transient voltage response of a PEM electrolyzer represented by a second-order electrical equivalent circuit model (ECM). 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. By conditioning the network on the operating current and the initial capacitor voltages, a single trained model can predict trajectories for many operating points without retraining. The inputs of the CPINN are u, the input of the system (electrical current in our case), $y(0)$ which is the initial condition of the system (the voltages in our model) and time t, while y_θ is the output of the neural network (see Figure 3).

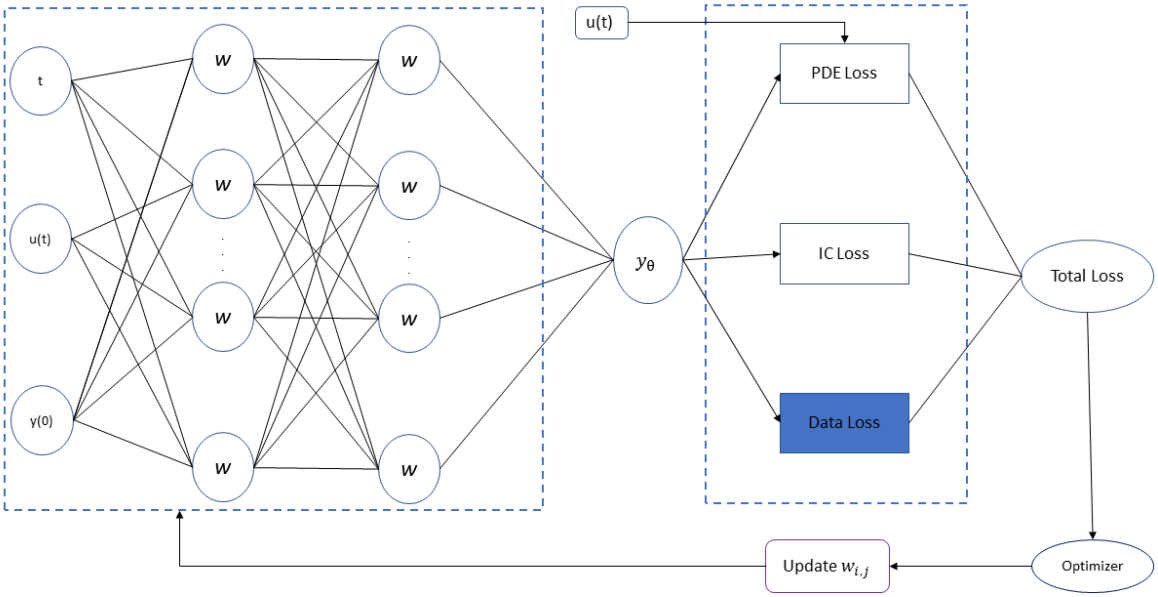


Figure 3: The CPINN network has the initial state $y(0)$ of the dynamic system and u as inputs, as well as time t. y_θ is the neural network prediction of the state of the system at time $t \in [0, T]$ for the given initial condition and input u.

3.2.6 Physical model and problem definition

Second-order ECM:

The system of ODE that governs the electrolyzer ECM is adopted from [26].

Initial conditions:

The capacitor voltages at $t = 0$ are

$$V_1(0) = V_{10}, \qquad V_2(0) = V_{20}. \tag{22}$$

Conditioning domain

We define the input–conditioning vector

$$\lambda := (I, V_{10}, V_{20}), \tag{23}$$

and seek a model that approximates the solution mapping over the admissible domain

$$\Omega := \left\{ (t, I, V_{10}, V_{20}) \;\middle|\; t \in [0, T],\ I \in [I_{\min}, I_{\max}],\ V_{10} \in [V_{10}^{\min}, V_{10}^{\max}],\ V_{20} \in [V_{20}^{\min}, V_{20}^{\max}] \right\}. \tag{24}$$

A natural choice is $T = \max(R_1C_1, R_2C_2)$, which covers the slowest time constant. For the initial-condition ranges, a physically consistent bound is

$$V_{10}^{\max} = R_1 I_{\max}, \qquad V_{20}^{\max} = R_2 I_{\max}, \tag{25}$$

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

3.2.7 Conditional PINN formulation

Operator learning viewpoint:

The goal is to approximate the solution operator

$$\mathcal{G} : \ (t, \lambda) \mapsto (V(t), V_1(t), V_2(t)), \tag{26}$$

so that one trained surrogate can be queried for different (I, V_{10}, V_{20}).

Neural network approximation

We introduce a fully-connected network $\mathcal{N}_\theta$ with parameters θ:

$$(\hat{V}(t), \hat{V}_1(t), \hat{V}_2(t)) = \mathcal{N}_\theta\big(t, I, V_{10}, V_{20}\big), \tag{27}$$

where $\hat{\cdot}$ denotes the network outputs. The architecture used in this work is

Input: $[t,\ I,\ V_{10},\ V_{20}] \in \mathbb{R}^4 \ \rightarrow$ FC(64)-Tanh $\rightarrow$ FC(64)-Tanh $\rightarrow$ FC(64)-Tanh $\rightarrow\ [\hat{V}, \hat{V}_1, \hat{V}_2] \in \mathbb{R}^3$.

3.2.8 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

$$x^{\mathrm{n}} = 2\frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1. \tag{28}$$

We apply (28) to t, I, V_{10}, and V_{20} using the bounds in (24)–(25). This ensures comparable magnitudes across inputs and improves gradient flow during optimization.

3.2.9 Physics residuals and automatic differentiation

Physics-informed training requires time derivatives of the predicted states. Using automatic differentiation (AD), the derivatives are obtained as

$$\frac{d\hat{V}_1}{dt}(t) = \frac{\partial \hat{V}_1}{\partial t}, \qquad \frac{d\hat{V}_2}{dt}(t) = \frac{\partial \hat{V}_2}{\partial t}. \tag{29}$$

At collocation points in Ω, we define the residuals

$$RES_1(t, I, V_{10}, V_{20}) = R_1 C_1 \frac{\partial \hat{V}_1}{\partial t} - R_1 I + \hat{V}_1, \tag{30}$$

$$RES_2(t, I, V_{10}, V_{20}) = R_2 C_2 \frac{\partial \hat{V}_2}{\partial t} - R_2 I + \hat{V}_2, \tag{31}$$

$$RES_3(t, I, V_{10}, V_{20}) = \hat{V} - E_{\mathrm{rev}} - R_{\mathrm{ohm}} I - \hat{V}_1 - \hat{V}_2. \tag{32}$$

Residuals RES_1 and RES_2 enforce the dynamics, while RES_3 enforces voltage consistency.

3.2.10 Training objective: physics and initial-condition losses

Physics loss:

Let $\{(t_i, I_i, V_{10,i}, V_{20,i})\}_{i=1}^{N_r}$ denote collocation points. The physics loss is

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left(RES_1(\cdot)_i^2 + RES_2(\cdot)_i^2 + RES_3(\cdot)_i^2 \right). \tag{33}$$

Initial-condition loss

To enforce (22) across the conditioning domain, we sample N_p conditioning tuples $\{(I_j, V_{10,j}, V_{20,j})\}_{j=1}^{N_p}$ and evaluate at $t = 0$:

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_p} \sum_{j=1}^{N_p} \Big[\big(\hat{V}_1(0, I_j, V_{10,j}, V_{20,j}) - V_{10,j} \big)^2 + \big(\hat{V}_2(0, I_j, V_{10,j}, V_{20,j}) - V_{20,j} \big)^2 \Big]. \tag{34}$$

Total loss

The total loss is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{phys}} + \lambda_{\text{ic}} \, \mathcal{L}_{\text{ic}}, \tag{35}$$

where λ_{IC} is a weighting factor controlling the strength of IC enforcement.

3.2.11 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

$$I_j = I_{\min} + \xi_{j,1} \, (I_{\max} - I_{\min}), \tag{36}$$

$$V_{10,j} = V_{10}^{\min} + \xi_{j,2} \, (V_{10}^{\max} - V_{10}^{\min}), \tag{37}$$

$$V_{20,j} = V_{20}^{\min} + \xi_{j,3} \, (V_{20}^{\max} - V_{20}^{\min}). \tag{38}$$

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

$$N_r = N_p \, N_c. \tag{39}$$

The physics residuals (30), (31) and (32) are evaluated at these points.

3.2.12 Optimization: Adam pre-training and L-BFGS refinement

Training uses a two-stage optimization strategy:

1. **Adam phase** ($e < S_{\text{epoch}}$): stochastic first-order updates to rapidly reduce residuals and reach a good basin of attraction.

2. **L-BFGS phase** ($e \geq S_{\text{epoch}}$): quasi-Newton refinement using a closure function, enabling high-precision minimization of the smooth residual-based loss.

This combination is commonly effective for PINN-type objectives because Adam is robust in early training, whereas L-BFGS can achieve lower final residuals once near an optimum.

3.2.13 Validation and early stopping

To monitor generalization over the conditioning domain, a validation loss is computed at each epoch using an *independent* Sobol sequence and time collocation set. The validation loss uses the same form as (35):

$$\mathcal{L}_{\text{val}} = \mathcal{L}_{\text{phys,val}} + \lambda_{\text{IC}}\, \mathcal{L}_{\text{IC,val}}. \tag{40}$$

Early stopping is applied by tracking the best validation loss $\mathcal{L}^{\star}_{\text{val}}$. If $\mathcal{L}_{\text{val}}$ does not improve by at least δ for a patience window of P epochs, training is terminated and the model is restored to the best checkpoint.

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

1. **Conditioning sampling:** draw N_{p} conditioning 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 $\{t_i\}_{i=1}^{N_{\text{c}}} \subset [0, T]$ (e.g., uniform or Sobol in 1D).

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

$$\mathcal{B} = \left\{ (t_i, \lambda_j) \right\}_{i=1,\dots,N_{\text{c}}}^{j=1,\dots,N_{\text{p}}}, \qquad |\mathcal{B}| = N_{\text{r}} = N_{\text{p}}\, N_{\text{c}}. \tag{41}$$

At each point in $\mathcal{B}$, the network outputs are evaluated, automatic differentiation is used to compute $\partial \hat{V}_1 / \partial t$ and $\partial \hat{V}_2 / \partial t$, and the physics residuals are computed to form $\mathcal{L}_{\text{phys}}$ in (33). 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., $N_{\text{r}} = N_{\text{p}} N_{\text{c}}$. 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}.

3.2.15 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

$$\{\lambda_j^{\text{train}}\}_{j=1}^{N_{\text{p}}} \sim \text{Sobol}([0,1]^3), \qquad \{\lambda_j^{\text{val}}\}_{j=1}^{N_{\text{p,val}}} \sim \text{Sobol}([0,1]^3), \tag{42}$$

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 3.2.11.

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.

3.2.16 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 discretization of current

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

$$I_k := I\left(\frac{t_k + t_{k+1}}{2}\right), \qquad [t_k, t_{k+1}] = [kT_{\text{sim}}, (k+1)T_{\text{sim}}], \tag{43}$$

for $k = 0, \ldots, K-1$, 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.

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

$$(\hat{V}^{(k)}(\tau), \hat{V}_1^{(k)}(\tau), \hat{V}_2^{(k)}(\tau)) = \mathcal{N}_{\theta^\star}(\tau, I_k, V_{10}^{(k)}, V_{20}^{(k)}), \qquad \tau \in [0, T_{\text{sim}}] \tag{44}$$

where $\theta^\star$ denotes the trained parameters and $\tau = t\text{-}t_k$ 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:

$$V_{10}^{(k+1)} = \hat{V}_1^{(k)}(T_{\text{sim}}), \qquad V_{20}^{(k+1)} = \hat{V}_2^{(k)}(T_{\text{sim}}). \tag{45}$$

This ensures continuity of capacitor voltages across interval boundaries, which is physically required because capacitor voltages cannot change instantaneously. The same trained network $\mathcal{N}_{\theta^\star}$

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 $\hat{V}(t)$, $\hat{V}_1(t)$, and $\hat{V}_2(t)$ over $[0, t_{\text{final}}]$

4 Performance Evaluation 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.

4.1 Mean Squared Error (MSE)

The mean squared error is defined as

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

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 [28].

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.

4.2 Root Mean Squared Error (RMSE)

The root mean squared error is evaluated as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y(t_i) - \hat{y}(t_i))^2}. \tag{47}$$

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 [30].

4.3 Mean Absolute Error (MAE)

The mean absolute error is given by

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y(t_i) - \hat{y}(t_i)|, \tag{48}$$

and it quantifies the worst-case deviation between the prediction of the model and the reference solution. The key characteristics of MAE are listed like: 1. Extremely sensitive to outliers 2. detects catastrophic prediction failures[31]. It is imperative in electrolyzer modeling and control to predict voltage with the least amount of error that could lead to incorrect control actions. In addition , incorrect accuracy is critical for operational and safety reliability.

5 Results

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.

5.1 Same range of time and current

The input current used in training for the PINN case is shown in Figure 4. 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.

The neural network architecture of PINN, is shown in Table 5, with 3 hidden layers with 64 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 8, 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 5858 epoch with $2.06.10^{-6}$ validation loss that is the best validation loss and training terminated at epoch 6558 after checking for the lower validation loss than the best validation loss to continue the training. With the same architecture and optimization approaches shown in table 6 and 8, implemented in CPINN, the train stopped at epoch 6078 with best validation loss $4.87.10^{-6}$ and the training ended at epoch 6558.

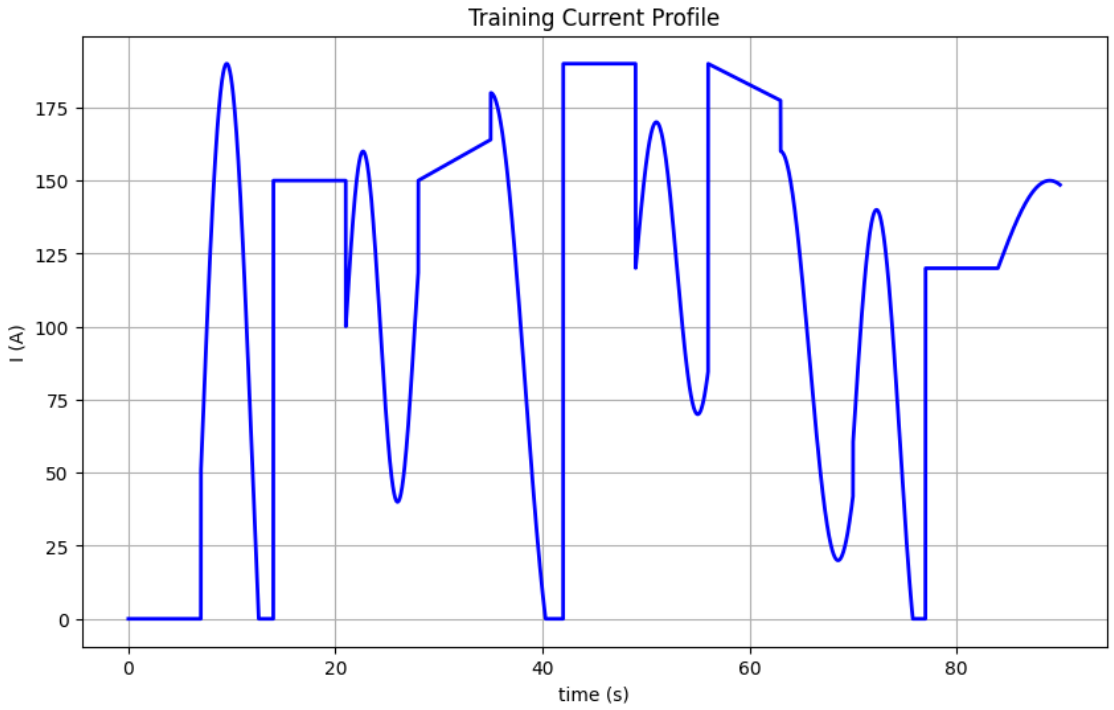


Figure 4: Current profile used in training for the PINN case

After training CPINN and PINN, the objective is to study the generalization of these models by test datasets. In Figure 7, as can be seen by comparing the performance of the each models, CPINN shows a better performance in the case of doing prediction and generalization compare to the PINN and it extrapolate slightly than PINN. To compare the prediction accuracy of CPINN and PINN in the context of voltages of each branches, figure 8, CPINN outperformed the PINN and shows a better performance. In addition to figures that are comparing the performance of the models, Table 9, quantitatively compares the models by evaluation metrics explained in the section 4 which represents that CPINN with MSE:1.62×10^{-5} , RMSE:4.03×10^{-3} and MAE:2.05×10^{-3} outperforms the PINN.

Table 4: Dataset splits and corresponding current excitation profiles

Dataset	Points	Time range (s)	Current profile	Current 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

5.2 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

Table 5: PINN architecture and input normalization settings

Component	Setting	Notes
Model type	Fully Connected Neural Network	
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, 64, 64, 64, 3]	Three hidden layers with 64 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	Fully Connected Neural Network	
Input dimension	4	Time t and input current $I(t)$, branch voltages V1(t=0) and V2(t=0)
Output dimension	3	Total voltage V, branch voltages V_1, V_2
Hidden layers	3	Fully connected layers
Layer sizes	[2, 64, 64, 64, 3]	Three hidden layers with 64 neurons each
Activation function	tanh	Smooth activation suitable for automatic differentiation
Time normalization	$t \in [-1, 1]$	Using training time range $[t_{\min}, t_{\max}]$
Current normalization	$I \in [-1, 1]$	Using $I_{\min} = 0$, $I_{\max} = 190$ A
Voltages branch (V1)	$V_1 \in [-1, 1]$	Using $V1_{\min} = 0$, $V1_{\max} = 0.28272$
Voltages branch (V2)	$V_2 \in [-1, 1]$	Using $V2_{\min} = 0$,$V2_{\max} = 0.25061$

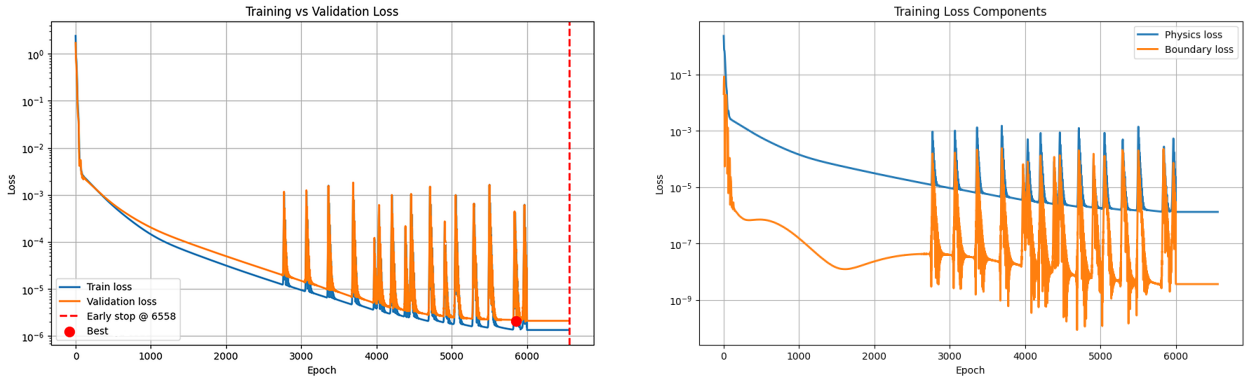
to maintain accurate predictions when simulating longer operating periods than those seen during training.

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, 9 , while both of them show better performance during the training time but beyond the training time , after 90seconds, 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) compared to the standard PINN under extended-time testing conditions. Although both models produce accurate predictions 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.

Table 7: Optimization strategy and early stopping configuration for PINN

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	700 epochs	Training stops after 700 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
Best validation epoch	≈ 5858	Lowest validation loss observed
Best validation loss	$\approx 2.06 \times 10^{-6}$	Selected model checkpoint
Stopped epoch	≈ 6558	Training terminated by early stopping



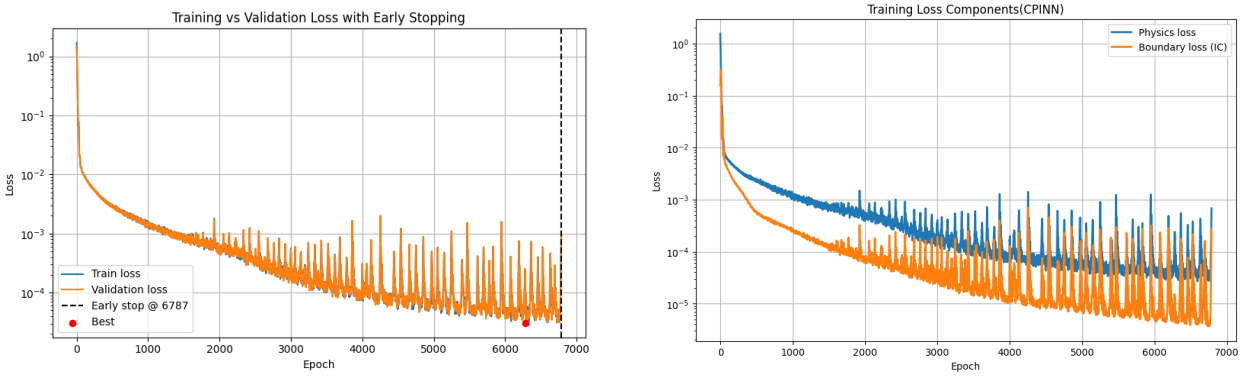
(a) **(b)**

Figure 5: (a) Evolution of the PINN training and validation losses during optimization on a logarithmic scale. Both losses decreases consistently, showing stable convergence of PINN. The red marker denotes the epoch with 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 decreasing of both terms demonstrates that the PINN simultaneously minimizes the governing-equation residual while satisfying imposed constraints, confirming stable physics-informed learning dynamics.

The reason behind the degradation observed in the PINN model shown in figure 9a 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 parameterized solution operator conditioned on the initial state and input current, which improves the stability of the model and mitigates error accumulation in long-time horizon.

Table 8: Optimization strategy and early stopping configuration for 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	700 epochs	Training stops after 700 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
Best validation epoch	≈ 6078	Lowest validation loss observed
Best validation loss	$\approx 4.87 \times 10^{-6}$	Selected model checkpoint
Stopped epoch	≈ 6787	Training terminated by early stopping



(a) **(b)**

Figure 6: (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 capability across different conditioning parameters. (b) Breakdown of the CPINN training loss into its main components, including the physics-based (PDE) loss and the Initial Condition loss. The consistent decrease of these terms shows that the CPINN effectively enforces the governing equations while respecting imposed constraints, demonstrating robust and stable conditional physics-informed learning behavior.

6 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,

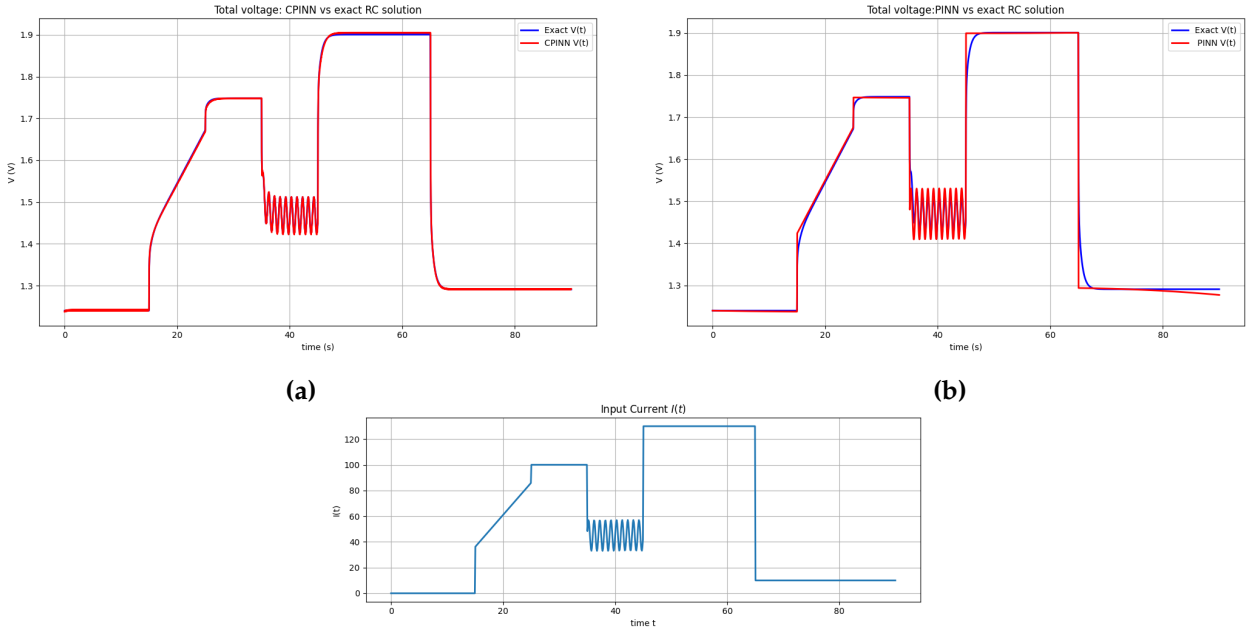


Figure 7: (a) Voltage prediction obtained using the CPINN model, showing strong agreement with the exact RC solution across dynamic transients and oscillatory regions. (b) Voltage prediction obtained using the PINN model under the same test conditions. While the PINN captures the general voltage trend, 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)

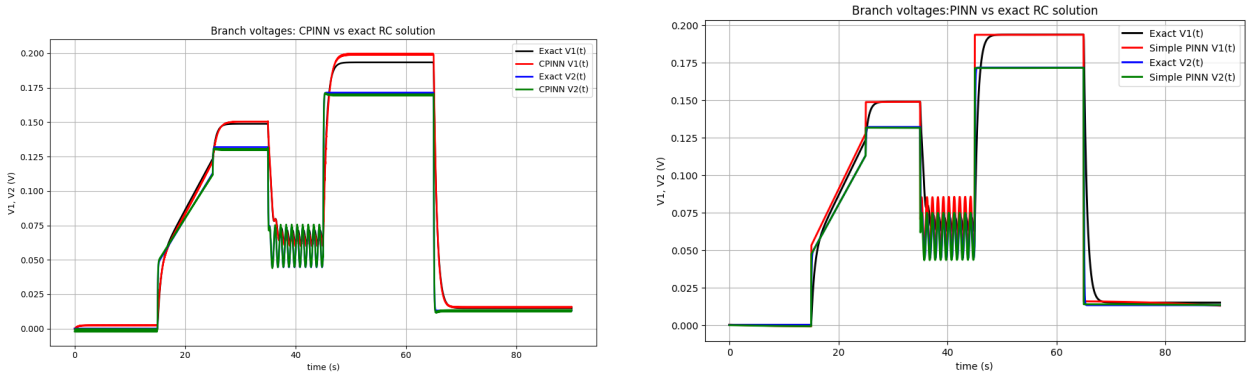


Figure 8: (a) CPINN prediction of the RC-branch voltages $V_1(t)$ and $V_2(t)$ compared with the exact equivalent-circuit solution. The CPINN accurately reproduces both transient behavior and oscillatory dynamics of the branches. (b) PINN prediction of the same branch voltages under identical test conditions. While the overall trends are captured, larger deviations appear during fast transients and oscillatory intervals compared to CPINN.

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 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

Table 9: Quantitative comparison of PINN and CPINN prediction performance on the generalization test dataset shown in the Figure 7

Method	Output	MSE (V^2)	RMSE (V)	MAE (V)
PINN	Total Voltage V	3.08×10^{-4}	1.76×10^{-2}	6.67×10^{-3}
	Branch Voltage V_1	2.16×10^{-4}	1.47×10^{-2}	4.81×10^{-3}
	Branch Voltage V_2	1.83×10^{-5}	4.28×10^{-3}	9.71×10^{-4}
CPINN	Total Voltage V	1.62×10^{-5}	4.03×10^{-3}	2.05×10^{-3}
	Branch Voltage V_1	9.52×10^{-6}	3.08×10^{-3}	2.45×10^{-3}
	Branch Voltage V_2	1.86×10^{-6}	1.36×10^{-3}	1.09×10^{-3}

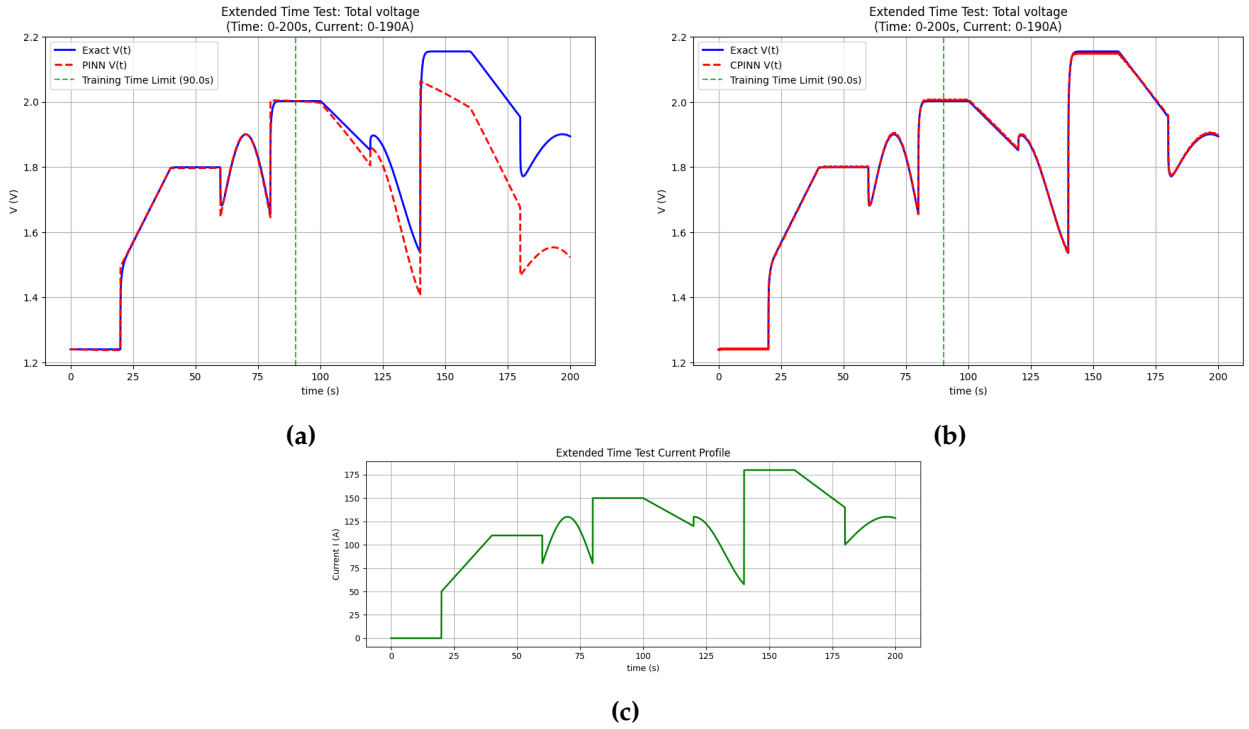
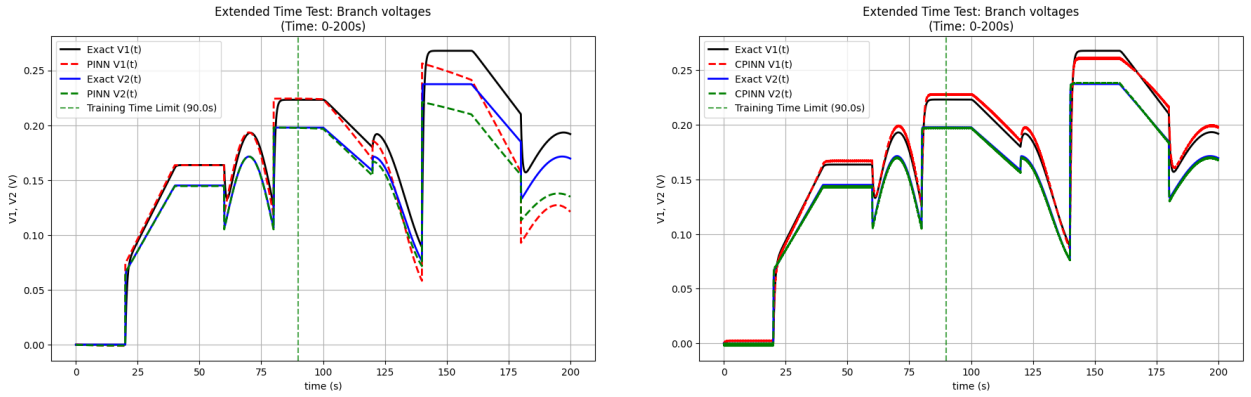


Figure 9: (a) PINN voltage prediction compared with the exact equivalent-circuit solution, where increasing deviation is observed outside the training time region.(b) CPINN voltage prediction under the same conditions, showing improved stability and closer agreement with the reference solution over the 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.

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 observed improvements can be attributed to the conditional structure of CPINN, which enables the model to learn a parameterized solution manifold rather than a single deterministic trajectory. This formulation improves the performance of the model in capturing variations in initial conditions and operating regimes and this improvement robust the framework against distribution



(a) **(b)**

Figure 10: (a) PINN prediction of branch voltages evaluated over the extended interval 0–200 s after training within 0–90 s, 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).

Method	**Output**	**MSE (V^2)**	**RMSE (V)**	**MAE (V)**
PINN	Total Voltage V	1.85×10^{-2}	1.36×10^{-1}	8.05×10^{-2}
	Branch Voltage V_1	7.23×10^{-4}	2.69×10^{-2}	1.65×10^{-2}
	Branch Voltage V_2	2.26×10^{-4}	1.50×10^{-2}	9.10×10^{-3}
CPINN	Total Voltage V	1.47×10^{-5}	3.84×10^{-3}	2.86×10^{-3}
	Branch Voltage V_1	1.92×10^{-5}	4.38×10^{-3}	3.94×10^{-3}
	Branch Voltage V_2	2.49×10^{-6}	1.58×10^{-3}	1.39×10^{-3}

shifts in time-domain simulations. 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 frameworks 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.

Acknowledgement: Not applicable

Funding Statement: This work has been supported by

- DECODER project (PID2024-158394OB-C22), funded by the Spanish Ministry of Science, Innovation and Universities, the Spanish National Research Agency and the European Regional Development Fund (MICIU/AEI/10.13039/501100011033/FEDER UE).
- AVANTE project (PDC2025-165204-C22), funded by the Spanish Ministry of Science, Innovation and Universities and the Spanish National Research Agency (MICIU/AEI/10.13039/501100011033).
- Siemens Energy AI Chair: Energy Sustainability for a Decarbonized Society 5.0 (TSI-100930-2023-5), funded by the Secretary of State for Digitalization and Artificial Intelligence through the ENIA 2022 Chairs call, and co-funded by the European Union-Next Generation EU.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Milad Sabamehr, Carles Batlle Arnau and Maria Serra Prat; methodology, Milad Sabamehr; software, Milad Sabamehr; validation, Milad Sabamehr, Carles Batlle Arnau and Maria Serra Prat; formal analysis, Milad Sabamehr; investigation, Milad Sabamehr; resources, Carles Batlle Arnau and Maria Serra Prat; data curation, Milad Sabamehr; writing original draft preparation, Milad Sabamehr; writing review and editing, Milad Sabamehr, Carles Batlle Arnau and Maria Serra Prat; visualization, Milad Sabamehr; supervision, Carles Batlle Arnau and Maria Serra Prat; project administration, Carles Batlle Arnau and Maria Serra Prat; funding acquisition, Carles Batlle Arnau and Maria Serra Prat. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets supporting this article are not currently available due to ongoing analyses. Requests for access may be directed to the corresponding author.

Ethics Approval: Not applicable

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

PEM	Proton Exchange Membrane
ECM	Equivalent Circuit Model
RC	Resistor–Capacitor
ODE	Ordinary Differential Equation
NN	Neural Network
ANN	Artificial Neural Network
PINN	Physics-Informed Neural Network
CPINN	Conditional Physics-Informed Neural Network
PINNC	Physics-Informed Neural Network for Control
SVR	Support Vector Regressor
XGB	Extreme Gradient Boosting
MPC	Model Predictive Control
IC	Initial Condition
AD	Automatic Differentiation
LBFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error

References

1. Gosu DL, Durvasula A, Annamalai J. Modelling of PEM electrolyzer dynamics for green hydrogen production. In: Hydrogen Energy System Modelling and Simulation. Elsevier; 2024. [CrossRef]
2. Hidouri D, Marouani R, Cherif A. Modeling and simulation of a renewable energy PV/PEM with green hydrogen storage. Eng Technol Appl Sci Res. 2024;14(3):1–10. [CrossRef]
3. Sayed-Ahmed H, Abdelsalam A, Mohamed A. Dynamic operation of proton exchange membrane water electrolyzers: A critical review. Renew Sustain Energy Rev. 2024;189:113930. [CrossRef]
4. Asiaban S, Houshfar E, Rosen MA. Development of a dynamic mathematical model of PEM electrolyser for flexibility studies. Results Eng. 2024;21:101810. [CrossRef]
5. Ratib MK, Azzouz A, Roux L, Agbossou K. Electrical circuit modeling of proton exchange membrane electrolyzers: An exhaustive review. Int J Hydrogen Energy. 2024;49(2):1234–1255. [CrossRef]
6. Guilbert D, Phattanasak M, Martin JP. Dynamic emulation of a PEM electrolyzer by time-dependent electrical equivalent model. Energies. 2019;12(12):2308. [CrossRef]
7. Zhang X, Chan SH. Data-driven modeling of electrochemical energy systems: A review. Appl Energy. 2022;305:117859. [CrossRef]
8. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019;378:686–707. [CrossRef]
9. Cuomo S, Di Cola VS, Giampaolo F, Rozza G, Raissi M. Scientific machine learning through physics-informed neural networks: Where we are and what's next. J Sci Comput. 2022;92:88. [CrossRef]
10. Zerrougui I, Li Z, Hissel D. Physics-informed neural network for modeling and predicting temperature fluctuations in proton exchange membrane electrolysis. Energy AI. 2025;20:100474. [CrossRef]
11. Ko T, Kim D, Park J, Lee SH. Physics-informed neural network for long-term prognostics of proton exchange membrane fuel cells. Appl Energy. 2025;382:125318. [CrossRef]
12. Karniadakis G.E., Kevrekidis I.G., Lu L., Perdikaris P., Wang S., Yang L. Physics-informed machine learning. *Nature Reviews Physics*. 2021;3:422–440. [CrossRef]
13. Kim J, Oh C, Kim K, Lee J, Kim T. Parameter identification of electrical equivalent circuits including mass transfer parameters for the selection of the operating frequencies of pulsed PEM water electrolysis. *Energies*. 2022;15(24):9303. [CrossRef]
14. Dang J, Yang F, Li Y, Deng X, Ouyang M. Transient behaviors and mathematical model of proton exchange membrane electrolyzer. *Journal of Power Sources*. 2022;542:231757. [CrossRef]
15. Raman KA, Hammacher L, Kungl H, Karl A, Jodat E, Eichel R, Karyofylli V. Data-driven surrogate modeling for performance prediction and sensitivity analysis of transport properties in proton exchange membrane water electrolyzers. Appl Energy. 2025;386:125529. [CrossRef]
16. Hurt SE, Toliyat HA. 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)*; 2025; Houston, TX, USA. pp. 1081–1086. [CrossRef]
17. Misyris GS, Venzke A, Chatzivasileiadis S. Physics-informed neural networks for power systems. In: Proc IEEE Power & Energy Society General Meeting (PESGM). Montreal, QC, Canada; 2020. p. 1–5. [CrossRef]
18. Misyris GS, Venzke A, Chatzivasileiadis S. PINNSim: A simulator for power system dynamics based on physics-informed neural networks. IEEE Open Access J Power Energy. 2023. [CrossRef]
19. Misyris GS, Venzke A, Chatzivasileiadis S. PINNSim: A simulator for power system dynamics based on physics-informed neural networks. arXiv preprint arXiv:2512.02712. 2025. [CrossRef]
20. Zhang Y, Wang S, Perdikaris P. Multi-fidelity physics-informed neural networks for time-dependent problems. arXiv preprint arXiv:2411.10483. 2024. [CrossRef]
21. Hurt SE, Toliyat HA. Physics-informed neural networks for electrical circuit analysis: Applications in dielectric material modeling. IEEE Trans Ind Appl. 2024. [CrossRef]

22. Dey S, Mallik A. Physics-informed neural network–estimated circuit parameter adaptive modulation of DAB. IEEE Trans Power Electron. 2025;40(10):14821–14841. [CrossRef]
23. Arnold F, King R. State–space modeling for control based on physics-informed neural networks. Eng Appl Artif Intell. 2021;101:104195. [CrossRef]
24. Gokhale G, Claessens B, Develder C. Physics-informed neural networks for control-oriented thermal modeling of buildings. Appl Energy. 2022;314:118852. [CrossRef]
25. Antonelo EA, Camponogara E, Seman LO, Jordanou JP, Souza ERd, Hübner JF. Physics-informed neural nets for control of dynamical systems. Neurocomputing. 2024;579:127419. [CrossRef]
26. Mao X, Tian Y, Yang A, Zhang G. Identification of equivalent circuit parameters for proton exchange membrane (PEM) electrolyzer engineering models. IEEE Access. 2024;12:15509–15524. [CrossRef]
27. Kovacs A, Exl L, Kornell A, Fischbacher J, Hovorka M, Gusenbauer M, Schrefl T. Conditional physics-informed neural networks. Commun Nonlinear Sci Numer Simul. 2022;104:106041. [CrossRef]
28. Manoj SO, Ananth JP, Rohini M, Dhanka B, Pooranam N, Arumugam SR. FWS-DL: forecasting wind speed based on deep learning algorithms. In: Artificial Intelligence for Renewable Energy Systems. Elsevier; 2022. p. 353–374. [CrossRef]
29. Hodson TO. Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. Geosci Model Dev Discuss. 2022:1–10. [CrossRef]
30. Willmott CJ, Matsuura K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Clim Res. 2005;30:79–82. [CrossRef]
31. Robeson SM, Willmott CJ. Decomposition of the mean absolute error (MAE) into systematic and unsystematic components. PLoS One. 2023;18(2):e0279774. [CrossRef]
32. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014. [CrossRef]
33. Andrew G, Gao J. Scalable training of ℓ_1-regularized log-linear models. In: Proc 24th Int Conf Mach Learn (ICML). 2007. p. 33–40. [CrossRef]
34. Cai S, Wang Z, Wang S, Perdikaris P, Karniadakis GE. Physics-informed neural networks for heat transfer problems. J Heat Transfer. 2021;143(6). [CrossRef]
35. Sabamehr M, Batlle C, Serra-Prat M. PINN for parameter identification of equivalent circuit models of PEM electrolyzer. Accepted abstract, European Hydrogen Energy Conference (EHEC); 2026. [Link]