Highlights

Self-loop physics-informed neural network for model predictive control of PEM electrolyzers

Islam Zerrougui, Zhongliang Li, Daniel Hissel

- Self-loop PINN enables fast one-step predictions for PEM electrolyzer control.
- Physics-informed training ensures accurate, consistent thermal modeling.
- PINN surrogate runs faster than first-principles ODE solver.
- NMPC with PINN ensures robust temperature tracking under power fluctuations.

Self-loop physics-informed neural network for model predictive control of PEM electrolyzers

Islam Zerrougui[a], Zhongliang Li[a,*] and Daniel Hissel[a,b]

[a]Université Marie et Louis Pasteur, UTBM, CNRS, Institut FEMTO-ST, FCLAB, F-90000 Belfort, France
[b]Institut Universitaire de France (IUF), Paris, France

ARTICLE INFO

Keywords:
PINN,
PEM electrolyzer,
Thermal management,
Self-loop PINN,
Real-time control

ABSTRACT

Effective thermal management of proton exchange membrane (PEM) electrolyzers is critical for ensuring both high efficiency and long-term durability. In this work, a physics-informed neural network (PINN) surrogate model is developed to accurately and efficiently predict the thermal behavior of a PEM electrolyzer, meeting real-time control requirements. The proposed surrogate employs a self-loop architecture that directly incorporates control inputs-electrical current and coolant flow rates-enabling recursive one-step-ahead predictions of temperature evolution under dynamic operating conditions. Physical knowledge derived from energy-balance equations is embedded into the training loss, enforcing consistency with conservation laws while reducing the need for extensive datasets and enhancing generalization. The trained PINN closely reproduces the results of a high-fidelity first-principles thermal model, maintaining negligible error across diverse transient scenarios. Furthermore, when integrated within a nonlinear model predictive control (NMPC) framework, the PINN surrogate successfully regulates the electrolyzer's temperature at desired setpoints, even under rapidly fluctuating power inputs. Overall, the proposed approach demonstrates a computationally efficient and physics-consistent pathway for real-time modeling and control of complex energy systems.

1. Introduction

In recent years, hydrogen has emerged as a pivotal energy carrier in the global transition toward clean and sustainable power. It offers a flexible means to store energy and decarbonize different sectors, such as heavy industry and long-haul transport. However, hydrogen is not yet ready for industrial-level use, and to increase its spread, a variety of production methods have been developed. Today, most hydrogen is produced from fossil fuels, a process that generates significant CO_2 emissions [1]. By contrast, hydrogen produced via water electrolyzer uses only water and electricity, is carbon-free, and its adoption is expected to grow rapidly in the years ahead [2]. Water electrolyzer is thus a cornerstone technology for carbon-free hydrogen. Among the various electrolyzer types, polymer exchange membrane (PEM) electrolyzers are particularly promising for integration with intermittent renewable energy sources: PEM systems can ramp up and down quickly, operate at high current densities, and maintain a wide operating range, making them well-suited to absorb fluctuating power inputs dynamically [3].

Thermal management is a critical aspect of PEM water electrolyzers, which use electrical energy to split water into hydrogen and oxygen. During operation, significant heat is generated due to inefficiencies and overpotential losses [4]. In fact, PEM electrolyzers typically operate in an exothermic regime where the heat produced by overpotentials exceeds the thermal energy requirement for water splitting [1]. If not properly controlled, this heat can lead to temperature rise, affecting efficiency, accelerating component degradation, and posing safety risks. Accurate dynamic thermal models of the electrolyzer are therefore essential for designing control systems (such as cooling control or load management) to maintain safe and optimal operating temperatures [5].

Developing an accurate model of the electrolyzer's thermal behavior is challenging. First-principles modeling based on energy balances and heat transfer can be performed, but model fidelity depends on accurately knowing parameters (heat capacities, heat transfer coefficients, etc.), which may vary with operating conditions [6]. Simplified lumped-parameter models can capture dominant dynamics but might deviate from reality if some physics are unmodeled or parameters are uncertain [7]. Purely data-driven black-box models, on the other hand, require extensive operational data for training and may struggle to extrapolate beyond the conditions seen in training. Thus, there is a need for hybrid

**Corresponding author: Islam ZERROUGUI Email: islam.zerrougui@femto-st.fr
ORCID(s):

modeling approaches that combine the interpretability of physics-based models with the adaptability of data-driven learning [8].

Physics-Informed Neural Networks (PINNs) have emerged as a promising approach to combine governing physics equations with machine learning. PINNs embed the known physical laws (expressed as differential equations or residuals) into the training objective of a neural network [9]. This technique has been successfully used for solving forward and inverse problems in physics, reducing the need for large datasets by guiding the model with prior knowledge. PINNs offer advantages such as incorporating partial differential equation constraints and yielding solutions that inherently satisfy physical properties and available data. Moreover, once trained, a PINN can produce solutions very rapidly by direct evaluation of the neural network, avoiding iterative numerical integration. A significant benefit of PINNs is the high speed and low computational cost of inference, which makes them attractive for real-time state estimation and control. Jiang et al. [10] propose a multi-fidelity physics-informed convolutional neural network for battery-pack thermal management, in which a simplified 2D heat-transfer model was used to embed physics knowledge into a CNN backbone, and a small projection head was trained with limited high-fidelity data to refine the output. They show 15 % improvement over purely data-driven surrogates, highlighting the value of hybrid modeling for thermal systems.Liang et al. [11] developed a PINN framework for optimal control of chiller plants, integrating two categories of prior knowledge: structure-type (physical equations embedded in network architecture) and trend-type (monotonic relationships formulated as physics-based loss functions). These models-S-PINN, T-PINN, and their hybrid TS-PINN-significantly improved extrapolation accuracy over gray-box and black-box models. Field experiments showed a 23.2 % improvement in energy efficiency, demonstrating that incorporating both structural physics and empirical trends enhances generalization and robustness in control-oriented energy modeling. In our last work [12], the PINN ability to estimate different parameters associated with the equation based on low-quality data was demonstrated. A parametric PINN was also developed for estimating the temperature fluctuation in the PEM electrolyzer system, and its generalization ability was investigated. Even though the PINN was able to sense the performance dynamics beyond the test range, its capability for real time and variable force term is not yet satisfying. This limitation came from the fact that PINN are typically formulated on fixed domains with prescribed initial conditions and time spans, and their original formulation cannot inherently accommodate time-dependent control inputs [13]. As a result, a PINN trained for one specific scenario often fails to generalize to new control trajectories or to maintain accuracy over extended prediction horizons, limiting its effectiveness in receding-horizon settings like model predictive control (MPC).

In recent literature, several extensions have been proposed to address these issues by making PINNs more adaptable to control tasks. For instance, Mowlavi and Nabi [14] extended the PINN framework to solve PDE-constrained optimal control problems by embedding the control variable and cost objective directly into the network's loss, thus finding optimal controls alongside the state solution. The authors highlight the issue of balancing the different objectives when training the network. Similarly, Barry-Straume et al. [15] introduced Control PINNs, which incorporate the necessary optimality conditions into the architecture and loss function, enabling simultaneous learning of system states and optimal control signals in a one-stage procedure. The two precedent works handle the inverse problem in which the variable control needs to be determined. Other approaches modify the network structure to handle time-varying inputs and long time horizons. Antonelo et al. [16] proposed a dedicated PINN for control (named PINC) architecture that explicitly includes control inputs and permits simulation over variable, extended time intervals, making it far more flexible than classical PINNs. Likewise, Nicodemus et al. [13] augmented PINNs with control actions and initial conditions as additional inputs, allowing a single network to approximate dynamics under different control sequences. This PINN surrogate was deployed within an NMPC for a multi-link manipulator, where the use of automatic differentiation to compute sensitivities yielded efficient gradient-based control updates. Li and Li [17], implemented a PINN-based NMPC for an automated guided vehicle, showing that a physics-informed network can replace an explicit vehicle model and achieve fast, precise trajectory tracking in complex environments. Zheng et al. [18] developed a physics-informed recurrent neural network (PIRNN) for controlled systems, combining data-driven learning with physics-based regularization; their PIRNN-based MPC scheme demonstrated improved noise robustness and generalization performance compared to purely data-driven models. Wu et al. [19] extended the PIRNN framework to a batch crystallization process, integrating detailed population balance equations and physical constraints (e.g., non-negativity and mass conservation) directly into the learning architecture. Their physics-informed machine learning-based MPC achieved accurate prediction of crystal growth dynamics and efficient control of temperature trajectories. Xu et al. [20] developed a physics-informed stochastic configuration network (PISCN) integrated with multi-objective MPC, embedding physical laws into the model to enhance accuracy and generalization. Their framework efficiently balanced multiple objectives such as energy consumption and product quality. In the realm of soft robotics, Habich

et al. [21] reported that a PINN surrogate model for an articulated soft robot achieved high generalization across different operating conditions (outperforming a comparable RNN) and ran over two orders of magnitude faster than a first-principles simulator, thereby enabling real-time NMPC with minimal loss of accuracy.

Collectively, these advancements show that by incorporating physics knowledge and new architectures, PINNs can be made far more amenable to control problems, mitigating many of their traditional limitations.

This work addresses the current gap in developing physics-informed, control-oriented surrogate models for real-time thermal management of PEM electrolyzers. Although conventional PINNs have shown strong potential for modeling nonlinear thermal dynamics, their standard formulations are typically limited to fixed initial conditions and static input profiles, preventing accurate long-horizon predictions under time-varying control inputs. Moreover, the use of PINNs within closed-loop control frameworks-particularly for electrolyzer thermal regulation-has not been thoroughly explored, and comparative evaluations against established controllers such as proportional-integral (PI) schemes are largely absent from the literature.

To overcome these limitations, we propose a self-loop PINN architecture explicitly designed for real-time thermal prediction and control of PEM electrolyzers. The approach begins with a reduced-order, lumped-parameter thermal model that captures the dominant energy-balance dynamics governing stack and coolant temperatures. This physical model is embedded within the neural network's training loss to enforce thermodynamic consistency while enabling recursive one-step-ahead prediction of the system's temperature evolution under dynamically changing inputs.

Comprehensive simulations demonstrate that the proposed self-loop PINN achieves high accuracy and strong generalization, closely matching the predictions of a high-fidelity Runge-Kutta (RK4) solver while operating several times faster, making it well-suited for embedded and real-time applications. When integrated into a NMPC framework, the surrogate enables precise temperature regulation across steady, fluctuating, and time-varying load scenarios. Comparative studies with a classical dual PI controller further confirm the method's superior transient performance, faster convergence, and significant computational efficiency, underscoring its practical advantages for real-time electrolyzer control.

The remainder of the paper is organized as follows. Section 2 reviews the fundamentals of PINN for dynamic systems and control. Section 3 introduces the proposed self-loop PINN formulation for one-step-ahead modeling. Section 4 presents the three-state thermal model of the PEM electrolyzer together with its governing equations. Section 5 describes the architecture of the PINN surrogate model, the normalization procedure, and the training algorithms. Section 6 details the MPC-based PINC control framework, while Section 7 discusses the design and tuning of the dual PI controllers. Section 8 reports the obtained results, and Section 9 provides a detailed discussion. Finally, Section 10 concludes the paper.

2. Background: PINNs for Dynamic Systems and Control

PINN are a class of deep learning models that incorporate physical laws into the training of neural networks [9]. In a typical PINN for dynamic systems, the network is set up to approximate the solution of the system's differential equations. For example, consider a system described by ODEs

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

with f is function representing the system dynamics, $x(t)$ is system state and $u(t)$ is system input. A conventional PINN would introduce a neural network $\mathcal{F}_\theta(t)$ that takes the time t (and possibly initial conditions or spatial coordinates, in the case of PDEs) as input and outputs an estimate of the state $x(t)$. The network's training objective includes a physics loss term penalizing any violation of the ODE. In particular, the physics loss can be defined as [22] :

$$L_{\text{phys}} = \| \frac{d}{dt}\mathcal{F}_\theta(t) - f(\mathcal{F}_\theta(t), u(t)) \|^2 \tag{2}$$

evaluated on a set of collocation points in the time domain. Additionally, if some measured data or initial condition is available (x_{data}), a data loss term might be included for those points as follows

$$L_{\text{data}} = \| \mathcal{F}_\theta(t) - x_{\text{data}}(t) \|^2 \tag{3}$$

by minimizing the combined loss, the PINN is trained to produce a function $\mathcal{F}_\theta(t)$ that not only fits observed data but also satisfies the ODE to a high degree. This results in a model that is consistent with known physics and often generalizes better than a purely data-driven model, especially when data are sparse [23].

While PINNs have demonstrated success in various domains (e.g., fluid dynamics, structural mechanics), their direct application to control-oriented modeling faces two primary limitations. First, the standard PINN formulation does not explicitly account for time-dependent control inputs. The network sees time t as an input coordinate, but if the system's behavior changes with a control input $u(t)$, one needs to somehow encode $u(t)$ into the network. One could treat u as an additional input variable to the network; however, if $u(t)$ is a piecewise constant or arbitrary signal, the PINN would have to be trained on a specific input trajectory or a family of trajectories, which complicates training. In their original form, PINNs typically cannot simply take an arbitrary control sequence as input and produce the corresponding state trajectory without retraining for that specific sequence. The second limitation is that conventional PINNs are trained on a fixed time interval or time window. For instance, a PINN might be trained to predict $x(t)$ for $t \in [0, T]$. If one wishes to simulate the system beyond T or with a different time horizon, the PINN's performance may degrade or it may not be directly applicable, since it was not trained on that interval. Essentially, the PINN solution is tied to the time domain used during training. In control applications, it is often necessary to have a model that can be queried over shifting horizons (e.g., the receding horizon in MPC) and extended time periods, without being constrained to predefined intervals.

These limitations motivate the need for more flexible architectures that retain the physics-based advantages of PINNs while accommodating variable control inputs and long or shifting prediction horizons. In the next section, a modified architecture-termed a self-loop or one-step-ahead PINN-is introduced to address these shortcomings and is specifically tailored for the thermal control of PEM electrolyzers.

3. Self-Loop PINN Formulation :

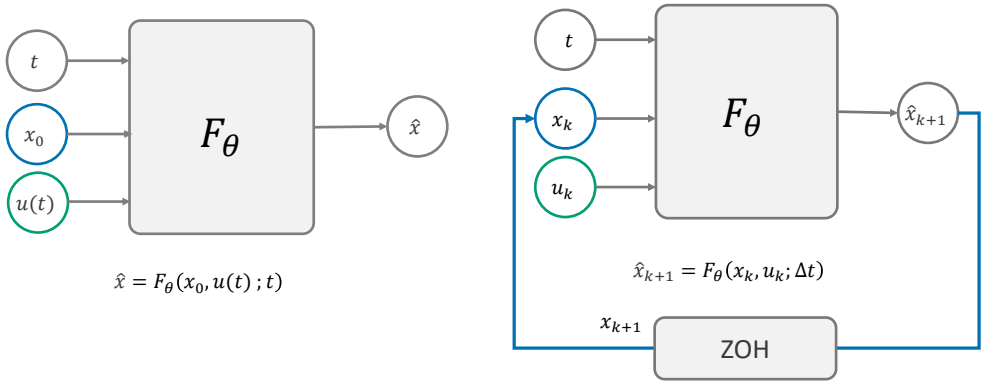
Let $\mathbf{x}(t) \in \mathbb{R}^n$ denote the state vector of the system, and let $u(t)$ be the control input. The system is considered under a discrete-time sampling scheme with a time step Δt (for example, $\Delta t = 1$ s or any appropriate interval for the system dynamics). The system's dynamics are given by a set of ODEs:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), u(t)), \tag{4}$$

where $f(\cdot)$ represents the known (or partially known) physics of the system.

3.1. Self-Loop PINN architecture

PINC is a special type of PINN that is used to predict on short time intervals. At each interval, it takes two additional inputs: the initial state $\mathbf{x}_k$ and the held control u_k. This enables it to adapt to different operating points, forcing terms, and initial conditions. PINC works in a closed-loop framework, in which after each time interval, the predicted state $\hat{\mathbf{x}}_{k+1}$ is fed back as a new initial state for the next time interval. This enables PINC to be deployed over many periods while remaining tied to the physics; this is why it is also referred to as a 'self-loop'. The conventional PINN and PINC architectures can be seen in the figure below.



(a) PINN (coordinate → state) (b) One-step PINN / PINC (self-loop)

Figure 1: Comparison of a conventional PINN vs. the proposed self-loop PINN (PINC) approach.

3.2. Training the self-loop PINN

In order to train the one-step predictive model, let us consider the neural network, represented by the function $\mathcal{F}_\theta$, with θ are network parameters. Its output is represented by the following equation:

$$\hat{\mathbf{x}}_{k+1} = \mathcal{F}_\theta(\mathbf{x}_k, u_k, \tau) \tag{5}$$

in which $\hat{\mathbf{x}}_{k+1}$ is the neural network output. $(\mathbf{x}_k, u_k, \tau)$ are the network input. A composite loss function is minimized to penalize prediction errors and enforce physical consistency over the interval $[t_k, t_{k+1}]$.

$$\begin{aligned}\mathcal{L}(\theta) = \frac{1}{M}\sum_{i=1}^{M}\Bigg[\frac{\lambda_{\text{phys}}}{N_c}\sum_{j=1}^{N_c}\Big\|\frac{1}{\Delta t}\,\partial_\tau \mathcal{F}_\theta(\mathbf{x}_k^{(i)}, u_k^{(i)}, \tau_{ij}) - f\big(\mathcal{F}_\theta(\mathbf{x}_k^{(i)}, u_k^{(i)}, \tau_{ij}),\, u_k^{(i)}\big)\Big\|_2^2 \\ + \lambda_{\text{IC}}\Big\|\mathcal{F}_\theta(\mathbf{x}_k^{(i)}, u_k^{(i)}, 0) - \mathbf{x}_k^{(i)}\Big\|_2^2\Bigg],\end{aligned} \tag{6}$$

Here $\tau \in (0, 1)$ are dimensionless interior collocation points that span the normalized interval $[0, 1]$, corresponding to the physical interval $[t_k, , t_{k+1}]$ of length Δt, ∂_τ is obtained by automatic differentiation. The control input $u(t)$ is held constant $(u(t) = u_k)$ within each interval. $\lambda_{\text{phys}}, \lambda_{\text{IC}}$ are weights that balance the physics residual term and initial consistency. In equation 6, two loss terms are defined. The physics residual term enforces the system dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, u_k)$ at the collocation points by minimizing the ODE residual between the network's derivative $\partial_\tau \mathcal{F}_\theta$ and the known physics $f(\cdot)$. This ensures physical consistency of the learned model. The initial condition term enforces $\mathcal{F}_\theta(\mathbf{x}_k, u_k, 0) = \mathbf{x}_k$, anchoring the predicted trajectory to the correct starting state. This constraint allows training over varying initial conditions, unlike conventional PINN formulations.

During training, for each training sample $(\mathbf{x}_k, u_k \to \mathbf{x}_{k+1})$ the following algorithm is performed:

Algorithm 1: One-step PINN training on the interval $[t_k, t_{k+1}]$

Input: ODE right-hand side f;
Neural network $\mathcal{F}_\theta(\mathbf{x},\, u,\, \tau)$ for $\tau \in [0, 1]$;
Initial state $\mathbf{x}_k$ at t_k; control input u_k;
Step length Δt; number of collocation points N_c;
Loss weights λ_{IC}, λ_{phys}; learning rate η;
Maximum training iterations N_{iter}.
Output: Trained network parameters θ.

Initialise network parameters θ (e.g., randomly)
for *iteration* $= 1$ **to** N_{iter} **do**
  $\mathcal{L} \leftarrow 0$
  A. Collocation sampling
  Draw $\tau_1, \ldots, \tau_{N_c} \sim \mathcal{U}(0, 1)$
  Set $\mathcal{C} \leftarrow \{\tau_1, \ldots, \tau_{N_c}\}$ `// random interior points`
  B. Initial condition loss
  $\hat{\mathbf{x}}_0 \leftarrow \mathcal{F}_\theta(\mathbf{x}_k,\, u_k,\, 0)$
  $\mathcal{L}$ += $\lambda_{\text{IC}} \left\|\hat{\mathbf{x}}_0 - \mathbf{x}_k\right\|_2^2$ `// enforce initial state at` t_k
  C. Physics residuals
  foreach $\tau \in \mathcal{C}$ **do**
    $\hat{\mathbf{x}} \leftarrow \mathcal{F}_\theta(\mathbf{x}_k,\, u_k,\, \tau)$
    $\dot{\hat{\mathbf{x}}} \leftarrow \frac{1}{\Delta t}\, \partial_\tau \mathcal{F}_\theta(\mathbf{x}_k,\, u_k,\, \tau)$ `// time derivative via autodiff`
    $\mathbf{r} \leftarrow \dot{\hat{\mathbf{x}}} - f(\hat{\mathbf{x}},\, u_k)$
    $\mathcal{L}$ += $\lambda_{\text{phys}} \left\|\mathbf{r}\right\|_2^2$ `// accumulate physics loss`
  D. Parameter update
  Compute $\nabla_\theta \mathcal{L}$ `// gradient of loss w.r.t.` θ `(automatic backprop)`
  $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}$ `// gradient descent step to minimize` $\mathcal{L}$
return θ `// optimized parameters after training`

By minimizing the total loss L, the network learns to produce correct one-step predictions for different inputs and different time domains.

3.3. Employing the self-loop PINN

The core idea of PINC lies in how it is used. Once the PINC model is trained, using it for simulation or control is straightforward. Given an initial state $\mathbf{x}_0$ and a sequence of future control inputs $u_0, u_1, u_2, \ldots$, the network can be rolled out step by step to predict the system trajectory.

This self-looping rollout is depicted schematically in the diagram shown in Figure 2. Within each time window $[t_k, t_{k+1}]$, the control input is held constant at u_k, and the PINC network generates the state trajectory for that interval (starting from the provided initial state $\mathbf{x}[k]$). The predicted end state $\hat{\mathbf{x}}[k+1]$ is then passed into the next window as the initial condition. For example given an initial state $\mathbf{x}_0$ and a sequence of inputs $u_0, u_1, u_2, \ldots$, the subsequent states are computed as $\mathbf{x}_1 = \mathcal{F}_\theta(\mathbf{x}_0, u_0)$, then $\mathbf{x}_2 = \mathcal{F}_\theta(\mathbf{x}_1, u_1)$, and so on. In this way, the model acts as a recursive simulator, advancing one step at a time under the given sequence of controls.

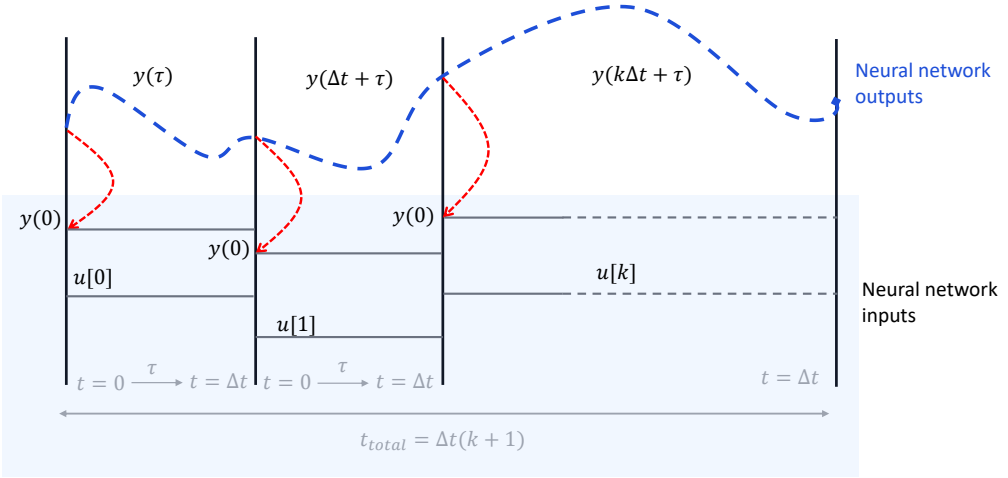


Figure 2: Self-loop rollout. Within each window, the control is held constant and the same initial state is used as a conditioning input. The state trajectory evolves inside the window. The predicted end state is fed as the next window's initial condition (red arrow) (reproduced from [16]).

Remark on Stability: A key challenge in self-loop (iterative) predictors is the potential accumulation of small local errors over time, which may lead to trajectory drift during long rollouts. In the proposed formulation, the PINN does not guarantee absolute stability in the theoretical sense. However, by embedding the governing physical laws into the loss function and enforcing the initial condition consistency at the beginning of each interval, the training process constrains the model's predictions to remain close to the physically valid manifold. This physics-based regularization acts as a stabilizing mechanism that helps to limit error growth and maintain coherent long-term behavior, even when the network is applied recursively. Thus, while the approach cannot fully eliminate numerical drift, it effectively mitigates its impact and promotes physically consistent predictions within practical horizons.

Having described the general PINC approach, we now introduce the specific thermal model of the PEM electrolyzer that provides the physics $f(\mathbf{x}, u)$ used to constrain the PINN.

4. Three-State Thermal Model of the PEM Electrolyzer

4.1. System description

PEM electrozer is a system that is used to split the water molecule into oxygen and hydrogen under the influence of an electrical load [24]. Thermal regulation of PEM electrolyzer is of high importance since it is related to its lifetime [25] and efficiency [5].

In the literature, different researchers propose control for the PEM electrolyzer system. Huang et al. [26] built a thermal model for the alkaline type of electrolyzer to regulate the outlet temperature using a proportional-integral (PI) controller. Qi et al. [27] propose a linearized state space representation for the entire balance-of-plant of the alkaline system, ending with a three-state space representation. In this work, they show that the MPC controller outperforms the feed-forward PID controller. Dang et al. [28] used a log-mean-temperature-difference (LMTD) method to couple the coolant system with the electrolyzer system; the method used a PID controller and was validated experimentally. Keller et al. [29] show that an adaptive PID controller combined with a feed-forward action can achieve better disturbance rejection compared to the PID controller.

In this work, we consider a PEM electrolyzer system with an active cooling circuit that can be seen in Figure 3. Three lumped states approximate the thermal dynamics: (1) the electrolyzer outlet temperature $T_{\text{out,ele}}$, (2) the heat exchanger outlet temperature which represent the hot side and the inlet temperature of the electrolyzer $T_{\text{in,ele}}$, and (3) the cooling circuit outlet temperature $T_{\text{out,c}}$. In this work, we are considering that the temperature of the separator outlet is quite stable.

These states (denoted x_1, x_2, x_3 respectively) are representative of the temperatures at key points in the system. The control inputs are the water flow rates on the *hot side* (u_1) and *cold side* (u_2) of the heat exchanger, which can be adjusted (e.g., via pump speeds) to regulate heat removal. An external disturbance enters the system in the form of the

electrolyzer's operating current $i(t)$ (from renewable energies), which generates heat internally via Joule heating and reaction inefficiencies.

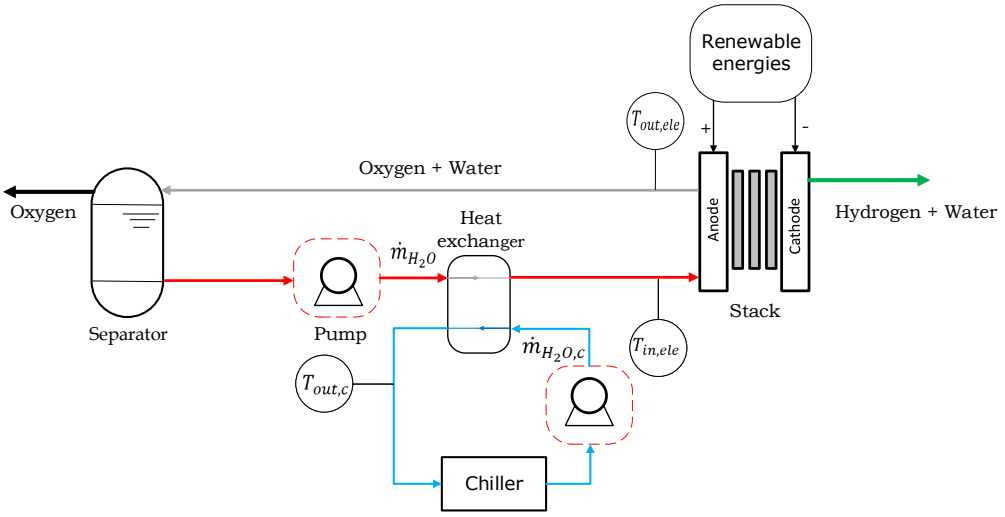


Figure 3: Temperature distribution in PEM electrolyzer system

4.2. Governing equations:

All model equations are derived from energy conservation in open systems. We use a lumped thermal capacitance approach, where in each state x_j has an associated thermal capacitance C_j that accounts for the combined mass and specific heat of the material at that node. The stack temperature is governed by an energy balance that weighs the internally generated heat against convective radiative losses and cooling by the feed water [5]:

$$C_{\text{th}}\dot{T}_{\text{out,ele}} = \dot{Q}_{\text{gen}} - \dot{Q}_{\text{loss}} - \dot{Q}_{\text{cool}} \tag{7}$$

The generated heat Q_{gen}, caused by irreversible electrochemical losses, which convert part of the electrical power into heat based on the following equation :

$$\dot{Q}_{\text{gen}} = n_{\text{cell}}\, I\left(V_{\text{cell}} - V_{th}\right), \tag{8}$$

n_{cell} is the number of cells in series. V_{th} is the thermodynamic voltage. V_{cell} is the single-cell voltage is estimated with the empirical correlation model proposed by Ulleberg [30]:

$$V_{cell} = V_{\text{rev}} + \frac{r_1 + r_2 T}{A}\, I + \left(s + s_1 T + s_2 T^2\right)\, \log\!\left(\frac{t_1 + t_2 T + t_3 T^2}{A}\, I + 1\right). \tag{9}$$

with $\{r_i, s_i, t_i\}$ experimental fitting constants. and A is the active surface area of the membrane. Heat losses Q_{loss} caused by convection and radiation effects can be lumped into an overall heat transfer coefficient h as expressed by the following equation

$$\dot{Q}_{\text{loss}} = h\,(T_{\text{out,ele}} - T_{\text{amb}}). \tag{10}$$

T_{amb} is the ambient temperature. Cooling losses due to feed water Q_{cool}, represent heat removal facilitated by the feed water during electrolyzer and can be described by:

$$\dot{Q}_{\text{cool}} = \dot{m}_{H_2O} c_{p,H_2O}\,(T_{\text{out,ele}} - T_{\text{in,ele}}). \tag{11}$$

with $\dot{m}_{H_2O}$ the mass flow rate. c_{p,H_2O} is the specific heat capacity of water.
The balance-of-plant heat exchanger connects two fluid loops: a hot loop associated with the electrolyzer outlet and a cold coolant loop. We model this nonlinear system using two lumped-capacitance energy balance equations as follows:

$$C_h \, \dot{T}_{\text{in,ele}} = \dot{m}_{H_2O} \, c_{ph} \, (T_{\text{sep}} - T_{\text{in,ele}}) - U \, A_{ex} \, (T_{\text{in,ele}} - T_{\text{out,c}}), \tag{12}$$

$$C_c \, \dot{T}_{\text{out,c}} = \dot{m}_{H_2O,c} \, c_{pc} \, (T_{\text{in,c}} - T_{\text{out,c}}) - U \, A_{ex} \, (T_{\text{out,c}} - T_{\text{in,ele}}). \tag{13}$$

where $U\,A_{ex}$ is the overall heat-transfer conductance, and $\dot{m}_{H_2Oc}$ is the cold water flow caused by the pump linked to the chiller and the heat exchanger. In this study, since the dynamic of the pump is fast compare to the dynamic of the temperature changes we neglect their dynamics, and for simplicity, we assume that the mass flows on the hot and cold sides are proportional to the control inputs u_1, u_2, which can physically represent two valves, specifically:

$$\dot{m}_{H_2O} = u_1 \upsilon_{H_2O} \tag{14}$$

$$\dot{m}_{H_2Oc} = u_2 \upsilon_{H_2O} \tag{15}$$

Summing up all the precedent equations, the final state space representation of the electrolyzer system can be formulated as follows:

$$\dot{x}(t) = \begin{pmatrix} -C_2 \, x_1 + C_3 \, x_1 \, u_1 - C_3 \, x_2 \, u_1 + D\big(i(t)\big) \\ C_5 \, x_3 - C_5 \, x_2 + C_4 \, x_1 \, u_1 - C_4 \, x_2 \, u_1 \\ C_6 \, x_2 - C_6 \, x_3 + C_8 \, u_2 - C_7 \, x_3 \, u_2 \end{pmatrix}, \tag{16}$$

All coefficients C_2 through C_8 are computed from known physical constants (cell geometry, heat capacities, flow areas), and their values are given in Table 1. For completeness, these coefficients are defined as follows:

$$C_2 = \frac{-h\,A}{C_{th}}, \quad C_3 = \frac{\upsilon_{H_2O} \, c_{p,H_2O}}{C_{th}}, \quad C_4 = \frac{\upsilon_{H_2O} \, c_{p,h}}{C_h}, \quad C_5 = \frac{U \, A_{\text{ex}}}{C_h}, \quad C_6 = \frac{U \, A_{\text{ex}}}{C_c}, \quad C_7 = \frac{\upsilon_{H_2O} \, c_{p,c}}{C_c},$$

$$C_8 = \frac{\upsilon_{H_2O} \, c_{p,c} \, T_{\text{in,c}}}{C_c},$$

The heat generation rate is represented by a nonlinear function $D(i)$ and is expressed as follows

$$D(i) = \frac{n_{cell} \, I \, \left(V_{cell} - V_{th}\right)}{C_{\text{th}}} + \frac{h \, A \, T_{\text{amb}}}{C_{\text{th}}},$$

All numeric constants can be found in the associated code.

Table 1: Model parameters extracted from [31]

Group	Symbol	Value	Units
Thermal capacities	C_{th}	1.20×10^5	J K^{-1}
	C_{h}	1.46×10^5	J K^{-1}
	C_c	2.30×10^4	J K^{-1}
Heat transfer & losses	$U\,A_{ex}$	140	W K^{-1}
	T_{amb}	300	K
	h	3.8	$\text{W m}^{-2}\,\text{K}^{-1}$
Electrical	N_{cell}	26	–
	V_{th}	1.48	V

4.3. Control strategy

Most existing works restrict attention to maintaining a single outlet or stack-average temperature. However, in practice, the temperature difference across the stack (i.e., $T_{out,ele} - T_{in,ele}$) can fluctuate a lot under heavy load, causing overheating and safety issues on the terminal of the cells. In literature, it was reported that excessive temperature difference not only accelerates membrane degradation but also increases mechanical stress and hydrogen crossover, compromising safety and durability [25]. Putting these remarks into consideration, our approach aims to optimize and regulate the two actuators (the two pumps) to optimize the temperature in the inlet and the outlet of the PEM electrolyzer, which can ensure better temperature distribution along the channels of the electrolyzer. In practice, the heat exchanger can fix the temperature inlet, while the water that passes through the electrolyzer can remove the heat from electrolyzer channels. Next, the implementation and training of the PINN using this model are discussed before presenting the results.

5. PINN Surrogate Architecture and Training

5.1. PINN Model and Training Setup

To create a fast surrogate for the above dynamics, we employ a PINN approach. The PINN is configured to learn the mapping from time (and control inputs) to the state variables, while being penalized for any violation of the physical laws. The surrogate model is implemented as a feedforward neural network comprising four hidden layers with 64 neurons per layer, employing hyperbolic tangent (tanh) activation functions. This architecture was selected based on empirical tuning and performance evaluation. The network outputs are the three state variables $\hat{x}(t) = [\hat{T}_{out,ele}, \hat{T}_{in,ele}, \hat{T}_{out,c}]^T$ approximating the true temperatures. In addition to the time t, the current $i(t)$ and the two flow inputs $u_1(t)$, $u_2(t)$ are provided to the network as input features; this allows the PINN to account for instantaneous control actions when predicting state derivatives or future states. All the data were normalized to the interval $[-1, 1]$ to align with the zero-centered range of the tanh activation. This keeps pre-activations near zero, reduces saturation, balances positive/negative gradients, and typically accelerates convergence [32]. Reflecting the expected physical range of temperature 300-350 K we have normalized it as follows:

$$x^{\text{norm}} = \frac{x - 325}{25} \tag{17}$$

The electrical current range is between $0 - 50$ is can be mapped analogously with

$$i^{\text{norm}} = \frac{i - 25}{25} \tag{18}$$

The two flow-rate commands are already generated in $[0, 1]$ and therefore require no additional scaling. The PINN is trained by minimizing a composite loss function that combines a physics-based residual term and an initial-condition consistency term. At a set of collocation time points τ_i sampled within the training interval, the physics loss enforces the governing ODE dynamics of the system as follows:

$$\mathcal{L}_{\text{phys}} = \frac{1}{M N_c} \sum_{i=1}^{M} \sum_{j=1}^{N_c} \left\| \frac{1}{\Delta t} \frac{\partial}{\partial \tau} \mathcal{F}_\theta(\mathbf{x}_k^{(i)}, u_k, \tau_{ij}) - f\left(\mathcal{F}_\theta(\mathbf{x}_k^{(i)}, u_k^{(i)}, \tau_{ij}), u_k^{(i)}, i^{(i)} \right) \right\|_2^2. \tag{19}$$

where M denotes the number of training samples, and N_c is the number of collocation points within each interval. The variable $\mathbf{x}_k^{(i)}$ represents the initial state of the i-th training sample at the beginning of the interval $[t_k, t_{k+1}]$, while $u_k^{(i)}$ and $i^{(i)}$ denote its corresponding control input and current. The variable $\tau_{ij} \in (0, \Delta t)$ represents the j-th collocation point within this interval, and Δt is the duration of the time step. The initial condition loss, ensures the network's output at $t = 0$ matches the known initial state $x(0)$.

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{i=1}^{N_{\text{IC}}} \left\| \mathcal{F}_\theta(\mathbf{x}_k^{(i)}, u_k^{(i)}, 0) - \mathbf{x}_k^{(i)} \right\|_2^2, \tag{20}$$

The total loss is

$$\mathcal{L} = \lambda_{phys} \mathcal{L}_{\text{phys}} + \lambda_{IC} \mathcal{L}_{\text{IC}} \tag{21}$$

where the coefficients λ_{phys} and λ_{IC} are adaptively updated to balance the two contributions such that:

$$\lambda_{\text{phys}}\,\mathcal{L}_{\text{phys}} = \lambda_{\text{IC}}\,\mathcal{L}_{\text{IC}}. \tag{22}$$

the update rule for these coefficients can then be expressed as:

$$\lambda_{\text{phys}} = \frac{\mathcal{L}_{\text{IC}}}{\mathcal{L}_{\text{phys}} + \mathcal{L}_{\text{IC}}}, \qquad \lambda_{\text{IC}} = \frac{\mathcal{L}_{\text{phys}}}{\mathcal{L}_{\text{phys}} + \mathcal{L}_{\text{IC}}}. \tag{23}$$

This adaptive reweighting ensures that both loss terms contribute comparably to the overall optimization, thereby stabilizing convergence and improving the model's ability to satisfy both data and physics constraints simultaneously. In practice, these coefficients are updated at each training iteration.

5.2. Training algorithms

We employed a two-stage optimization process to train the PINN. First, we used the Adam stochastic gradient descent optimizer to minimize L, with collocation points randomly sampled in each epoch from the time interval of interest (this introduces the observed fluctuations in the physics loss during training). Subsequently, we refined the weights with a second optimization stage using the limited-memory BFGS (L-BFGS) algorithm. Thanks to its quasi-Newton updates, L-BFGS offers stronger local convergence than Adam and is therefore well suited for fine-tuning once the stochastic phase has brought the parameters into the vicinity of a minimizer [33]. Figure 4 shows the training flows of the network.

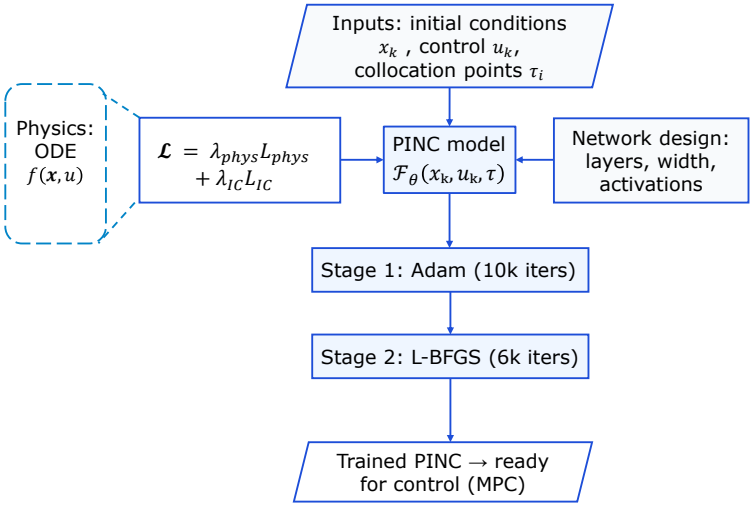
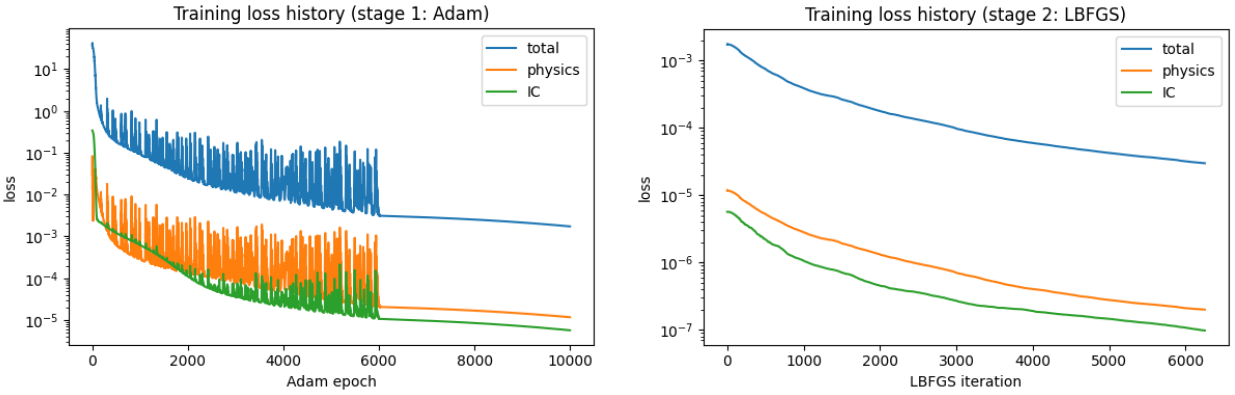


Figure 4: Training pipeline for the PINC model: inputs, physics-informed loss, two-stage optimization, and deployment for MPC.

Figure 5 illustrates the evolution of training losses over the two optimization stages (Adam on the left, L-BFGS on the right). The total loss (blue curve) decreases by several orders of magnitude throughout training, both the physics residual (orange) and initial condition error (green) shows a fluctuation when the learning rate was 10^{-4} in the first 6000, after that the learning rate decreased to 10^{-6} which shows that the learning was more stable, after that the L-BFGS fine tune the network parameters leading to a further decrease of the total loss to reach an lesser than 10^{-6}. This confirms that the PINN effectively learned to satisfy both the thermal model equations and the prescribed initial conditions. The PINN was trained on a Dell workstation equipped with a 12$^{\text{th}}$ generation Intel® Core™ i7-12700H CPU (base frequency 2.30 GHz); the full training run completed in 58 min.

6. MPC-Based PINC Control

MPC is an advanced control strategy that uses a model to predict the system's evolution over a finite horizon. At each sampling instant, given the current measured (or estimated) state, MPC solves an optimization problem to find a



(a) Stage 1

(b) Stage 2

Figure 5: Evolution of the total objective losses, physics residual (orange), and initial-condition loss (green) during the two-stage optimization (Adam and L-BFGS phases).

sequence of future control actions that minimizes a chosen cost while respecting the system dynamics and constraints. Only the first control move is applied; then the horizon is shifted forward and the problem is resolved with updated measurements. This receding-horizon procedure provides continuous, constraint-aware optimization.

In the proposed control scheme, the trained self-loop PINN model is embedded within a MPC framework to serve as a one-step-ahead differentiable predictor of the electrolyzer's thermal dynamics. In other words, the PINN acts as a surrogate model that predicts the outlet temperature in the next time step given the current state and candidate control inputs. Because the PINN is differentiable, the MPC can efficiently compute gradients of the predicted temperature with respect to the control inputs, enabling gradient-based optimization of the control actions. This approach effectively yields a PINC in a receding-horizon setting. The PINN surrogate offers fast prediction speeds compared to solving first-principles physics models, which makes real-time control feasible.

At each control interval, the MPC solves an optimization problem to determine the optimal control inputs u_1 and u_2 over a finite prediction horizon (receding horizon strategy). The objective is to track a target inlet temperature of 315 K while satisfying operational constraints and minimizing control effort. The cost function penalizes three key terms: (i) the deviation of outlet temperature $T_{\text{in,ele}}$ from the setpoint (315 K), (ii) any deviation of the outlet-inlet temperature difference $\Delta T = T_{\text{out,ele}} - T_{\text{in,ele}}$ from the desired 5 K, and (iii) the magnitude of the control actions u_1 and u_2. This can be expressed, for a single time-step prediction, by the following cost function:

$$J_{\text{cost}} = \alpha_0(x_1^{\text{pinn}} - x_1^{\text{target}})^2 + \alpha_1 (u_1)^2 + \alpha_2 (u_2)^2 + \beta \left((x_1^{\text{pinn}} - x_2^{\text{pinn}}) - 5 \right)^2 + J_{\text{smooth}} \tag{24}$$

where the additional smoothness penalty term is defined as

$$J_{\text{smooth}} = \gamma \Big[(u_1(k+j) - u_1(k+j-1))^2 + (u_2(k+j) - u_2(k+j-1))^2 \Big], \tag{25}$$

where k denotes the current control step and j indexes the prediction steps within the horizon, γ denoting a tuning parameter that controls the trade-off between control aggressiveness and smoothness. This term discourages abrupt changes in the control signals, promoting gradual adjustments and improving overall actuator longevity and system stability.

The result is a standard NMPC problem, except that the system model is provided by the neural network $\mathcal{F}_\theta$ instead of explicit equations. This finite-horizon optimal control problem is solved at each time step k, yielding an optimal input sequence $(u(k), u(k+1), \ldots, u(k+N-1)$. Receding-horizon control routine: After solving the above optimization, only the first control action $u(k)$ is applied to the plant. At the next time step $k+1$, the horizon is shifted forward: the state is re-measured or estimated as $x(k+1)$, and the optimization is repeated starting from this new state. This receding horizon strategy (illustrated in Figure 6) continually re-initializes the PINN model with feedback from the real system, which helps to correct any prediction errors before they grow unbounded. In other words, although the

PINN might accumulate some error if simulated open-loop over many steps, the MPC loop mitigates this by using frequent state feedback (after the end of every horizon prediction) to reset the prediction.

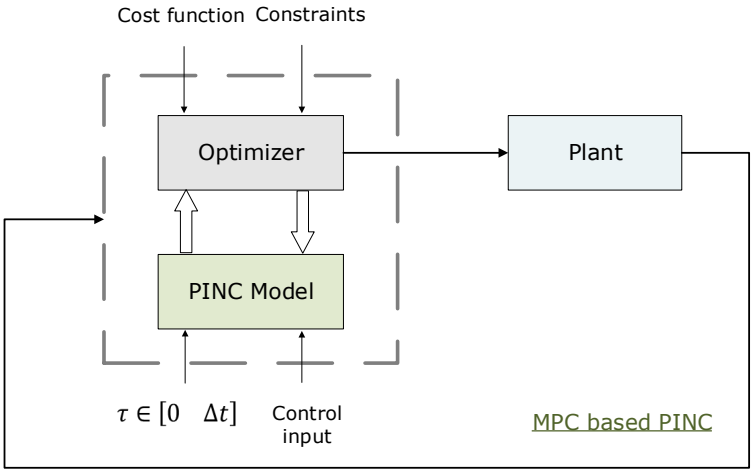


Figure 6: MPC based PINC

To solve this constrained nonlinear optimization problem at each MPC step, a Sequential Quadratic Programming (SQP) approach is adopted. SQP is selected due to its proven efficiency and reliability in solving general constrained nonlinear optimization problems, particularly when constraints and objectives exhibit complex nonlinearities [34, 35]. Specifically, the optimization routine used here is the quasi-Newton SQP method originally proposed by Kraft [34] and subsequently integrated into SciPy's optimization toolbox [35]. In our implementation, function evaluations required by the optimizer are computed via Equation 24, and the corresponding Jacobians are automatically obtained through automatic differentiation. Additionally, control input bounds are directly provided to the solver, eliminating the need for manual clamping

Algorithm 2 summarizes the MPC-based PINC control procedure. Essentially, at each iteration, the controller optimizes future inputs using the PINN (PINC predictor) and implements the first input, thereby achieving closed-loop control that is informed by both data and physics.

Algorithm 2: MPC-Based PINC Control Routine

Input: trained PINN $\mathcal{F}_\theta$; horizon length N; reference trajectory $x^{\text{ref}}(t)$

for $k \leftarrow 0, 1, 2, \ldots$ **do**

- Observe the current state $x(k)$;
- Solve the finite-horizon optimal control problem to get the optimal sequence $\{u(k), \ldots, u(k+N-1)\}$ that minimizes the cost J_{cost} ;
- Apply the first control input $u(k)$ to the plant;
- Advance to the next step: set $k \leftarrow k+1$ and repeat the loop;

The main parameters used in the MPC-based PINC controller are summarized in Table 2. These parameters were determined empirically through extensive testing to ensure stable convergence, smooth actuator behavior, and accurate temperature regulation under varying operating conditions.

Parameter	Symbol / Value	Description
Prediction horizon length	$N_p = 10$	Number of future steps predicted by the controller.
Sampling time	$\Delta t = 1$ s	Time interval between control updates.
Temperature inlet reference	$T_{\text{in,ele}}^{\text{ref}} = 315$ K	Target inlet temperature.
Temperature difference reference	$\Delta T^{\text{ref}} = 5$ K	Desired outlet-inlet temperature difference.
Control input bounds	$0 \le u_2 \le 1$	Cooling pump.
Control input bounds	$0.3 \le u_1 \le 1$	Pump in the principal circuit.
Weight on u_1 magnitude	$\alpha_1 = 0.02$	Penalizes excessive hot-side flow control effort.
Weight on u_2 magnitude	$\alpha_2 = 0.02$	Penalizes excessive cold-side flow control effort.
Weight on ΔT deviation	$\beta = 10$	Penalizes deviation from ΔT^{ref}.
Weight on $T_{\text{in,ele}}$ deviation	$\alpha_0 = 10$	Penalizes deviation from ΔT^{ref}.
Smoothness penalty	$\gamma = 0.1$	Penalizes large variations between successive control moves.
Maximum closed-loop steps	$M = 1000$	Total number of simulation steps in the closed-loop run.
SQP inner iterations	$N_{\text{iter}} = 30$	Maximum optimization iterations per MPC step.

Table 2: Summary of the main parameters used in the MPC-based PINC controller.

7. Dual PI Controllers Design and Tuning

For benchmarking purposes, a dual proportional-integral (PI) controller was implemented to regulate the electrolyzer's thermal dynamics. The structure consists of two independent control loops: one for the outlet-inlet temperature difference ($\Delta T = T_{\text{out,ele}} - T_{\text{in,ele}}$) via the hot-side coolant valve (u_1), and another for the inlet temperature ($T_{\text{in,ele}}$) via the cold-side coolant valve (u_2), the two controllers can be seen the following figure 7. This decoupled configuration mirrors common industrial practice for temperature management in electrolyzers and similar heat-exchange systems.

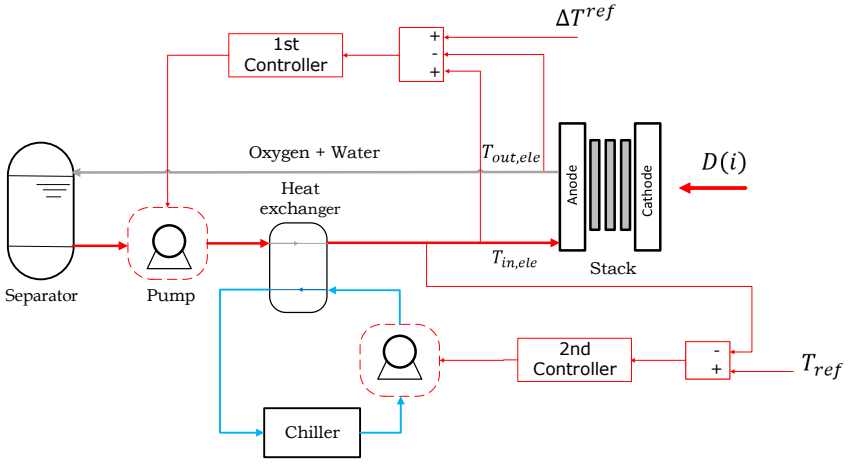


Figure 7: Schematic of the two-loop thermal control strategy for the PEM electrolysis system

The discrete-time control laws are given by:

$$\begin{aligned} u_1(k) &= K_{p1}\, e_{\Delta T}(k) + K_{i1} \sum_{j=0}^{k} e_{\Delta T}(j)\, \Delta t, \\ u_2(k) &= K_{p2}\, e_{T_{\text{in}}}(k) + K_{i2} \sum_{j=0}^{k} e_{T_{\text{in}}}(j)\, \Delta t, \end{aligned} \tag{26}$$

where $e_{\Delta T}(k) = \Delta T_{\text{ref}} - \Delta T(k)$ and $e_{T_{\text{in}}}(k) = T_{\text{ref}} - T_{\text{in,ele}}(k)$ denote the respective control errors. Instead of manual Ziegler-Nichols or trial-and-error tuning, the four PI gains were identified automatically using a global optimization procedure based on differential evolution. The objective function minimized a composite cost:

$$J = 10\,\frac{\overline{|\,\Delta T - 5\,|}}{5} + 20\,\frac{\overline{|\,T_{\text{in,ele}} - 315\,|}}{315} + \lambda_{\text{smooth}}\,\overline{\left(\nabla \Delta T\right)^2}, \tag{27}$$

where the first two terms represent normalized tracking errors, and the last term penalizes rapid temperature fluctuations to promote smooth, physically realistic responses.

The procedure was performed under random stepwise variations of the stack current (5-50 A) over 1000 s of simulation. This scenario allowed the optimization to account for dynamic coupling between the two control loops. The resulting gains, shown in Table 3, were found to yield stable and accurate temperature regulation across both steady and transient conditions. The PI control was implemented in discrete time with a sampling period $\Delta t = 1$ s, matching the MPC controller's timing. An anti-windup mechanism was included by freezing the integral action when either actuator reached saturation.

Table 3: Automatically identified tuning parameters of the dual PI controller.

Controller	K_p	K_i	Controlled Variable	Actuator
PI_1 (ΔT loop)	2.83	0.0015	ΔT	u_1 (hot-side coolant valve)
PI_2 ($T_{\text{in,ele}}$ loop)	1.46	0.0023	$T_{\text{in,ele}}$	u_2 (cold-side coolant valve)

Both controllers were tested under identical boundary conditions and disturbance profiles to ensure fair comparison in the results section.

8. Results

After the training was finished, the performance of the proposed self-loop PINN model was evaluated on previously unseen excitation inputs to assess its prediction accuracy and generalization. In this test, we applied randomly generated time-series signals to the two water flow inputs (u_1 and u_2) and the electrical current input (i). As shown in Figure 8, these excitation signals vary significantly over time, providing a challenging scenario that was not encountered during the training phase. The trained self-loop PINN was then used to predict the thermal state evolution one step ahead in a recursive fashion for each new input, repeatedly feeding its output as the next initial state in a recursive fashion.

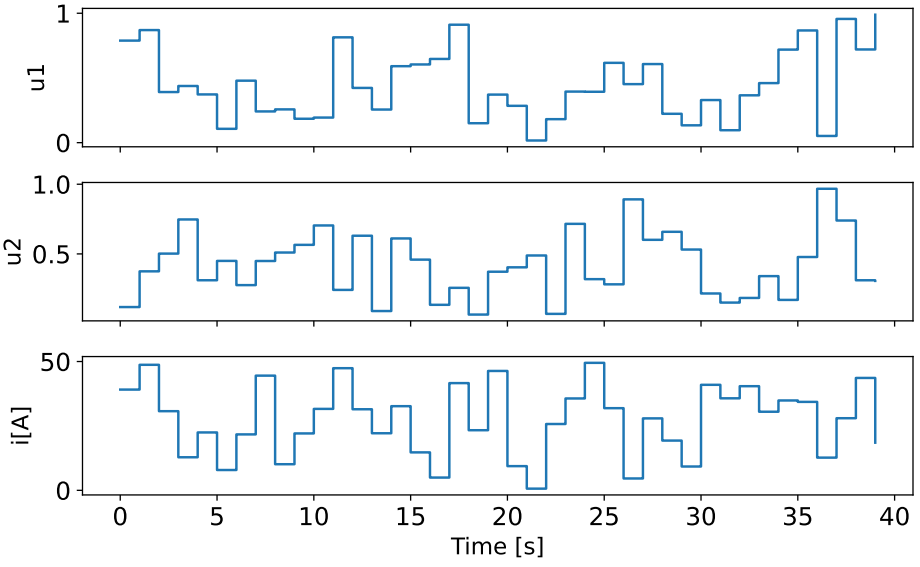


Figure 8: Random excitation input profiles for the stack current and the two coolant flow rates

Figure 9 presents a comparison of the PINN-predicted temperature states (x_1, x_2, x_3) against the reference solution obtained by a classical fourth-order Runge-Kutta (RK4) ODE solver. It is evident that the PINN predictions (solid lines) closely track the ground-truth trajectories (dashed lines) for all three states. The model captures both the short-term transients and long-term trends induced by the fluctuating inputs with high fidelity. Notably, no significant divergence or drift is observed between the PINN and RK4 curves over the entire simulation horizon, indicating that the physics-informed network remains stable and accurate even when operating in a self-loop (multi-step prediction) mode.

To quantify the accuracy of the PINN, the root-mean-square error (RMSE) was computed for each state trajectory with respect to the RK4 solution. The resulting RMSE values are 0.02640725 for x_1, 0.04075646 for x_2, and 0.02495539 for x_3. This corresponds to a combined RMSE of approximately 0.09212 across the three temperature states. These low error magnitudes confirm that the PINN's one-step-ahead forecasts are highly precise. The model effectively generalizes to the new, randomly varying input profiles, demonstrating that it has learned the underlying thermal dynamics rather than merely overfitting to the training scenarios. The predictions remain robust throughout the simulation: even as the input signals push the system through a wide operating range, the PINN consistently produces physically plausible and accurate state estimates without any noticeable degradation in performance.

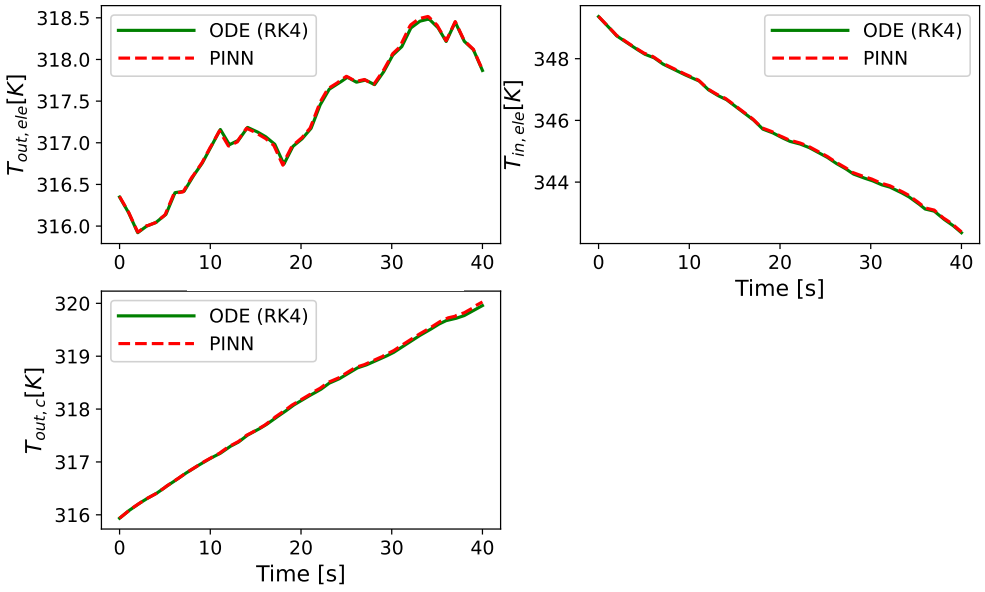


Figure 9: One-step PINN predicted temperature trajectories (dashed lines) versus ground-truth simulation of the physics model (solid lines) for a sample input current profile.

In addition to accuracy and robustness, a major advantage of the proposed PINN surrogate model is its computational efficiency. Each prediction of the next state is obtained via a single forward pass of the neural network, which is virtually instantaneous compared to the iterative calculations required by traditional ODE solvers. In our experiments, the PINN significantly outperformed the RK4 integration method in simulating the system's thermal behavior, achieving comparable accuracy in approximately one-third of the computational time (PINN self-loop propagation: 0.0789 seconds vs. RK4 solver: 0.5232 seconds). This speed advantage is attributed to the RK4 method requiring multiple intermediate computational steps to reach similar accuracy. This drastic reduction in execution time means that the PINN can generate long-horizon predictions in a fraction of the time of the ODE solver. Such speed is particularly beneficial for real-time applications: for instance, in a MPC framework, the controller needs to evaluate the model many times per control cycle, and the PINN's ability to provide rapid predictions can significantly improve the feasible control update rate. In summary, the results indicate that the self-loop PINN model serves as a fast, accurate, and generalizable surrogate for the PEM electrolyzer's thermal dynamics, matching the fidelity of classical integration methods while offering substantial improvements in speed and computational efficiency.

In closed-loop experiments, the trained PINN model is embedded in a NMPC framework to regulate the electrolyzer's temperature in real time. Three representative scenarios were tested: a constant 41 A load, random step changes in the load current, and a slowly varying (sinusoidal) temperature setpoint under fluctuating current. In these

scenarios, the neural network predicts the electrolyzer dynamics over a 10-step horizon. At each sampling instant, the MPC implements only the first optimized control action before advancing to the subsequent optimization step.

Scenario 1 - Steady-State Operation (Constant Current Profile): Figure 10 illustrates the thermal regulation performance of the proposed PINN-based MPC and the conventional dual PI controller under constant-load operation. In this scenario, the electrolyzer stack operates at a fixed current of approximately 43 A, resulting in steady internal heat generation and a near-static thermal boundary condition. This setup evaluates the controllers' ability to maintain the desired inlet temperature $T_{in,ele}$ and temperature gradient ΔT under steady-state conditions without external disturbances.

Both controllers successfully regulate the system to the target values of 315 K for $T_{in,ele}$ and 5 K for ΔT. However, distinct differences are observed in their transient responses. The MPC (red curves) exhibits a faster convergence toward the setpoints, reaching the desired thermal states with minimal overshoot and smooth settling. In contrast, the PI controller (blue curves) responds more conservatively, showing a slightly slower rise and a marginally delayed settling period. This highlights the MPC's ability to optimize the transient phase even in a static operating condition, minimizing thermal deviation and reducing the time required to achieve steady operation.

The MPC control actions (u_1, u_2) demonstrate coordinated and efficient actuator behavior. This indicates the MPC's capability to exploit actuator limits intelligently without inducing oscillatory or unstable behavior. The steady current profile ensures that after the initial transient, both u_1 and u_2 remain constant, maintaining stable thermal conditions.

Overall, under constant-load conditions, both controllers perform satisfactorily, maintaining the desired thermal states with negligible steady-state error. Yet, the MPC achieves faster convergence, smoother transients, and more balanced actuator effort, emphasize the MPC's efficiency in regulating coupled thermal dynamics in steady scenarios.

In addition to improved control accuracy, the PINN-based approach offers significant computational advantages. A timing comparison showed that evaluating the PINN surrogate within the NMPC loop dramatically reduces computation time relative to using a traditional ODE solver. For the 1000 s simulation of Scenario 1, the NMPC with the PINN (denoted "PINC") required about 874 seconds of CPU time, whereas the same NMPC using a standard fourth-order Runge-Kutta (RK4) integrator took approximately 4443 seconds. This five-fold speedup underscores the benefit of the PINN surrogate model in enabling faster control loop updates.

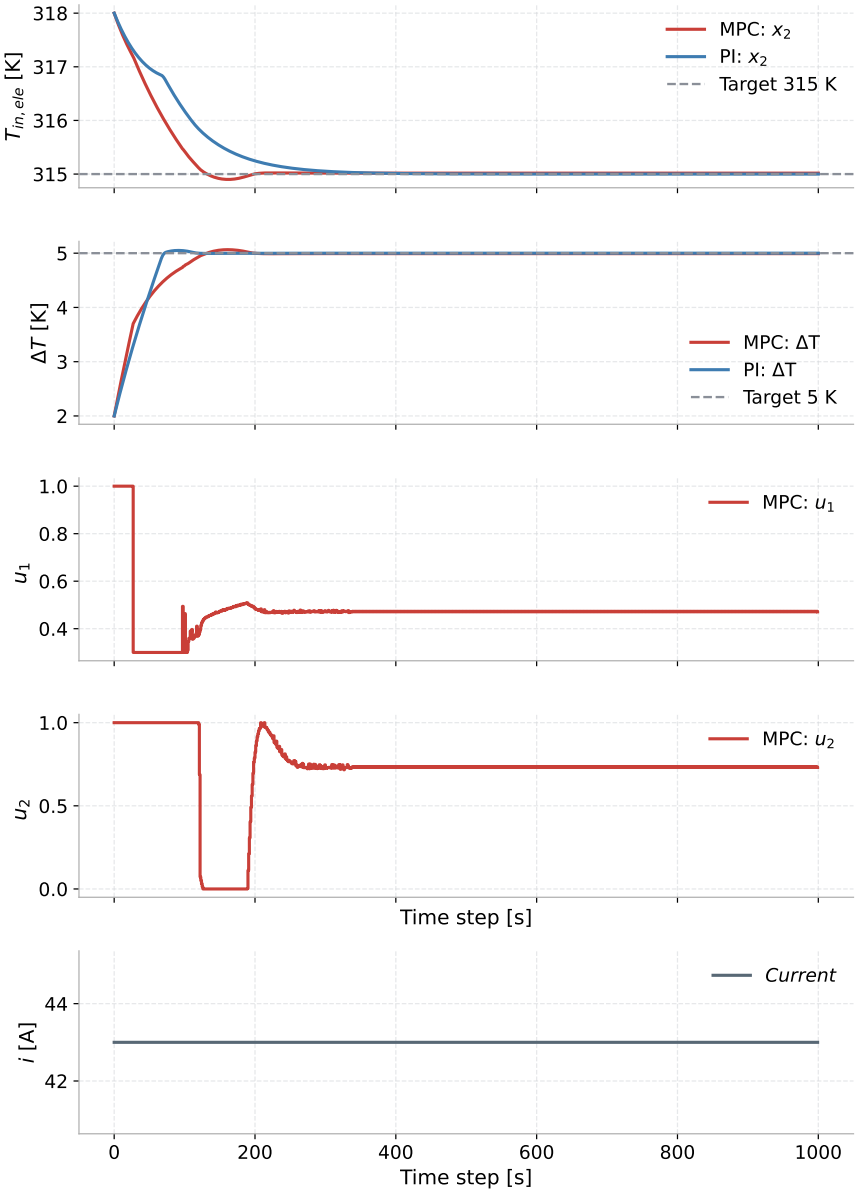


Figure 10: Closed-loop setpoint regulation using an NMPC powered by the PINN surrogate model.

Scenario 2 - Variable Load Operation (Dynamic Current Profile): Figure 11 presents the dynamic performance of the proposed PINN-based MPC and the classical dual PI controller when the electrolyzer stack is subjected to a time-varying current profile ranging from 5 to 50 A. This scenario emulates realistic load fluctuations, where heat generation within the stack changes abruptly, challenging both controllers to maintain the target inlet temperature and temperature gradient.

The results clearly demonstrate the advantages of the MPC approach under dynamic operating conditions. Following each current change, the MPC (red curves) promptly adjusts both control inputs u_1 and u_2 to counteract the thermal disturbance, keeping $T_{in,ele}$ and ΔT close to their setpoints with minimal overshoot and settling faster. Notably, the MPC maintains strong disturbance rejection and coordinated actuation throughout the variable-load sequence. The control actions u_1 (ΔT control valve) and u_2 (inlet coolant valve) vary smoothly and preemptively in response to the changing current, reflecting the predictive nature of the NMPC framework. The PI loops, on the other hand, lack foresight and must correct errors after they occur, leading to slightly oscillatory transient behavior and longer recovery times.

Despite the high nonlinearity and coupling between the thermal states, both controllers succeed in keeping the electrolyzer within the safe thermal envelope. However, the MPC achieves this with faster response, lower overshoot, and better coordination of the actuators, while the PI exhibits lag and mild oscillations following large current transitions. This oscillation is consistent with the cross-coupling between the two PI loops: as one loop compensates for temperature deviation, the other reacts to restore the gradient, producing a damped interaction not observed in the NMPC case.

In summary, under variable load conditions, the PINN-based MPC demonstrates clear superiority: it provides anticipatory compensation, tighter temperature regulation, and smoother actuator effort, ensuring reliable operation even under rapidly changing heat loads. The dual PI controller, although simpler and computationally lighter, cannot match the predictive coordination or dynamic precision of the MPC. These findings highlight the potential of PINN-assisted NMPC to deliver real-time, model-informed control for highly coupled electrothermal systems subject to fluctuating power demands.

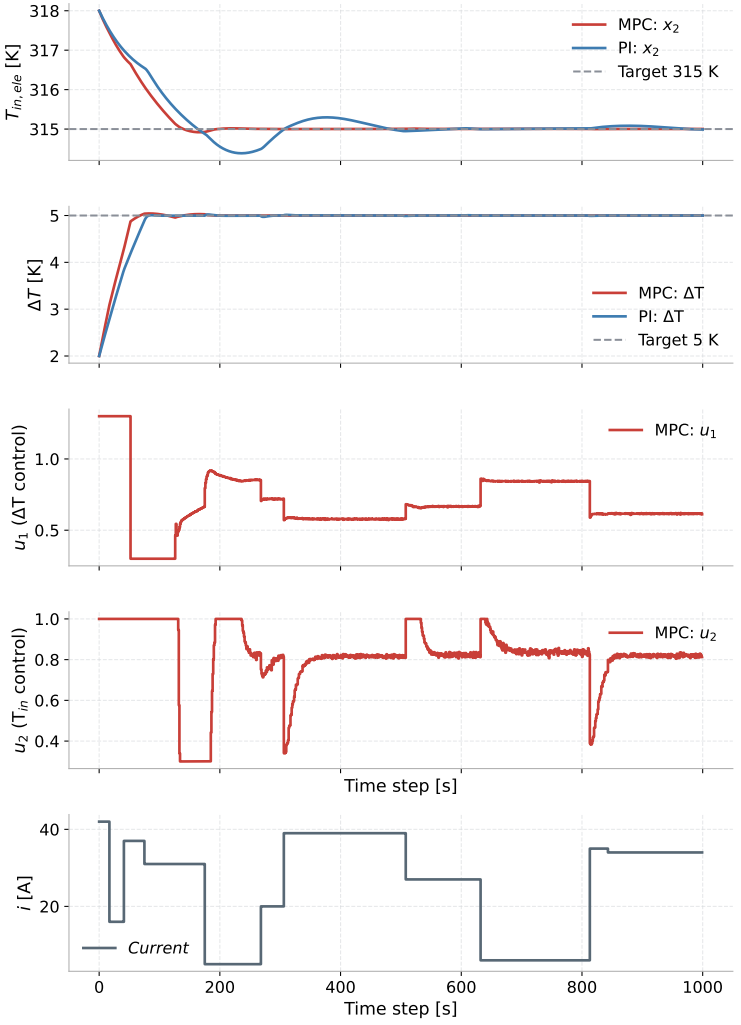


Figure 11: Closed-loop setpoint regulation using an NMPC powered by the PINN surrogate model, for changing current supply

Scenario 3 - Dynamic Setpoint Tracking (Sinusoidal Temperature Reference): In this scenario, the inlet temperature setpoint is modulated as a low-frequency sinusoidal signal oscillating within ±0.5 K around the nominal value of 315 K, while the electrolyzer current continues to vary randomly. Figure 12 illustrates the dynamic performance of the proposed PINN-based MPC under these challenging and time-varying operating conditions. This case evaluates the

controller's capability to handle a continuously changing reference while maintaining the desired temperature gradient ΔT within safe limits.

The obtained results clearly demonstrate that the MPC maintains excellent tracking performance despite the combined disturbances. The inlet temperature $T_{in,ele}$ follows the sinusoidal reference closely, with peak tracking error below 0.1 K and virtually no overshoot. The predictive horizon enables the controller to anticipate upcoming reversals in the setpoint trajectory, allowing for smooth and preemptive control actions. Both control inputs, u_1 and u_2, vary continuously and in coordination to counteract the coupled thermal dynamics, ensuring stable operation and precise regulation of the target thermal states. Throughout the experiment, the ΔT constraint remains satisfied at all times, confirming the MPC's ability to maintain the thermal envelope while tracking a moving target.

It is important to note that the dual PI controller was not evaluated in this scenario. Given the increased complexity of the control objectives-simultaneous dynamic reference tracking and fluctuating heat generation-the reactive nature of the PI approach would not be sufficient to achieve satisfactory performance. The MPC, in contrast, leverages its model-based predictive capabilities to anticipate future system behavior and adjust the actuation proactively, demonstrating superior adaptability and precision under highly dynamic conditions.

In summary, the results confirm that the PINN-based MPC successfully handles dynamic setpoints and nonlinear coupled dynamics with high accuracy and stability. Its predictive and constraint-aware structure ensures smooth and precise temperature tracking even under rapidly changing thermal and electrical loads, underscoring its suitability for real-time control of complex electrothermal systems.

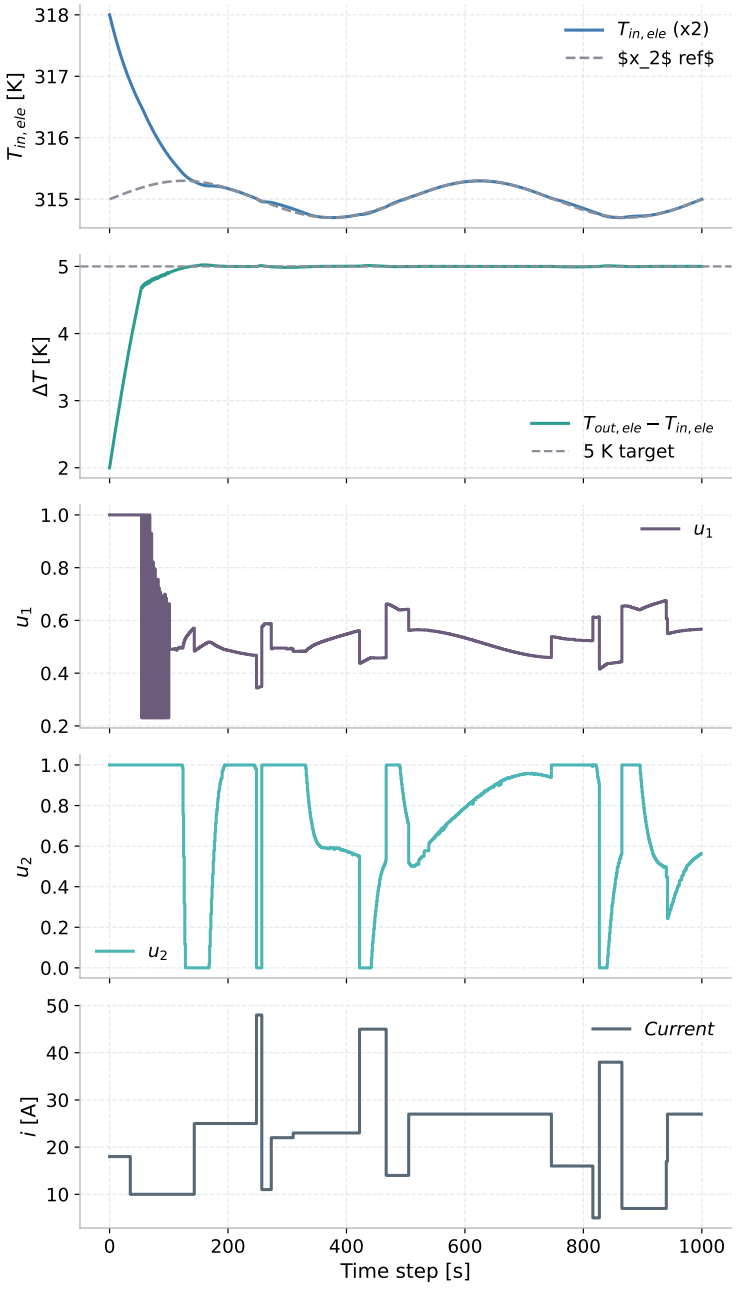


Figure 12: Closed-loop setpoint regulation using an NMPC powered by the PINN surrogate model, for changing current supply

In summary, the PINN-based MPC controller meets all thermal regulation objectives under both steady and dynamic conditions. Across all scenarios, the controller achieves precise temperature tracking and fast convergence to setpoints while strictly enforcing the thermal-gradient constraint. The control inputs adjust and remain within their limits, reflecting an optimal allocation of effort between heating and cooling. These results validate the proposed self-loop PINN surrogate as an accurate, high-speed model for real-time control. They also confirm that the PINC strategy can effectively regulate a PEM electrolyzer's temperature under fluctuating loads and changing setpoints, with dramatically reduced computational demand. The demonstrated performance underscores the potential of physics-informed neural surrogates for advanced control of complex energy systems.

9. Discussion

The results above demonstrate that the self-loop PINN approach is capable of learning and reproducing the thermal dynamics of the PEM electrolyzer with high fidelity. This section discusses the principal advantages of the proposed approach, its limitations, and potential avenues for future enhancement.

By incorporating the known physics directly into the training process, the PINN model respects energy conservation and the qualitative behavior of the thermal system. This means the model is less likely to produce physically implausible predictions. Even with limited data, the physics prior guides the model to reasonable solutions. In our case, the accuracy achieved was on par with the numerical integration of the known model, indicating that the PINN successfully learned the underlying equations. This is a strong testament to the approach-essentially, the neural network became a surrogate for the ODE solver. Unlike a traditional PINN trained over a fixed time window, our one-step PINN is inherently generalizable to different time horizons and input sequences. The network has learned the transition dynamics rather than a specific trajectory. This means it can be used for arbitrary input profiles, as evidenced by its performance on test scenarios that were not in training. This is crucial for control applications: the model can be embedded in a control algorithm that might explore many possible input sequences, and the PINN will be able to predict the outcome of each, even if that exact sequence was never seen before, as long as it is within the realm of the training distribution (which was quite broad in our case). Once trained, the PINN model runs very fast. Evaluating a neural network is typically much faster than solving differential equations, especially as the complexity of the physical model or the required resolution increases. In an MPC context, for instance, a model might need to be evaluated hundreds or thousands of times per control step (inner loop of optimization). A speed-up of an order of magnitude can directly translate to either faster computation (allowing higher control frequency) or the ability to incorporate more complexity (longer horizons, finer discretization, more scenarios) within the same time. Our timing tests showed a clear benefit, and this gap would likely widen for more complex multi-physics models (where the PINN could still approximate the mapping in one step, whereas a solver might slow down significantly).

Despite its strengths, the approach has some limitations; for instance, training a PINN can be more involved than training a standard neural network. The inclusion of the physics loss often makes the problem stiffer and can require careful tuning of hyperparameters (like the weights λ_{phys}, learning rates, network architecture). In the present implementation, a combination of the Adam and L-BFGS optimization algorithms was employed, which is a standard practice. Nevertheless, the training process remained slower compared to that of a purely data-driven network trained on the same dataset. This is not a serious issue for offline training, but it is a consideration. In our work, the PINN was only trained through physics; however, in real applications, the equations might not totally obey the observation. In this case, data are necessary, for this, one can integrate real data on the training process by adding it to the loss alongside the physics residual. This would allow the network to correct for unmodeled phenomena by adjusting its output to fit data, while still obeying the general structure of the known physics. This approach is particularly powerful in scenarios where the underlying model is only partially known or certain terms remain uncertain. In our case, since the data were generated using a fully known model, this aspect was not demonstrated; however, it represents a major perspective for future work involving experimental data.

In summary, the self-loop PINN model provides high-fidelity predictions and several practical benefits (physical consistency, generalization, and speed) at the cost of increased training complexity and domain limitations. Despite these trade-offs, the combination of physics-based accuracy and computational efficiency makes this approach a promising tool for electrolyzer modeling and control. Future work will address the remaining issues by incorporating experimental data, allowing variable time steps, and expanding the model structure as needed to further improve the accuracy and robustness of the approach.

10. Conclusion

This study presented a real-time modeling and control framework for a PEM electrolyzer based on a self-loop Physics-Informed Neural Network (PINN). By embedding the governing thermal physics directly into the neural network's architecture, the proposed approach combines the interpretability of first-principles models with the computational efficiency of data-driven learning. The resulting self-loop PINN accurately reproduces the nonlinear thermal dynamics of the electrolyzer, achieving a combined RMSE of only 0.09212 across all temperature states when compared with a high-fidelity RK4 solver. Moreover, the model operates several times faster than conventional numerical integration, enabling rapid long-horizon predictions suitable for real-time applications.

When deployed within a nonlinear Model Predictive Control (NMPC) framework, the PINN surrogate successfully regulated the electrolyzer's temperature under a variety of operating scenarios, including steady-state, variable-load, and dynamically varying setpoint conditions. Across all cases, the PINN-based MPC consistently achieved precise temperature tracking, fast convergence, and smooth actuator coordination while satisfying all operational constraints. Compared to traditional PI control, the proposed method demonstrated superior transient performance, improved disturbance rejection, and up to a fivefold reduction in computational time, confirming its suitability for embedded and real-time control.

The presented framework illustrates the power of physics-informed machine learning as a surrogate modeling approach for complex electrothermal systems. Beyond the immediate application to PEM electrolyzers, this methodology can be generalized to other energy conversion devices where strong coupling between thermal and electrical dynamics exists.

Future work will focus on extending the self-loop PINN to include more detailed physics-such as electrochemical kinetics, multiphase transport, and degradation mechanisms-to further enhance model fidelity. Additionally, integrating online adaptation and continual learning will allow the PINN to update its parameters in real time as new data become available, maintaining accuracy under evolving operating conditions. Finally, experimental validation and digital-twin integration will bridge the gap between simulation and practice, paving the way for the deployment of PINN-based predictive control in real PEM electrolyzer systems. These advancements promise to enhance the efficiency, safety, and reliability of renewable hydrogen production.

Acknowledgements

This work has been supported by the EIPHI Graduate School (contract ANR-17-EURE-0002) and the Region Bourgogne Franche-Comté.

References

[1] Martín David, Carlos Ocampo-Martínez, and Ricardo Sánchez-Peña. Advances in alkaline water electrolyzers: A review. *Journal of Energy Storage*, 23:392–403, 2019.

[2] Vahid Madadi Avargani, Sohrab Zendehboudi, Noori M Cata Saady, and Maurice B Dusseault. A comprehensive review on hydrogen production and utilization in north america: Prospects and challenges. *Energy Conversion and Management*, 269:115927, 2022.

[3] H Sayed-Ahmed, Árpad Istvan Toldy, and A Santasalo-Aarnio. Dynamic operation of proton exchange membrane electrolyzers—critical review. *Renewable and Sustainable Energy Reviews*, 189:113883, 2024.

[4] Pierre Olivier, Cyril Bourasseau, and Pr Belkacem Bouamama. Low-temperature electrolysis system modelling: A review. *Renewable and Sustainable Energy Reviews*, 78:280–300, 2017.

[5] Islam Zerrougui, Zhongliang Li, and Daniel Hissel. Investigating bubble impacts on pem electrolysis performance through enhanced multiphysics modeling. *International Journal of Hydrogen Energy*, 98:626–638, 2025.

[6] Tohid Adibi, Atta Sojoudi, and Suvash C Saha. Modeling of thermal performance of a commercial alkaline electrolyzer supplied with various electrical currents. *International Journal of Thermofluids*, 13:100126, 2022.

[7] Islam Zerrougui, Zhongliang Li, and Daniel Hissel. Toward optimal operations of long-lifetime pem electrolysis: Degradation mechanisms, modeling, diagnostics, and control. *International Journal of Hydrogen Energy*, 193:152297, 2025.

[8] Nanzhe Wang, Yuntian Chen, and Dongxiao Zhang. A comprehensive review of physics-informed deep learning and its applications in geoenergy development. *The Innovation Energy*, 2(2):100087–1, 2025.

[9] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[10] Yuan Jiang, Zheng Liu, Pouya Kabirzadeh, Yulun Wu, Yumeng Li, Nenad Miljkovic, and Pingfeng Wang. Multi-fidelity physics-informed convolutional neural network for heat map prediction of battery packs. *Reliability Engineering & System Safety*, 256:110752, 2025.

[11] Xinbin Liang, Ying Liu, Siliang Chen, Xilin Li, Xinqiao Jin, and Zhimin Du. Physics-informed neural network for chiller plant optimal control with structure-type and trend-type prior knowledge. *Applied Energy*, 390:125857, 2025.

[12] Islam Zerrougui, Zhongliang Li, and Daniel Hissel. Physics-informed neural network for modeling and predicting temperature fluctuations in proton exchange membrane electrolysis. *Energy and AI*, page 100474, 2025.

[13] Jonas Nicodemus, Jonas Kneifl, Jörg Fehr, and Benjamin Unger. Physics-informed neural networks-based model predictive control for multi-link manipulators. *IFAC-PapersOnLine*, 55(20):331–336, 2022.

[14] Saviz Mowlavi and Saleh Nabi. Optimal control of pdes using physics-informed neural networks. *Journal of Computational Physics*, 473:111731, 2023.

[15] Jostein Barry-Straume, Arash Sarshar, Andrey A Popov, and Adrian Sandu. Physics-informed neural networks for pde-constrained optimization and control. *arXiv preprint arXiv:2205.03377*, 2022.

[16] Eric Aislan Antonelo, Eduardo Camponogara, Laio Oriel Seman, Jean Panaioti Jordanou, Eduardo Rehbein de Souza, and Jomi Fred Hübner. Physics-informed neural nets for control of dynamical systems. *Neurocomputing*, 579:127419, 2024.

[17] Yinping Li and Li Liu. Physics-informed neural network-based nonlinear model predictive control for automated guided vehicle trajectory tracking. *World Electric Vehicle Journal*, 15(10):460, 2024.
[18] Yingzhe Zheng, Cheng Hu, Xiaonan Wang, and Zhe Wu. Physics-informed recurrent neural network modeling for predictive control of nonlinear processes. *Journal of Process Control*, 128:103005, 2023.
[19] Guoquan Wu, Wallace Tan Gian Yion, Khoa Le Nguyen Quang Dang, and Zhe Wu. Physics-informed machine learning for mpc: Application to a batch crystallization process. *Chemical Engineering Research and Design*, 192:556–569, 2023.
[20] Lei Xu, Chunhua Yang, Xiaodong Xu, Biao Luo, and Tingwen Huang. Physics-informed stochastic configuration network promoted model predictive control with multi-objective optimization. *Artificial Intelligence Review*, 58(9):281, 2025.
[21] Tim-Lukas Habich, Aran Mohammad, Simon FG Ehlers, Martin Bensch, Thomas Seel, and Moritz Schappler. Generalizable and fast surrogates: Model predictive control of articulated soft robots using physics-informed neural networks. *arXiv preprint arXiv:2502.01916*, 2025.
[22] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.
[23] Félix Fernández de la Mata, Alfonso Gijón, Miguel Molina-Solana, and Juan Gómez-Romero. Physics-informed neural networks for data-driven simulation: Advantages, limitations, and opportunities. *Physica A: Statistical Mechanics and its Applications*, 610:128415, 2023.
[24] Marcelo Carmo, David L Fritz, Jürgen Mergel, and Detlef Stolten. A comprehensive review on pem water electrolysis. *International journal of hydrogen energy*, 38(12):4901–4934, 2013.
[25] Fawwad Nisar Khatib, Tabbi Wilberforce, Oluwatosin Ijaodola, Emmanuel Ogungbemi, Zaki El-Hassan, Andy Durrant, James Thompson, and Abdul Ghani Olabi. Material degradation of components in polymer electrolyte membrane (pem) electrolytic cell and mitigation mechanisms: A review. *Renewable and Sustainable Energy Reviews*, 111:1–14, 2019.
[26] Chunjun Huang, Xin Jin, Yi Zong, Shi You, Chresten Træholt, and Yi Zheng. Operational flexibility analysis of alkaline electrolyzers integrated with a temperature-stabilizing control. *Energy Reports*, 9:16–20, 2023.
[27] Ruomei Qi, Jiarong Li, Jin Lin, Yonghua Song, Jiepeng Wang, Qiangqiang Cui, Yiwei Qiu, Ming Tang, and Jian Wang. Thermal modeling and controller design of an alkaline electrolysis system under dynamic operating conditions. *Applied Energy*, 332:120551, 2023.
[28] Jian Dang, Fuyuan Yang, Yangyang Li, Yingpeng Zhao, Minggao Ouyang, and Song Hu. Experiments and microsimulation of high-pressure single-cell pem electrolyzer. *Applied Energy*, 321:119351, 2022.
[29] R Keller, E Rauls, M Hehemann, M Müller, and M Carmo. An adaptive model-based feedforward temperature control of a 100 kw pem electrolyzer. *Control Engineering Practice*, 120:104992, 2022.
[30] Øystein Ulleberg. Stand-alone power systems for the future: optimal design, operation and control of solar-hydrogen energy systems. 1998.
[31] Ruomei Qi, Jiarong Li, Jin Lin, Yonghua Song, Jiepeng Wang, Qiangqiang Cui, Yiwei Qiu, Ming Tang, and Jian Wang. Design of the pid temperature controller for an alkaline electrolysis system with time delays. *International Journal of Hydrogen Energy*, 48(50):19008–19021, 2023.
[32] Aydin Demircioğlu. The effect of feature normalization methods in radiomics. *Insights Into Imaging*, 15(1):2, 2024.
[33] Elham Kiyani, Khemraj Shukla, Jorge F Urbán, Jérôme Darbon, and George Em Karniadakis. Which optimizer works best for physics-informed neural networks and kolmogorov-arnold networks? *arXiv preprint arXiv:2501.16371*, 2025.
[34] Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.
[35] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.