electronics

MDPI

Article

Novel Adaptive Intelligent Control System Design

Worrawat Duanyai [1], Weon Keun Song [2,*], Min-Ho Ka [3], Dong-Wook Lee [4] and Supun Dissanayaka [2]

[1] Department of Robotics and Computational Intelligence Systems, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand; 65016082@kmitl.ac.th
[2] Department of Robotics and AI Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand; supun.di@kmitl.ac.th
[3] School of Integrated Technology, Yonsei University, Incheon 21983, Republic of Korea; kaminho@yonsei.ac.kr
[4] Medical Imaging & Intelligent Reality Lab, Convergence Medicine Asan Medical Center, Seoul 05505, Republic of Korea; dwleee1971@gmail.com
* Correspondence: bauman98@naver.com

Abstract

A novel adaptive intelligent control system (AICS) with learning-while-controlling capability is developed for a highly nonlinear single-input single-output plant by redesigning the conventional model reference adaptive control (MRAC) framework, originally based on first-order Lyapunov stability, and employing customized neural networks. The AICS is designed with a simple structure, consisting of two main subsystems: a meta-learning-triggered mechanism-based physics-informed neural network (MLTM-PINN) for plant identification and a self-tuning neural network controller (STNNC). This structure, featuring the triggered mechanism, facilitates a balance between high controllability and control efficiency. The MLTM-PINN incorporates the following: (I) a single self-supervised physics-informed neural network (PINN) without the need for labelled data, enabling online learning in control; (II) a meta-learning-triggered mechanism to ensure consistent control performance; (III) transfer learning combined with meta-learning for finely tailored initialization and quick adaptation to input changes. To resolve the conflict between streamlining the AICS's structure and enhancing its controllability, the STNNC functionally integrates the nonlinear controller and adaptation laws from the MRAC system. Three STNNC design scenarios are tested with transfer learning and/or hyperparameter optimization (HPO) using a Gaussian process tailored for Bayesian optimization (GP-BO): (scenario 1) applying transfer learning in the absence of the HPO; (scenario 2) optimizing a learning rate in combination with transfer learning; and (scenario 3) optimizing both a learning rate and the number of neurons in hidden layers without applying transfer learning. Unlike scenario 1, no quick adaptation effect in the MLTM-PINN is observed in the other scenarios, as these struggle with the issue of dynamic input evolution due to the HPO-based STNNC design. Scenario 2 demonstrates the best synergy in controllability (best control response) and efficiency (minimal activation frequency of meta-learning and fewer trials for the HPO) in control.

Keywords: adaptive intelligent control; online learning and control; meta-learning-triggered mechanism; transfer learning; PINN; self-supervised learning without labelled data; self-tuning neural network controller; Bayesian optimization

Academic Editor: Na Dong

Received: 30 June 2025
Revised: 27 July 2025
Accepted: 4 August 2025
Published: 7 August 2025

Citation: Duanyai, W.; Song, W.K.; Ka, M.-H.; Lee, D.-W.; Dissanayaka, S. Novel Adaptive Intelligent Control System Design. *Electronics* **2025**, *14*, 3157. https://doi.org/10.3390/electronics14153157

1. Introduction

Traditional control algorithms often struggle to efficiently manage intricate nonlinear systems. This study proposes a novel adaptive intelligent control system (AICS) as an

alternative to the conventional model reference adaptive control (MRAC) system [1], which is restricted in effectively controlling highly nonlinear systems. A higher-order Lyapunov function may serve as a viable solution for enhancing the MRAC system's performance by offering greater adaptive flexibility. Additionally, combining the MRAC with traditional control systems such as PID control, H∞ control, sliding mode control, and others can be considered as alternative approaches. However, it is important to note that not only the analytical discovery of the Lyapunov function but also the design of the hybrid control systems require extensive expertise in mathematics and control theory. This highlights the necessity of advanced intelligent control frameworks to address these challenges.

The key advantages, including a highly parallel architecture, adaptability, nonlinear mapping capability, and robustness, strongly encourage the use of neural networks for nonlinear system identification and/or control. According to a paper [2], the term "intelligent control" was first introduced in the 1970s. Since then, diverse intelligent control systems have gained considerable attention in modern engineering applications for their potential to address challenges posed by highly nonlinear and intricate dynamic processes. Branches in intelligent control include neural network (NN)-based control, fuzzy control, genetic algorithm-based control, planning system-based control, expert system-based control, and hybrid system-based control, all of which are active research areas. This paper offers the literature review of NN-based control strategies that incorporate either a neural network-based plant identifier [1,3], a neural network controller [4–9], or both [10–17]. Most studies [1,3–7,9,13,16] permitted online learning, but some [8,11,17] did not. Studies [1,7,11,15–17] explored single-input single-output (SISO) plants, while another [3] focused on multiple-input multiple-output plants. One study [13] handled both. Their applications primarily targeted the diverse branches of robotics [4–6,9]. A study [9] tested an inverted pendulum, a cartpole, a vehicle, a pendubot, and a power system with recurrent NN-based controllers. From the perspective of algorithmic principles, NN-based intelligent control can be categorized into the following: (I) adaptive control [1,4–8,12,13,15–17]; (II) adaptive inverse control [10]; (III) internal model control [11,12]; (IV) predictive control [12]; (V) adaptive critic control [14]; (VI) reinforcement learning-based control [12]. Secondary considerations, which can be challenging and may require extensive simulations, could be viewed as potential weaknesses, complicating control system design and impacting control performance. The following are some examples: adaptation laws [1]; a fictitious controller [4]; subsystems for generating an auxiliary term to adjust a control signal [7]; a robustness filter for plant-model mismatch [11,12] and control law [11] in (III); a numerical optimization routine [12] in (IV); and control and feedback laws [15]. Ultimately, the aforementioned studies [1,4,7,11,12,15] required one or more additional subsystems to implement their control strategies, which are redundant for the proposed AICS. Studies [18,19] using reinforcement learning based on rewards and penalties inherently exhibit a trial-and-error nature. This can lead to risky behaviours during exploration with potentially harmful consequences, particularly in critical real-world applications based on online learning-while-controlling. Despite significant progress in safety from recent studies [20–23], reinforcement learning-based approaches may not provide a fundamental solution. Q-learning [24–26] can help mitigate the risk, but it is not expected to fully eliminate the issue either. This study develops the AICS, categorized under (I), through the redesign of the MRAC system based on first-order Lyapunov stability, as it incorporates customized neural networks to integrate and replace subsystems in the MRAC system.

In intelligent control system design, domain knowledge [2] is essential for the following: (a) plant modelling; (b) design of a controller with adaptive parameters; (c) adaptation of a control system to a changing environment; (d) acquisition of new design objectives and constraints; (e) stability verification of a proposed control system. The AICS consists of two

main subsystems: a meta-learning-triggered mechanism-based physics-informed neural network (MLTM-PINN) for online plant identification and a self-tuning neural network controller (STNNC). To address (a), this study designs the MLTM-PINN based on the findings from an earlier study [1], integrating several innovative techniques: each PINN trained online through self-supervised learning without the need for labelled data; a meta-learning process activated by a triggered mechanism for consistent fine adaptation; transfer learning combined with meta-learning for finely customized initialization and quick adaptation to input changes. Raissi et al. [27] made significant advancements in PINNs. Their success has motivated many researchers to explore similar approaches [28–35] across various branches of applied mathematics, science, and engineering. Robinson et al. [35] attempted a new method for integrating domain knowledge during training. However, they were unable to generalize their integration method due to the different modelling nature of benchmark problems. This study instead uses regularization, which streamlines the structure of PINNs. When it comes to meta-learning, Hochreiter et al. [36] and Younger et al. [37] signified a pivotal moment. A gradient-based model-agnostic meta-learning (MAML) model [38] used in this study marked a notable advancement in learning algorithms with multiple inner learners and a single outer learner for further adjustment. Since then, various ideas [39–41] based on the MAML, aimed at enhancing meta-learning algorithms, have emerged. The MAMLs offer key advantages over metric- and model-based approaches. It is model-agnostic and applicable to any differentiable model for tasks in supervised, self-supervised, and reinforcement learning. Unlike metric- and model-based methods, they support direct parameter adaptation via gradient updates without relying on task-specific model structures and prior knowledge of system dynamics, enhancing generalization, especially in few-shot or domain-shift settings. It integrates efficiently with standard optimization workflows and scales, making it both flexible and effective. To avoid a decline in control performance, some studies [42,43] opted to use labelled data for a PINN, even if it was minimal, and another study [44] addressed boundary value problems, inherently reducing the usage of labelled data. In designing control systems, meta-learning was tested for quick adaptation and/or finely tailored initialization [45–48]. Transfer learning facilitated computational speed-ups by initializing subnetworks [1,49]. The study [1] by Duanyai was the first attempt to integrate the MAML, PINNs, and transfer learning in control.

This study highlights (b) by designing the STNNC with online adaptive parameters. It functionally integrates the nonlinear controller and adaptation laws in the MRAC system. This integration streamlines the AICS structure while retaining high controllability and control efficiency. To evaluate and compare control performances, three distinct scenarios are explored, each employing a different STNNC design approach with either transfer learning, hyperparameter optimization (HPO), or a combination of both. Bayesian optimization (BO) is a type of black-box optimization [50] that operates without explicit knowledge of an objective function's internal structure. This study employs a Gaussian process tailored for BO (GP-BO), which is particularly efficient in low-dimensional hyperparameter spaces. Traditional optimization methods, such as grid search and random search, often struggle with highly nonlinear system dynamics. In contrast, GP-BO learns complex patterns and relationships in data, enabling more efficient navigation of a hyperparameter search space. It offers several advantages, including sample efficiency, adaptive search, uncertainty quantification, the intelligent balance between exploration and exploitation, applicability across continuous, discrete, and categorical hyperparameter spaces, and effectiveness in global optimization. The core idea of BO is to model an unknown objective function using a surrogate model or a response surface model, each of which is used to determine the next evaluation point. This strategy can be traced back to the work of Kushner [51]. The work by Zhilinskas [52] and Močkus [53] built upon the research, but the efficient global

optimization algorithm by Jones et al. [54] received greater attention. HPO in machine learning, contributed by Snoek et al. [55], has gained incredible popularity. Salemi et al. [56] and Mehdad and Kleijnen [57] advanced insight into GP-BO for system optimization, making it a practical tool for a wide range of applications. Booker et al. [58] and Regis and Shoemaker [59–61] explored surrogate methods. A study [59] noted that in gradient-based optimization (GDO), a well-known form of non-black-box optimization, derivatives are not always available, and finite-difference approximations can be too costly to perform. These shortcomings explain why GDO is not as popular as BO in HPO. GPs are commonly regarded as one of the representative surrogate models in BO [62–64]. A GP-BO process follows three steps: (1) definition of GP prior distribution; (2) acquisition of new sets of hyperparameters and the update of observed data relative to an underlying objective function; (3) update of GP posterior distribution. GP is fully characterized by a mean function and a covariance function. The choice of a kernel family was made manually in advance [65–67] or automatically [68]. Roman et al. [69] proposed adaptive kernel selection strategies for BO. In this study, a Matérn kernel with pre-tuned parameters [65] (see Equation (33)) is used. Acquisition functions include expected improvement (EI) [54,55,70], probability of improvement [70], upper confidence bound [55], Thompson sampling [71], and others. Močkus [53] introduced the fundamental concept of EI. This study employs EI (see Equation (34)) without requiring parameter tuning [55], effectively balancing exploration and exploitation.

In this study, one aspect of (d) involves the decision-making of the AICS in control, with a particular focus on the trade-off between high controllability and control efficiency. Scenario 2 identifies the best solution to the conflict. It enhances the predictive capability and the computational efficiency of the MLTM-PINN, providing qualitative feedback to the STNNC and receiving a high-quality input in return, as both subsystems seamlessly operate within the cohesive AICS. This exemplifies the best synergy achieved by the data-driven model in predicting the behaviour of the highly nonlinear plant and optimizing control actions. The series of our forthcoming studies (see Section 6) will address (c) adaptation to environmental changes using a denoising autoencoder and (e), which focuses on estimating the stability of a designed control system.

2. AICS Design Strategy and Motivation

This section introduces the design strategy of the AICS, which is derived from the traditional MRAC framework. Initially, we design the MRAC system based on first-order Lyapunov stability and then modify it to develop the AICS with two subsystems. This section tests the MRAC system to highlight its shortcomings and the motivation for proposing the AICS design.

2.1. First-Order Lyapunov Stability Analysis of the MRAC System for a Single SISO Plant

This section presents a rigorous mathematical model that underpins the theoretical framework for MRAC system design, followed by AICS design derived from the reconstruction of the MRAC system. We first examine the first-order Lyapunov-based nonlinear MRAC stability [1] for the adaptive control of a first-order plant with the initial condition $u_p^0 = 0$, described by the equation:

$$\dot{u}_p^t = -A_p \cdot u_p^t - C_p \cdot f_p^t + B_p \cdot u^t \tag{1}$$

here, $A_p = -1.0$, $B_p = 3.0$, $C_p = -1.0$, and $f_p^t = (u_p^t)^2$ are used for all examples throughout this study. $\dot{(\ )}$ implies the first derivative of $(\ )$ with respect to t, and $(\ )^t$ indicates the function of t. u^t is a control input. Online plant identification is implemented by a

differential equation (DE) solver (see Figure 1) when a governing differential equation is given by Equation (1). u_p^t represents an output from the plant. Let the desired output u_m^t of the reference model also be specified by the first-order differential equation as follows:

$$\dot{u}_m^t = -A_m \cdot u_m^t - C_m \cdot f_m^t + B_m \cdot r^t \tag{2}$$

where $A_m = 4.0 > 0, B_m = 4.0 > 0, C_m = 0.0$, and $f_m^t = 0.0$ indicate constants.

$$r^t = 4\, sin(3t) \tag{3}$$

is a reference signal. A control law in the form of

$$u^t = a_u^t \cdot u_p^t + a_r^t \cdot r^t + a_f^t \cdot f_p^t \tag{4}$$

is established to achieve adaptive control, incorporating the adaptive feedback gains, a_u^t, a_r^t, and a_f^t. This results in closed-loop dynamics that allow for systematic feedback adjustments. Let the tracking error be

$$e^t = u_p^t - u_m^t \tag{5}$$

and the error of parameter estimation be

$$\tilde{a}_u^t = a_u^t - a_u^* \tag{6a}$$

$$\tilde{a}_r^t = a_r^t - a_r^* \tag{6b}$$

$$\tilde{a}_f^t = a_f^t - a_f^* \tag{6c}$$

here, a_u^*, a_r^*, and a_f^* are constants. According to Barbalat's lemma, the following set of adaptation laws is identified as

$$\dot{\boldsymbol{a}}_u^t = -\gamma \cdot e^t \cdot u_p^t \tag{7a}$$

$$\dot{\boldsymbol{a}}_r^t = -\gamma \cdot e^t \cdot r^t \tag{7b}$$

$$\dot{\boldsymbol{a}}_f^t = -\gamma \cdot e^t \cdot f_p^t \tag{7c}$$

where γ implies an arbitrary positive constant, representing adaptation gain. To identify adaptive parameters for the nonlinear controller, these adaptive feedback gains are updated by

$$a_u^{t+\Delta t} = a_u^t + \dot{a}_u^t \cdot \Delta t \tag{8a}$$

$$a_r^{t+\Delta t} = a_r^t + \dot{a}_r^t \cdot \Delta t \tag{8b}$$

$$a_f^{t+\Delta t} = a_f^t + \dot{a}_f^t \cdot \Delta t \tag{8c}$$

which ensures the convergence of e^t. Therefore, the dynamics of the tracking error,

$$\dot{e}^t = -A_m e^t + B_p(\tilde{a}_u^t \cdot u_p^t + \tilde{a}_r^t \cdot r^t + \tilde{a}_f^t \cdot f_p^t) \tag{9}$$

can be found by subtracting Equation (2) from Equation (1). The first-order Lyapunov function for the plant is proposed by

$$V(e^t, \tilde{a}_u^t, \tilde{a}_r^t, \tilde{a}_f^t) = \frac{1}{2}(e^t)^2 + \frac{1}{2\gamma}|B_p|(\tilde{a}_u^{t\,2} + \tilde{a}_r^{t2} + \tilde{a}_f^{t\,2}) \geq 0 \tag{10}$$

of which the first derivative becomes

$$\dot{V}\left(e^t, \tilde{a}_u^t, \tilde{a}_r^t, \tilde{a}_f^t\right) = -A_m\left(e^t\right)^2 < 0. \tag{11}$$

Equation (10) supports the boundness of the signals e^t, $\tilde{a}_u^t$, $\tilde{a}_r^t$, and $\tilde{a}_f^t$, thereby ensuring the boundness of $\dot{e}^t$, and therefore, the uniform continuity of $\dot{V}\left(e^t, \tilde{a}_u^t, \tilde{a}_r^t, \tilde{a}_f^t\right)$. As a result, the MRAC system is globally stable, and the globally asymptotic convergence of e^t is achieved by Barbalat's lemma.

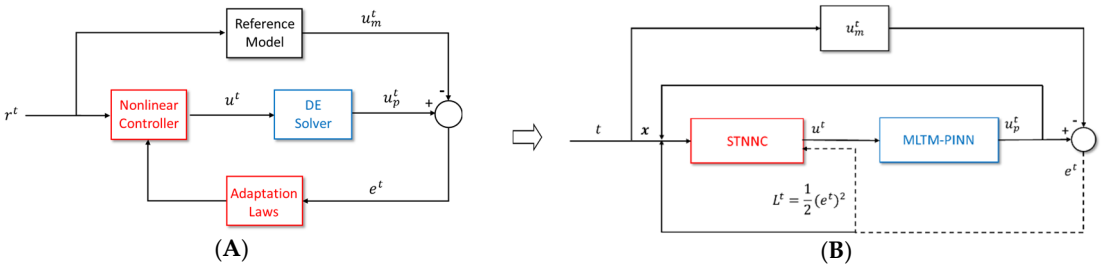


Figure 1. Structure modification from the MRAC system to the AICS. (**A**) MRAC system. (**B**) AICS.

2.2. AICS Design Strategy Derived from the MRAC Framework

We could pursue three directions to enhance the performance of the MRAC system: (i) analytically discovering a higher-order Lyapunov function to ensure tracking dynamics within the MRAC framework; (ii) developing a hybrid control algorithm by combining the MRAC system with other methods; (iii) modifying the MRAC system by using functionally customized neural networks (see Figure 1). The design approach proposed in this paper falls into the third category.

This section proposes a transition design strategy from the MRAC system to the AICS, focusing on structural simplification while maintaining desired performance. Figure 1 depicts the design of the AICS through the integration and reconstruction of the MRAC's subsystems.

$$L^t(\boldsymbol{\theta}, \boldsymbol{\lambda}; \boldsymbol{D}_{tr}^c) = \frac{1}{2}\left(e^t\right)^2 \tag{12}$$

is tracking loss, evaluated automatically. All online adaptive parameters in the STNNC are updated at each time t based on Equation (12), with no need for labelled data. $\boldsymbol{\theta}$ indicates a set of model parameters, and $\boldsymbol{\lambda}$ indicates a hyperparameter vector. $\boldsymbol{D}_{tr}^c$ denotes the auto-created training set of $\{t, u_m^t\}$ for the STNNC. An output u_p^t from the MLTM-PINN is evaluated at t, where $\boldsymbol{t} = \{t_0, \Delta t, 2\Delta t, \ldots, t, \ldots, T\}$ indicates an entire input time set with the current t, the initial time $t_0 = 0$ and the terminal time $T = 4.5$ for operating the AICS. $\Delta t = 0.01$, used in controlling both the MRAC system and the AICS, represents a time step. For running the AICS, all input samples of an automatically discretized sub-time span $\boldsymbol{t}_{sub}$ for t (see Figure 5) are sequentially input into the MLTM-PINN during each control cycle. $\boldsymbol{x} = \left\{u_p^t, e^t\right\}$ represents a state vector for a two-dimensional state space $\mathbb{R}^2$, excluding the remaining constant variables in Equation (7), and serves as a feedback input to ensure the closed-loop dynamics. An initial input set at $t = 0$ is defined by $\boldsymbol{x}^0 = \{u_p^0, e^0\}$ for all scenarios.

To design the AICS by reorganizing the MRAC subsystems, the adaptation laws and the adaptive feedback gains for the nonlinear controller are functionally integrated into the STNNC with multiple online adaptive parameters, without requiring extensive control

domain knowledge. The DE solver is upgraded to the MLTM-PINN. Additionally, the AICS directly sets the desired output u_m^t as given by

$$-\frac{48}{25}{}^{-4t}\left(-1+{}^{4t}\cdot cos(3t)-\frac{4}{3}\,{}^{4t}\cdot sin(3t)\right) \tag{13}$$

rather than deriving it from Equation (2). This study proposes three distinct design scenarios through the transition design strategy and examines the controllability (best control response) and the control efficiency (minimal activation frequency of meta-learning and a small number of trials for the HPO) of the AICS across these scenarios.

2.3. Motivation Behind the AICS Design

The MRAC system with two pre-tuned parameters γ and B_p is simulated for $t_0 = 0$ and $t_f = 6.0$ using the sixth order Runge–Kutta method as the DE solver. The plant and the reference models from Section 2.1 are used.

As shown in Figure 2, the subfigures display control responses for the B_p values of 1.5, 3.0, 6.0, and 12.0, with each graph depicting the system's response at the pre-tuned γ values ranging from 0.1 to 10.0. All system responses indicate suboptimal adaptation either in the initial phase or throughout the entire control process. Even increasing the value of γ to 10.0 fails to mitigate the initial fluctuations in all control responses across all B_p values. Figure 3 details a specific case with a wider range of γ at $B_p = 3.0$. As the value of γ increases to 72.0, the response diverges without any improvement in controllability, showing the initial phase of divergence in subfigure (A). The findings suggest that the MRAC system faces limitations in controlling highly nonlinear plants, possibly due to a lack of adaptive parameters. Nonetheless, the discovery of a higher-order Lyapunov function to ensure tracking dynamics for traditional MRAC system design remains a significant challenge.

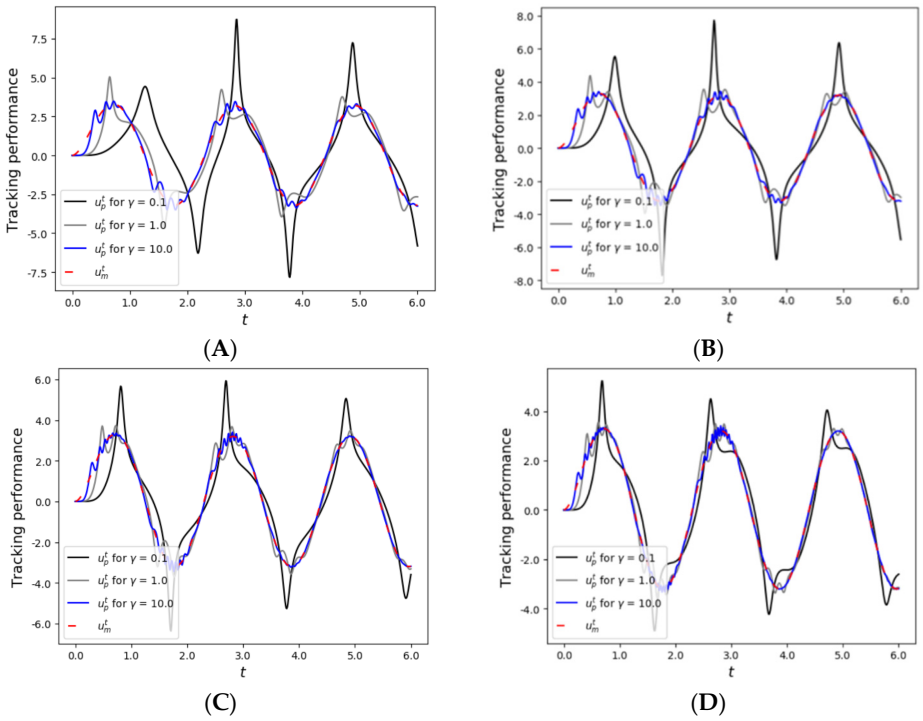
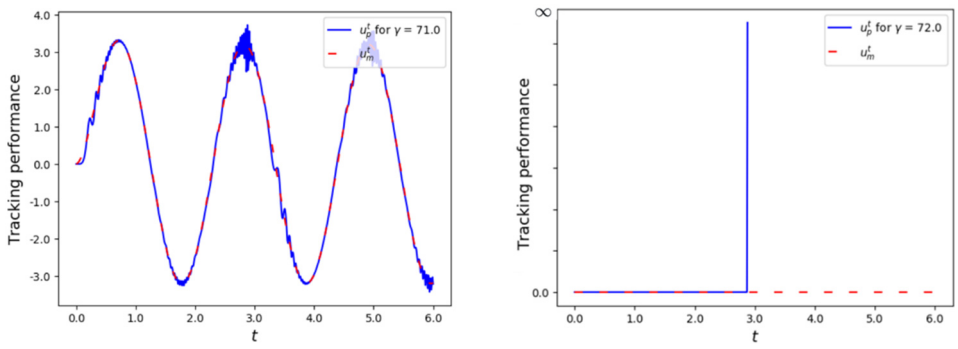


Figure 2. Controllability of the MRAC system for varying γ and B_p. (**A**) Control responses at $B_p = 1.5$. (**B**) Control responses at $B_p = 3.0$. (**C**) Control responses at $B_p = 6.0$. (**D**) Control responses at $B_p = 12.0$.



(**A**) (**B**)

Figure 3. A subsequent process of control response divergence at $B_p = 3.0$. (**A**) Initial phase of control response divergence at $\gamma = 71.0$. (**B**) Control response divergence at $\gamma = 72.0$.

3. AICS Design

This section provides a comprehensive introduction to the algorithms underlying the MLTM-PINN and the STNNC. The theoretical foundations and computational mechanisms of each component are discussed in detail to highlight their roles within the overall AICS framework. Particular emphasis is placed on how these algorithms contribute to achieving adaptive control and plant identification in dynamic environments.

3.1. MLP and PINN

Traditional machine learning algorithms often face challenges in data generation, which poses a significant obstacle in many scientific and engineering domains. PINNs can address this issue by integrating physical laws, typically expressed as differential equations, into a training process. This advantage allows PINNs to make more reliable predictions with minimal or even no data. In this study, a single PINN is employed to create each τ (task) in the MLTM-PINN. Figure 4 displays the PINN architecture that connects a multilayer perceptron (MLP) to physics laws.

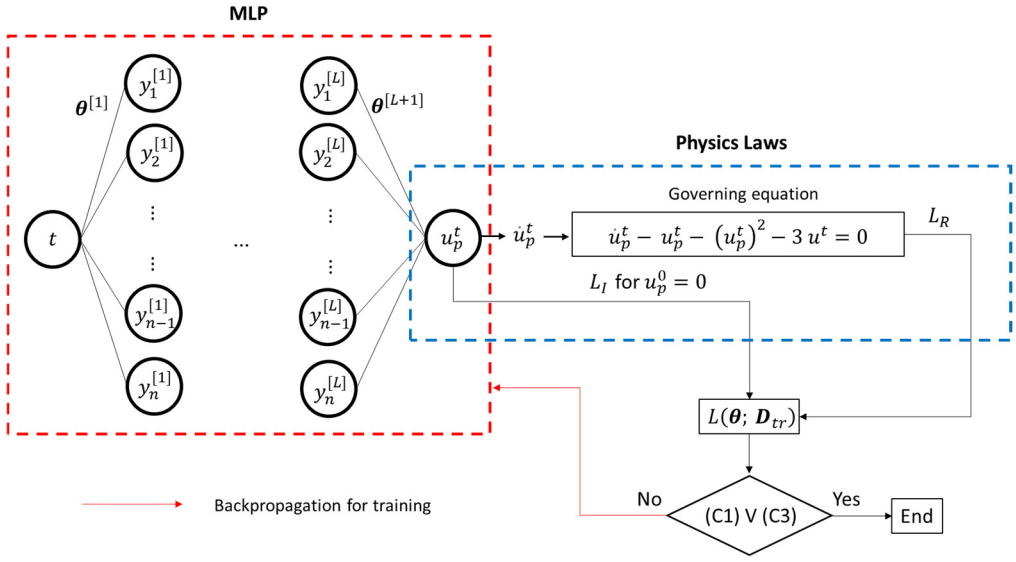


Figure 4. Architecture of the PINN with an MLP and a module integrating physics laws.

The PINN uses the MLP with L hidden layers, and their outputs can be represented by the following equation:

$$\boldsymbol{y}^{[l]} = \sigma^{[l]}\left(\boldsymbol{\theta}^{[l]} \cdot \boldsymbol{y}^{[l-1]} + \boldsymbol{b}^{[l]}\right), \text{ for } l = 1,\ 2,\ 3,\ \cdots,\ L \tag{14}$$

and here, $\sigma^{[l]}$ denotes an activation function in the l-th hidden layer. $\boldsymbol{y}^{[0]}$ is represented by t for $l = 1$. u_p^t is evaluated by

$$u_p^t = \boldsymbol{\theta}^{[L+1]} \cdot \boldsymbol{y}^{[L]}. \tag{15}$$

A set of optimal model parameters is achieved by implementing

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbb{R}^{\boldsymbol{m}}}{\arg\min}\, L(\boldsymbol{\theta}; \boldsymbol{D}_{tr}) \tag{16}$$

where $L(\boldsymbol{\theta}; \boldsymbol{D}_{tr})$ indicates total loss combining initial constraint loss L_I and residual loss L_R. A training dataset $\boldsymbol{D}_{tr}$ includes $\boldsymbol{t}_{sub}$ (see Figure 5) at t as an input set without labelled data for the MLTM-PINN. $(\cdot)^*$ indicates optimal $(\cdot)$. $\mathbb{R}^m$ means an m-dimensional real space. Each τ for the MLTM-PINN employs four hidden layers, each containing 256 NE (neurons). Unlike, the MLTM-PINN, the STNNC does not include the physical laws. An MLP with two inputs in an input layer, originally comprising two hidden layers, is used for the STNNC. The first hidden layer includes eight NE, while the second contains 256 NE when the HPO strategies are not involved. It is automatically trained by Equation (12) (see Figure 1). Initialization is performed by Variance Scaling for the MLTM-PINN with Swish activations, while the STNNC with Tanh activations uses an Xavier Uniform initializer. None of the layers are frozen during the training in applying transfer learning.

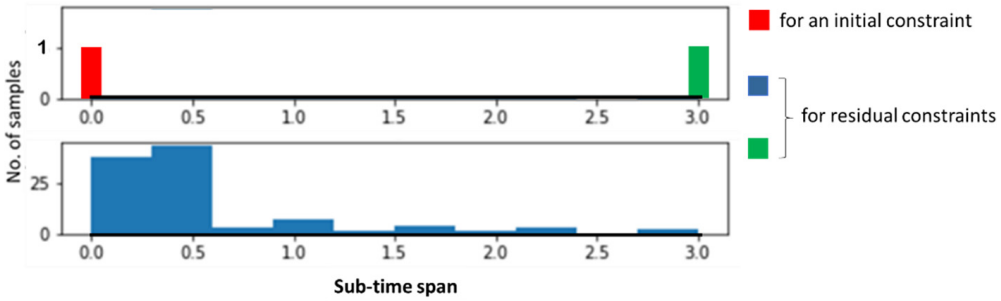


Figure 5. A discretized sub-time span at $t = 3.0$.

In our proposed framework, the target unstable plant is identified by the MLTM-PINN using a differential equation, which serves as one of physical laws. $f_{\boldsymbol{\theta}}^m(t)$ (a single meta-model) employing inner learners is trained to approximate the solution to this differential equation. To incorporate physical constraints, both L_I and L_R are used to enforce consistency with the physical laws.

3.2. MLTM-PINN Design

This section focuses on developing the MLTM-PINN for online plant identification. All τs in the subsystem adhere to a specific discretization rule for each $\boldsymbol{t}_{sub}$. Figure 5 depicts the sampling manner for this initial value problem (refer to Section 2) by randomly sampling an input sub-time span for $f_{\boldsymbol{\theta}}^m(t)$ with samples more closely concentrated around its initial constraint. It demonstrates a discretization case with a total of 100 samples at $t = 3.0$, including one sample at each end. All $\boldsymbol{t}_{sub}$s follow the same sampling manner, but the distribution of input samples for each τ varies throughout the entire control process.

As stated in a study [72], the combination of meta-learning and transfer learning provides finely tailored initial features to τs. This enables inner learners to develop more adaptable features, thereby improving the output accuracy of a single meta-model through iterative fine-tuning. Algorithm 1 outlines the bi-level learning process of the MAML-based meta-learning for the MLTM-PINN with ten inner learners at t, using a limited-memory Broyden–Fletcher–Goldfarb–Shanno with box constraints (L-BFGS-B) optimization strategy.

It is founded on the basic concept presented in a work [73] and the further evolved versions by studies [74,75]. The strategy is detailed in steps 1 to 7 below:

Step 1: set $k = 0$ and $ms = 50$.

Step 2: set $\boldsymbol{\theta}^{(lo)} \leq \boldsymbol{\theta}_k \leq \boldsymbol{\theta}^{(up)}$.

Step 3: compute $\boldsymbol{d}_k$ by using two-loop recursion

set $\boldsymbol{q} = \boldsymbol{g}_k$ (17)

define $P(\boldsymbol{\theta}_k) = min(max\left(\boldsymbol{\theta}^{(lo)}, \boldsymbol{\theta}_k\right), \boldsymbol{\theta}^{(up)})$ (18)

for c $= k$ *to* $max(0,\ k - ms)$ ***do***: this first loop iterates backwards

compute $\delta_c^{(1)} = \boldsymbol{\rho}_c \cdot \boldsymbol{s}_c^T \cdot \boldsymbol{q}$ (19a)

where

$$\boldsymbol{\rho}_c = \frac{1}{\boldsymbol{y}_c^T \cdot \boldsymbol{s}_c} \tag{19b}$$

update $\boldsymbol{q} \leftarrow \boldsymbol{q} - \delta_c^1 \cdot \boldsymbol{y}_c$ (20)

end for

set $\boldsymbol{r} = \boldsymbol{H}_k \cdot \boldsymbol{q}$ (21)

for c $= \ max(0,\ k - ms) tok$ ***do***: this second loop iterates forwards

compute $\delta_c^{(2)} = \boldsymbol{\rho}_c \cdot \boldsymbol{y}_c^T \cdot \boldsymbol{r}$ (22)

update $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{s}_c(\delta_c^{(1)} - \delta_c^{(2)})$ (23)

end for

such that $\boldsymbol{r} \approx \boldsymbol{H}_k \cdot \nabla L(\boldsymbol{\theta}_k; \boldsymbol{D}_{tr})$ for each inner learner or (24a)

$$\boldsymbol{r} \approx \boldsymbol{H}_k \cdot \nabla \sum L(\boldsymbol{\theta}_k; \boldsymbol{D}_{tr}) \text{ for the outer learner} \tag{24b}$$

set $\boldsymbol{d}_k = -\boldsymbol{r}$. (25)

Step 4: perform $\boldsymbol{\theta}_{k+1} = P(\boldsymbol{\theta}_k + \Delta_k \cdot \boldsymbol{d}_k)$ (26)

compute $\boldsymbol{g}_{k+1} = \nabla L(\boldsymbol{\theta}_{k+1};\ \boldsymbol{D}_{tr})$ for each inner learner or (27a)

$$\boldsymbol{g}_{k+1} = \nabla \sum L(\boldsymbol{\theta}_{k+1};\ \boldsymbol{D}_{tr}) \tag{27b}$$

for the outer learner, where a step size Δ_k is determined to satisfy Wolfe conditions without knowing $\boldsymbol{\theta}_{k+1}$.

Step 5: compute the memory $\boldsymbol{s}_{k+1} = \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$ and (28)

$$\boldsymbol{y}_{k+1} = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k \tag{29}$$

store the pairs $(\boldsymbol{s}_{k+1},\ \boldsymbol{y}_{k+1})$, then remove the oldest pair to keep only the most recent ms pairs.

Step 6: let $\widetilde{m} = \ min(k,\ ms - 1)$ and update Equation (30a) without storing its previous state.

$$\begin{aligned} \boldsymbol{H}_{k+1} \ &= (\boldsymbol{V}_k^T \cdot \cdots \cdot \boldsymbol{V}_{k-\widetilde{m}}^T) \boldsymbol{H}_0 (\boldsymbol{V}_{k-\widetilde{m}}^T \cdot \cdots \cdot \boldsymbol{V}_k^T) \\ &+ \ \boldsymbol{\rho}_{k-\widetilde{m}} (\boldsymbol{V}_k^T \cdot \cdots \cdot \boldsymbol{V}_{k-\widetilde{m}+1}^T)\ \boldsymbol{s}_{k-\widetilde{m}} \cdot \boldsymbol{s}_{k-\widetilde{m}}^T (\boldsymbol{V}_{k-\widetilde{m}+1}^T \cdot \cdots \cdot \boldsymbol{V}_k^T) \\ &+ \ \boldsymbol{\rho}_{k-\widetilde{m}+1} (\boldsymbol{V}_k^T \cdot \cdots \cdot \boldsymbol{V}_{k-\widetilde{m}+2}^T)\ \boldsymbol{s}_{k-\widetilde{m}+1} \cdot \boldsymbol{s}_{k-\widetilde{m}+1}^T (\boldsymbol{V}_{k-\widetilde{m}+2}^T \cdot \cdots \cdot \boldsymbol{V}_k^T) \\ &+ \ \vdots \\ &+ \ \boldsymbol{\rho}_k \cdot \boldsymbol{S}_k \cdot \boldsymbol{S}_k^T \end{aligned} \tag{30a}$$

where

$$\boldsymbol{V}_k = \boldsymbol{I} - \boldsymbol{\rho}_k \cdot \boldsymbol{y}_k \cdot \boldsymbol{s}_k^T \tag{30b}$$

Step 7: update $k \ \leftarrow k + 1$ and go to step 3

where k indicates an iteration counter, and ms is the memory size, indicating how many previous updates are stored and used in the algorithm. $\boldsymbol{\theta}^{(lo)}$ and $\boldsymbol{\theta}^{(up)}$ indicate lower and upper bounds, respectively. $P(\cdot)$ represents a projection function that keeps the iterations within the bounds. No bounds are applied to the L-BFGS-B in this study. $\delta_c^{(1)}$ refers to a step size for the backward process. $\boldsymbol{H}_k$ denotes the approximation of an inverse Hessian matrix. $\delta_c^{(2)}$ refers to a step size for the forward process. Equation (24a,b) are aligned with the goal of the L-BFGS-B algorithm to maintain $\boldsymbol{d}_k$ as the approximation of a search direction vector, which points toward the update direction of $\boldsymbol{\theta}_k$, incorporating curvature information. $\nabla L(\boldsymbol{\theta}_k;\ \boldsymbol{D}_{tr})$ and $\nabla \sum L(\boldsymbol{\theta}_k;\ \boldsymbol{D}_{tr})$ (or $\boldsymbol{g}_k$) denote gradients. $\boldsymbol{s}_k$ represents the difference

between the successive sets of model parameters, and $\boldsymbol{y}_k$ represents the difference between successive gradients. $\boldsymbol{I}$ refers to an identity matrix. $\boldsymbol{H}_0$ is the initial approximation of $\boldsymbol{H}_k$.

Algorithm 1 MLTM-PINN algorithm

input t
create $\boldsymbol{t}_{sub}$
initialize all required tensors and scalars
create and sample $f_{\boldsymbol{\theta}}^{m}(t)$, τ_{su}s and $\tau_{qu}s$
for $iter = 1$ to $iter^{max}$:
    an initial $\boldsymbol{\theta}_0$ is used for $iter = 1$
    initialize the first τ_{su} by using the updated $\boldsymbol{\theta}$ of $f_{\boldsymbol{\theta}}^{m}(t)$: transfer learning
    for each τ_{su} ***do***:
        perform step 1 to step 7 using Equation (27a) to update each τ_{su}
        if (C1) $\bigvee$ (C3):
            break
        initialize a following τ_{su} by using the updated $\boldsymbol{\theta}$ of the current τ_{su} (except for the last τ_{su}): transfer learning
    initialize all $\tau_{qu}s$ by using the $\boldsymbol{\theta}^*$ of each corresponding τ_{su} : transfer learning
    for all τ_{qu}s do:
        perform step 1 to step 7 using Equation (27b) to update $f_{\boldsymbol{\theta}}^{m}(t)$
        if (C1) $\bigvee$ (C2) $\bigvee$ (C3):
            break
        else:
            go to 5
evaluate $u_p^t = f_{\boldsymbol{\theta}^*}^{m}(t)$ at t

where τ_{su} implies a support task and τ_{qu} implies a query task. (C1) $iter = iter^{max}$, (C2) $\sum L(\boldsymbol{\theta}; \boldsymbol{D}_{tr}) < \varepsilon$, and (C3) $\|\boldsymbol{g}_k\| < \varepsilon^H$ are stopping criteria. $iter$ is an iteration counter. $iter^{max}$ is a maximum iterative number: 5000 is used for each inner learner, and 50 is used for the outer learner. $\varepsilon = 8.0 \times 10^{-4}$ and $\varepsilon^H = 2.22045 \times 10^{-9}$ indicate convergence tolerance and loss reduction tolerance, respectively. The number of line search steps per iteration is limited to 50.

3.3. STNNC Design

This section proposes three different scenarios for designing the STNNC. The foundational ideas of scenarios 1 and 2 are derived from a study by Duanyai [76]. Scenario 1 employs the transfer learning alone between neighbouring control cycles. Scenario 2, in addition to that, leverages the GP-BO (refer to Algorithm 2) to adjust a single hyperparameter α (learning rate). Lastly, scenario 3 optimizes both α and the number of NE in each hidden layer without applying the transfer learning. The GP-BO determines $\boldsymbol{\lambda}^*$ (set of optimal hyperparameters) that minimizes Equation (12) during the training, based on the HPO configurations presented in Table 1.

Algorithm 2 with β (balancing parameter) intelligently explores target regions within the hyperparameter search space that are likely to yield better control performance, while also considering potentially optimal solutions. This algorithm implements the GP-BO using a nested approach (refer to Equation (31)). The STNNC uses an Adam optimizer. $\mathbb{R}^n$ represents an n-dimensional real space.

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbb{R}^m}{\arg\min}\, L^T(\boldsymbol{\theta}, \boldsymbol{\lambda}^*; \boldsymbol{D}_{tr}^c) \text{ subjected to } \boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda} \in \mathbb{R}^n}{\arg\min}\, L^T(\boldsymbol{\theta}, \boldsymbol{\lambda}; \boldsymbol{D}_{tr}^c) \tag{31}$$

Here, the objective function $f(\boldsymbol{\lambda}) = f(\lambda_1, \lambda_2, \ldots, \lambda_n)$, defined by Equation (12) in the study, is modelled as a Gaussian process for the training set $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}^T$ in a hyperparameter space. λ_i indicates an individual hyperparameter, and n indicates the number of hyperparameters. $m(\boldsymbol{\lambda})$ denotes a mean function, and $K(\boldsymbol{\lambda}, \boldsymbol{\lambda}')$ denotes a covariance function, referred to as a Matérn kernel function. $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ imply different initial samples. In this study, a single initial input is considered to estimate the initial shape of $f(\boldsymbol{\lambda})$. A smoothness parameter $\nu = 2.5$ controls the smoothness of the function. $\Gamma(\cdot)$ refers to a gamma function. $K_\nu(\cdot)$ indicates a modified Bessel function. $ls = 1.0$ is a length scaling factor, controlling how quickly correlations decay over distance. EI identifies a new hyperparameter set $\boldsymbol{\lambda}^{new} = \left\{\lambda_1^{new}, \lambda_2^{new}, \ldots, \lambda_n^{new}\right\}^T$, where its value is maximized. $f(\boldsymbol{\lambda}_{best})$ serves as the best (minimal) function value at the currently best-known hyperparameter set $\boldsymbol{\lambda}_{best}$. $\beta = 2.6$ controls the trade-off between the exploration and the exploitation. $\boldsymbol{\Phi}(\cdot)$ denotes a cumulative distribution function, and $\phi(\cdot)$ denotes a probability density function. y represents the loss value defined in Equation (12). σ implies the standard deviation of noise in y. The GP-BO internally handles this parameter during the training. σ^2 indicates a parameter that is added to the diagonal of the kernel matrix as the expected amount of noise and influences the GP posterior distribution by adjusting its covariance structure. $\mathcal{N}$ denotes a normal distribution.

Table 1. HPO configurations for the scenarios.

	MTR	**Target Hyperparameters**								
		α			**NE (First Hidden Layer)**			**NE (Second Hidden Layer)**		
		Min. V	**Int. V**	**Max. V**	**Min. V**	**Int. V**	**Max. V**	**Min. V**	**Int. V**	**Max. V**
Scenario 1	–		0.01			8			256	
Scenario 2	5	0.001	0.001	0.02		8			256	
Scenario 3	20	0.001	0.001	0.02	8	256	2048	8	256	2048

MTR denotes the maximum number of trials for the HPO at each t. Min. V, Int. V, and Max. V denote, respectively, the lower bound value, the initial value, and the upper bound value of each target hyperparameter search space.

Algorithm 2 Bi-level optimization process for the GP-BO

1: initialize $u_p^0 = 0$ and $e^0 = 0$ as inputs
2: initialize $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$
3: **define GP prior distribution:**
define $f(\boldsymbol{\lambda}) \sim \mathcal{GP}(m(\boldsymbol{\lambda}), K(\boldsymbol{\lambda}, \boldsymbol{\lambda}'))$, (32)
where
$K(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}}{ls}|\boldsymbol{\lambda} - \boldsymbol{\lambda}'|\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{ls}|\boldsymbol{\lambda} - \boldsymbol{\lambda}'|\right)$ (33)
4: **acquires $\boldsymbol{\lambda}^{new}$ when the EI is maximized and evaluate y**
define $EI(\boldsymbol{\lambda}^{\boldsymbol{new}}) \equiv \mathbb{E}[max(f(\boldsymbol{\lambda}_{\boldsymbol{best}}) + \beta - f(\boldsymbol{\lambda}), 0)]$ (34)
$= \begin{cases} (f(\boldsymbol{\lambda}_{\boldsymbol{best}}) + \beta - m(\boldsymbol{\lambda}))\, \boldsymbol{\Phi}\left(\frac{f(\boldsymbol{\lambda}_{best}) + \beta - m(\boldsymbol{\lambda})}{\sigma}\right) + \sigma \cdot \phi\left(\frac{f(\boldsymbol{\lambda}_{best}) + \beta - m(\boldsymbol{\lambda})}{\sigma}\right) \text{ for } \sigma > 0 \\ 0 \text{for } \sigma = 0 \end{cases}$
5: ***if*** $f(\boldsymbol{\lambda}_{\boldsymbol{best}}) + \beta > f(\boldsymbol{\lambda})$: exploration
6: search for $\boldsymbol{\lambda}^{new}$ where the maximized EI is observed in a target hyperparameter search space
7: ***else*** : exploitation
8: intensively exploring $\boldsymbol{\lambda}^{new}$ in the vicinity of the best hyperparameter combination found so far to further improve $\boldsymbol{\lambda}_{\boldsymbol{best}}$
9: update $\boldsymbol{\theta}^{new} \leftarrow \boldsymbol{\theta} - \alpha \frac{\partial L(\boldsymbol{\theta}, \boldsymbol{\lambda}^{new})}{\partial \boldsymbol{\theta}}$ (35)

Algorithm 2 *Cont.*

10: evaluate u^t of the STNNC using $\boldsymbol{\lambda}^{new}$ and $\boldsymbol{\theta}^{new}$
11: evaluate e^t and y
12: **update GP posterior distribution**
evaluate $f(\boldsymbol{\lambda}^{new}) \mid y, \boldsymbol{\lambda}, \boldsymbol{\lambda}^{new} \sim \mathcal{N}(m'(\boldsymbol{\lambda}^{new}), K'(\boldsymbol{\lambda}^{\boldsymbol{new}}, \boldsymbol{\lambda}^{\boldsymbol{new}}))$ (36a)
where
$$m'(\boldsymbol{\lambda}^{new}) = K(\boldsymbol{\lambda}^{new}, \boldsymbol{\lambda})\big(K(\boldsymbol{\lambda}, \boldsymbol{\lambda}) + \sigma^2 \boldsymbol{I}\big)^{-1} y \tag{36b}$$
and
$$K'(\boldsymbol{\lambda}^{new}, \boldsymbol{\lambda}^{new}) = K(\boldsymbol{\lambda}^{new}, \boldsymbol{\lambda}^{new}) - K(\boldsymbol{\lambda}^{new}, \boldsymbol{\lambda})\big(K(\boldsymbol{\lambda}, \boldsymbol{\lambda}) + \sigma^2 \boldsymbol{I}\big)^{-1} K(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{\boldsymbol{new}}) \tag{36c}$$
13: update $f(\boldsymbol{\lambda}^{new}) \sim \mathcal{GP}(m'(\boldsymbol{\lambda}^{new}), K'(\boldsymbol{\lambda}^{new}, \boldsymbol{\lambda}^{new}))$ (37)
14: update $\boldsymbol{\lambda}_{\boldsymbol{best}} \leftarrow \boldsymbol{\lambda}^{new}$ and $\boldsymbol{\theta}_{\boldsymbol{best}} \leftarrow \boldsymbol{\theta}^{new}$ when y achieves the smallest value observed so far
15: ***if*** trial $<$ MTR :
16: go to 4
17: **else:**
18: ***break***
19: update $\boldsymbol{\lambda}^* \leftarrow \boldsymbol{\lambda}_{\boldsymbol{best}}$ and $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}_{\boldsymbol{best}}$
20: choose the best u^t computed using $\boldsymbol{\lambda}^*$ and $\boldsymbol{\theta}^*$

4. Simulation Results and Discussion

In this section, the three scenarios are evaluated regarding the controllability and the control efficiency of the AICS through rigorous simulations and analyses. The results of these evaluations will provide insights into the strengths and weaknesses of each scenario, guiding the further refinements of the proposed control system. The simulations are performed on Windows 11 with TensorFlow 2.10.0 running in Python 3.9.21.

Figures 6 and 7 visualize changes in the optimal values of the hyperparameters for scenarios 2 and 3. In Figure 6, scenario 2 illustrates that α fluctuates with several significant peaks, reaching the Max. V at the last two peaks, while the valleys attain the Min. V during the transitions. The upper bound appears adequate to cover all possible upper variations before the second-to-last peak, but the adequacy of the lower bound still remains uncertain. The transitions, marked by prolonged valleys, persist for extended durations rather than quickly rebounding. This could be interpreted as suggesting that the GP-BO algorithm continues to extensively explore the search space near the lower bound without being able to descend further, under the conditions of a single initial input point and MTR $= 5$. Nevertheless, the outputs in Figure 12 show that adjusting these three factors (the bounds, the number of initial samples, and the MTR) for the line search space is unnecessary. As depicted in Figure 7, the optimal hyperparameter values for scenario 3 wander with pronounced oscillations and no clear patterns, failing to gather sufficient data to accurately predict optimal values within the search space, which is constrained by inadequate bounds. Unlike in the 1D optimization case, a single initial input point in the 2D optimization may provide limited information about the objective function's behaviour across the search space, thereby hindering the GP-BO from constructing an effective surrogate model from the outset. In addition, setting the MTR to 20 may also reduce the likelihood of finding better solutions. To observe discernible patterns in the variations, it is advisable to adjust the three factors by expanding the size of the hyperparameter search space, permitting more opportunities to explore it depending on its scale, and considering multiple initial samples. However, this solution is likely inefficient, as it requires additional computational resources, despite employing the fourfold larger MTR in scenario 3 compared to scenario 2

(see Table 1). It is easily expected that using multiple initial samples will lead to a longer search duration during the initial phase of the optimization.

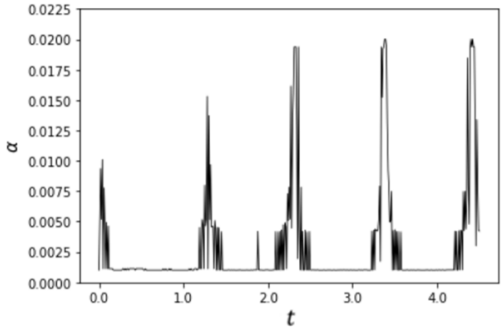


Figure 6. Change in the optimal value of α over t in scenario 2.

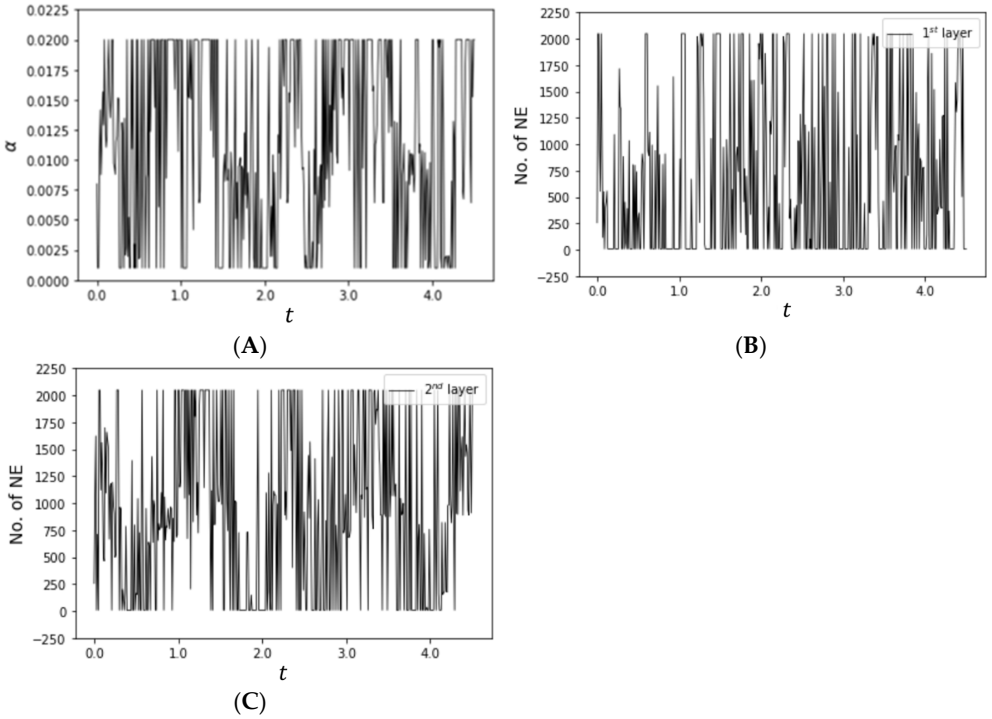


Figure 7. Changes in the optimal values of the target hyperparameters over t in scenario 3. (**A**) Change in the optimal value of α. (**B**) Change in the optimal number of NE in the first hidden layer. (**C**) Change in the optimal number of NE in the second hidden layer.

In Figure 8, unlike the other scenarios, scenario 1 has a different profile for u^t that sets it apart. The distinct overall shape, along with the unstable portion observed at the end of the control process, reflects the inability of the STNNC to optimally compensate for the system's dynamics, leading to the inferior tracking performance, as shown in Figures 11–13. In scenario 2, an increase in the micro-scale fluctuations, characterized by the formation of sharp spikes and drops, is observed after $t = 0.8$. These fluctuations are more obvious in scenario 3, which involves optimizing a greater number of hyperparameters. These findings suggest that the dynamic evolution of the u^t inputs to the MLTM-PINN may contribute to its delayed adaptation (see Figure 10B,C).

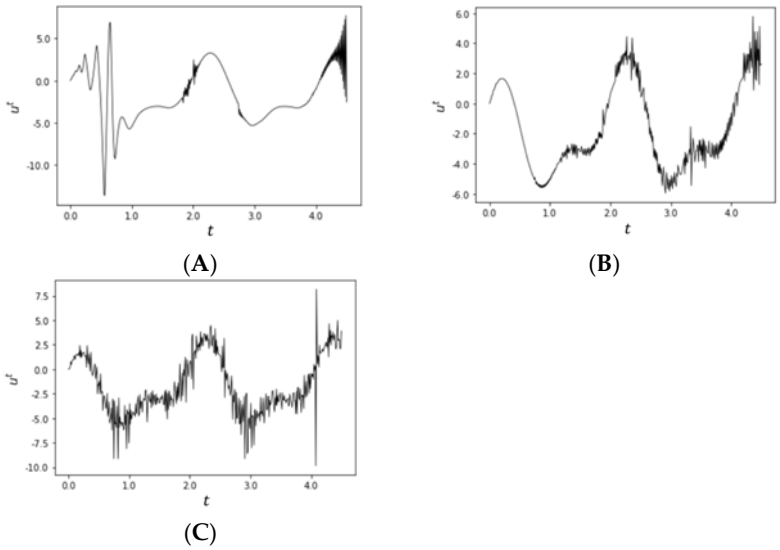


Figure 8. Changes in u^t over t in the scenarios. (**A**) Scenario 1. (**B**) Scenario 2. (**C**) Scenario 3.

Figures 9–13 illustrate changes in the performance of the MLTM-PINN and the AICS according to the distinct STNNC design scenarios. Figure 9 displays triggering event profiles across the scenarios. Scenario 2 triggers meta-learning the fewest number of times, totalling only 34 activations and indicating an advantage in minimizing computational demands. The findings from subfigures (B) and (C) highlight that achieving the high-quality tracking performances (see Figures 11–13) necessitates activating it concentratedly after the midpoint of the control process. Figure 10 shows the training frequency of $f_{\theta}^{m}(t)$ for each scenario. As evidenced by the subfigures, a quick adaptation effect is observed in scenario 1 after the midpoint, despite its small magnitude. In contrast, no significant effect is observed in the other scenarios involving the HPO. Optimal hyperparameter settings in the STNNC may require further training for the MLTM-PINN to converge, as it slowly adapts to u^ts with the dynamic input evolutions, which could alter the magnitude and direction of gradients in τs for the MLTM-PINN in a manner different from scenario 1. According to the hypothesis, subfigure (C) in Figure 8 can also elucidate why subfigure (C) in Figure 10 illustrates the extended training duration in scenario 3. The observations point to the fact that the MLTM-PINN's quick adaptation can be challenging when employing the HPO strategies for the STNNC design.

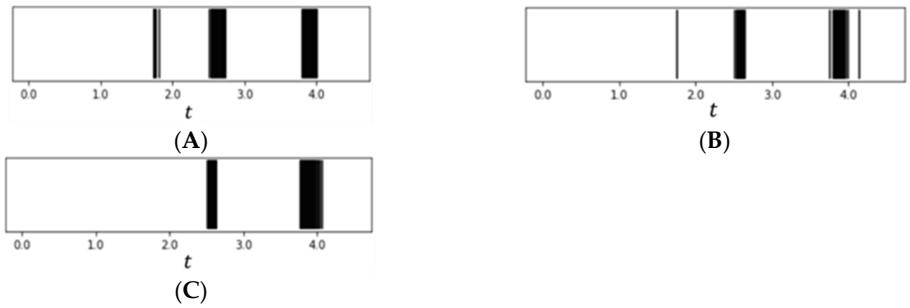


Figure 9. Scenarios with distinct triggering event profiles. (**A**) Scenario 1 with 49 activations. (**B**) Scenario 2 with 34 activations. (**C**) Scenario 3 with 40 activations.

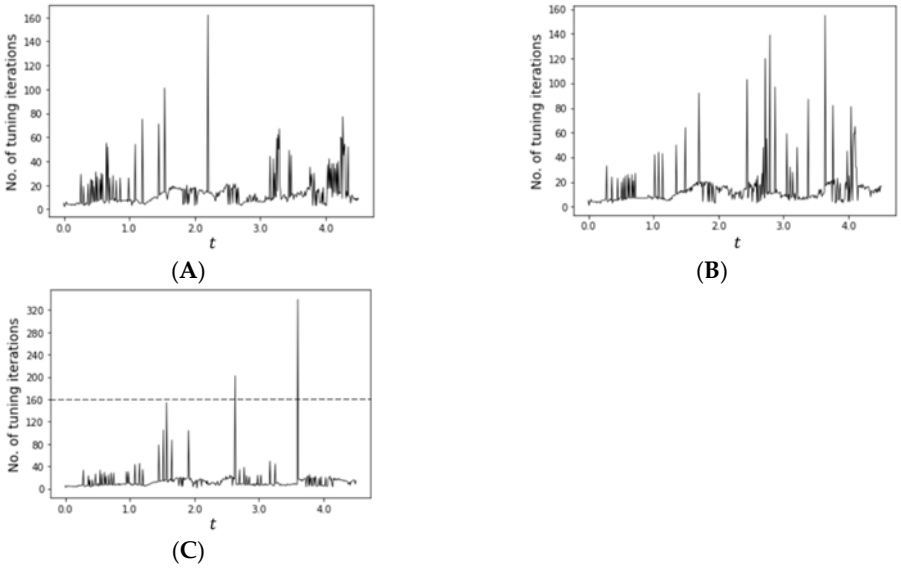


Figure 10. Changes in the training frequency of $f_\theta^m(t)$ across the scenarios. (**A**) Training frequency observed in scenario 1. (**B**) Training frequency observed in scenario 2. (**C**) Training frequency observed in scenario 3.

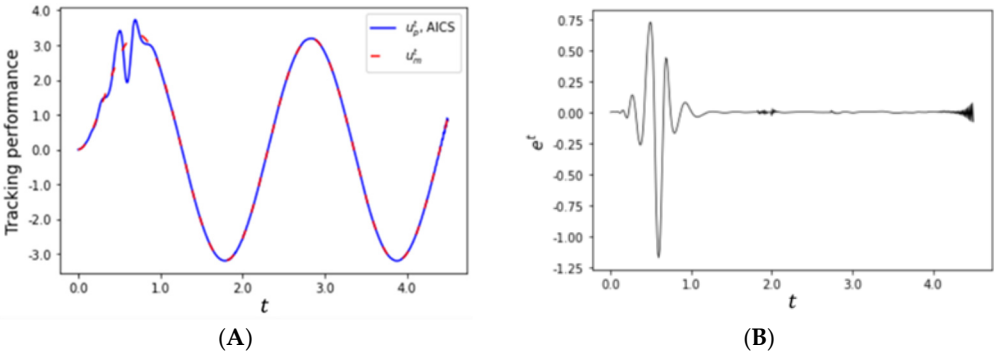


Figure 11. Controllability of the AICS designed by scenario 1. (**A**) Control response. (**B**) Tracking error.

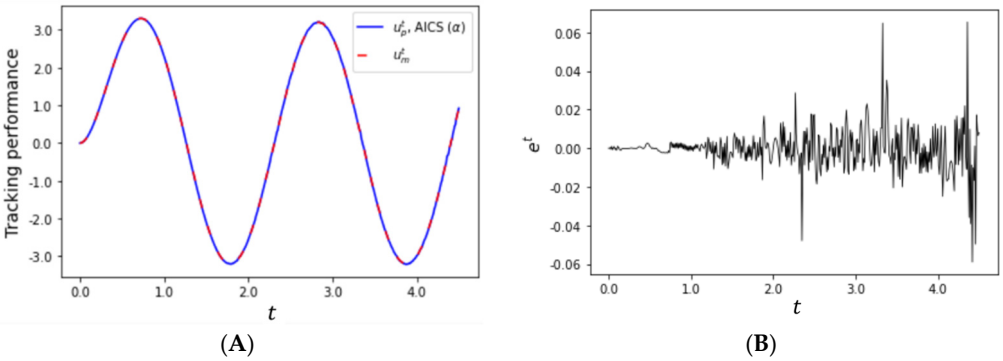
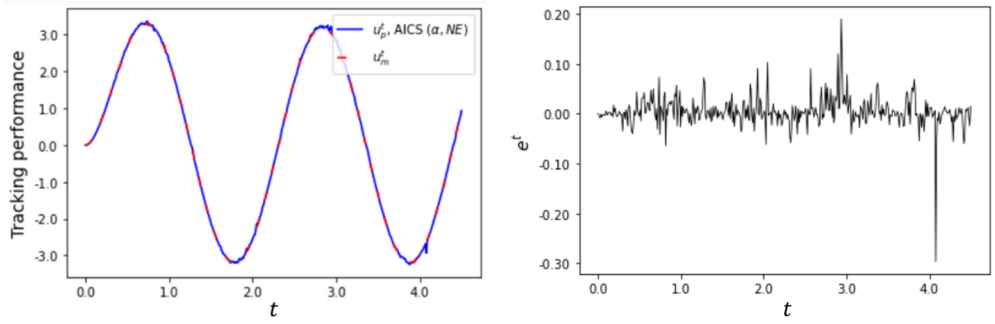


Figure 12. Controllability of the AICS designed by scenario 2. (**A**) Control response. (**B**) Tracking error.

Figure 11 illustrates the control response [76] and the tracking error of the AICS designed in scenario 1. Subfigure (A) initially exhibits fluctuations in the control response, which stabilize as t progresses. The notably different shape of u^t in the initial control process (see Figure 8A) may contribute to these fluctuations, suggesting that the STNNC without incorporating the HPO lacks the self-tuning capacity to mitigate them. The best control response, as shown in Figure 12 [76], is achieved in scenario 2 with 2048 model parameters only in the STNNC, using the transfer learning and optimizing α. e^t exhibits fluctuations

but converges to a range within one-tenth of that observed in scenario 1. Prior to $t = 3.0$, as shown in Figure 6, the upper bound (see Table 1) sufficiently supports the effective optimization of α. However, after this time point, it may begin to constrain the effectiveness of the optimization, as variation in α becomes restricted within an imprecisely defined range by both bounds. Nevertheless, the findings underscore that optimizing α without adjusting the size of the given search space, combined with the transfer learning, achieves the best controllability of the AICS. In scenario 3, despite exploring the hyperparameter search space four times more extensively than in scenario 2 (see Table 1) during the GP-BO process, subfigure (A) in Figure 13 displays the noisy control response. To eliminate the outliers in e^t, any solution—such as all possible STNNC-related (e.g., adding HPO options including initial samples or adjusting the architecture of the MLP) and MLTM-PINN-related strategies (e.g., increasing the activation frequency)—must require an increase in computational resources. Figures 9 and 10 indicate that scenario 2 is also superior to scenario 3 in terms of the computational efficiency, with less frequent meta-learning activation and quicker adaptation in control.



(**A**) (**B**)

Figure 13. Controllability of the AICS designed by scenario 3. (**A**) Control response. (**B**) Tracking error.

5. Conclusions

We propose the AICS with a learning-while-controlling capability for the highly nonlinear SISO plant. The control system consists of two subsystems, streamlining its structure: the MLTM-PINN for plant identification and the self-tuning STNNC with high adaptability. The AICS design involves revising the MRAC framework, integrating the nonlinear controller and adaptation laws into the STNNC based on the GP-BO strategy, and upgrading the DE solver to the MLTM-PINN.

The MLTM-PINN is devised with several essential techniques. The MLTM can intelligently resolve the conflict between high controllability and control efficiency, as the triggered mechanism activates the meta-learning only when one of the error thresholds detects a deterioration in the controllability. The quick adaptation in the MLTM-PINN, expected through the transfer learning and meta-learning, is restrictive when the HPO strategies are involved with the STNNC design. The PINN, trained through self-supervised learning without the need for labelled data, facilitates the online integration of learning and control modes.

For designing the STNNC, three distinct scenarios are tested. Scenario 1 shows the poorest tracking performance, particularly in the initial control response, due to the exclusion of the HPO, although it enables $f_{\theta}^{m}(t)$ to quickly adapt to u^t. Scenario 2 demonstrates that the combination of the transfer learning and the HPO in the STNNC effectively corrects the shape of u^t, leading to superior controllability and the highest control efficiency. However, Scenarios 2 and 3 leave the challenge of quick adaptation in the MLTM-PINN.

The study demonstrates the feasibility for challenging real-world applications in scenario 2. Yet, it still needs the resource-intensive meta-learning-based approach. Our upcoming study will provide solutions to this issue by the use of more adaptable neural networks for each subsystem.

6. Future Study

We plan to tackle the challenge of further reducing computational resources without compromising control performance by replacing the MLP with either a liquid neural network (LNN), a transformer neural network (TNN) with a single attention mechanism, or a combination of both, with and without the meta-learning. This replacement aims to preserve the high adaptability of the PINN by leveraging the key strengths of the neural networks. For the LNN, we anticipate higher adaptability to the dynamic input evolution. This enhanced adaptability is attributed to the use of ODE-based activations, which allow both subsystems to more effectively model continuous-time dynamics under rapidly changing inputs. In the case of the TNN, the self-attention mechanism is anticipated to sustain the influence of an initial constraint throughout the entire control process. Ultimately, combining physics-informed machine learning with a TNN could globally propagate the long-range dependency between initial constraints and outputs across a long time span. In addition, fine-tuning target continuous hyperparameters using hybrid optimization is also expected to enhance the adaptability of the STNNC and mitigate the dynamic input evolution, thereby reducing the frequency of the meta-learning activations.

In the final stage of control system design, stability verification is essential to ensure that the proposed system maintains bounded behaviour over time under both nominal and perturbed conditions. We will offer a numerical framework for assessing whether the system's states converge to a desired equilibrium point or remain within an acceptable region of attraction, using a Lyapunov function based on a partial differential equation.

Author Contributions: Conceptualization, W.K.S.; funding acquisition, M.-H.K.; investigation, W.D., W.K.S. and D.-W.L.; methodology, W.K.S.; project administration, M.-H.K.; resources, D.-W.L.; software, W.D. and S.D.; supervision, W.K.S. and M.-H.K.; validation, W.D. and W.K.S.; writing—original draft, W.D. and W.K.S.; writing—review and editing, W.K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by a grant from the National Research Foundation of Korea (NRF) supported by the Korea government (MSIT) under grant number 2021R1A2C2006025. The APC was funded by the NRF grant.

Data Availability Statement: Dataset available on request from the authors.

Acknowledgments: The authors gratefully acknowledge the financial support provided by the National Research Foundation of Korea (NRF) and the Korea government (MSIT) through grant number 2021R1A2C2006025.

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.

References

1. Duanyai, W.; Song, W.K.; Konghuayrob, P.; Parnichkun, M. Event-triggered model reference adaptive control system design for SISO plants using meta-learning-based physics-informed neural networks without labeled data and transfer learning. *Int. J. Adapt. Control Signal Process.* **2024**, *38*, 1442–1456. [CrossRef]
2. Antsaklis, P.J. *Intelligent Control. Wiley Encyclopedia of Electrical and Electronics Engineering*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1999; pp. 493–505.

3. Jagannathan, S.; Lewis, F.L.T. Identification of nonlinear dynamical systems using multilayered neural networks. *Automatica* **1996**, *32*, 1707–1712. [CrossRef]
4. Kwan, C.M.; Lewis, F.L.; Dawson, D.M. Robust neural-network control of rigid-link electrically driven robots. *IEEE Trans. Neural Netw.* **1998**, *9*, 581–588. [CrossRef] [PubMed]
5. Lewis, F.L.; Liu, K.; Yesildirek, A. Neural net robot controller with guaranteed tracking performance. *IEEE Trans. Neural Netw.* **1995**, *6*, 703–715. [CrossRef] [PubMed]
6. Lewis, F.L.; Yesildirek, A.; Kai, L. Multilayer neural-net robot controller with guaranteed tracking performance. *IEEE Trans. Neural Netw.* **1996**, *7*, 388–399. [CrossRef]
7. Yeşildirek, A.; Lewis, F.L. Feedback linearization using neural networks. *Automatica* **1995**, *31*, 1659–1664. [CrossRef]
8. Yu, S.-H.; Annaswamy, A.M. Stable neural controllers for nonlinear dynamic systems. *Automatica* **1998**, *34*, 641–650. [CrossRef]
9. Gu, F.; Yin, H.; Ghaoui, L.E.; Arcak, M.; Seiler, P.J.; Jin, M. *Recurrent Neural Network Controllers Synthesis with Stability Guarantees for Partially Observed Systems*; Association for the Advancement of Artificial Intelligence (AAAI): Washington, DC, USA, 2022; pp. 5385–5394.
10. Widrow, B.; Walach, E. *Adaptive Inverse Control: A Signal Processing Approach*; Wiley-IEEE Press: Hoboken, NJ, USA, 2007.
11. Nahas, E.P.; Henson, M.A.; Seborg, D.E. Nonlinear internal model control strategy for neural network models. *Comput. Chem. Eng.* **1992**, *16*, 1039–1057. [CrossRef]
12. Hunt, K.J.; Sbarbaro, D.; Żbikowski, R.; Gawthrop, P.J. Neural networks for control systems—A survey. *Automatica* **1992**, *28*, 1083–1112. [CrossRef]
13. Narendra, K.S.; Parthasarathy, K. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Netw.* **1990**, *1*, 4–27. [CrossRef]
14. Wang, D.; Liu, D.; Zhang, Q.; Zhao, D. Data-based adaptive critic designs for nonlinear robust optimal control with uncertain dynamics. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *46*, 1544–1555. [CrossRef]
15. Jin, L.; Nikiforuk, P.N.; Gupta, M.M. Direct adaptive output tracking control using multilayered neural networks. *J. Eng. Technol.* **1993**, *140*, 393–398. [CrossRef]
16. Jin, L.; Nikiforuk, P.N.; Gupta, M.M. Fast neural learning and control of discrete-time nonlinear systems. *IEEE Trans. Syst. Man Cybern.* **1995**, *25*, 478–488. [CrossRef]
17. Levin, A.U.; Narenda, K.S. Control of nonlinear dynamical systems using neural networks. II. Observability, identification, and control. *IEEE Trans. Neural Netw.* **1996**, *7*, 40–42. [CrossRef]
18. Sutton, R.S.; Barto, A.G. *Reinforcement Learning, an Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 1998.
19. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
20. Shen, Y.; Tobia, M.J.; Sommer, T.; Obermayer, K. Risk-sensitive reinforcement learning. *Neural Comput* **2014**, *26*, 1298–1328. [CrossRef]
21. Chen, H.; Liu, C. Safe and sample-efficient reinforcement learning for clustered dynamic environments. *IEEE Control Syst. Lett.* **2022**, *6*, 1928–1933. [CrossRef]
22. Cheng, Y.; Zhao, P.; Hovakimyan, N. Safe and efficient reinforcement learning using disturbance-observer-based control barrier functions. In Proceedings of the 5th Annual Learning for Dynamics and Control Conference, Philadelphia, PA, USA, 14–16 June 2023; PMLR 211: New York, NY, USA, 2023; pp. 104–115. Available online: https://proceedings.mlr.press/v211/cheng23a.html (accessed on 3 August 2025).
23. Wachi, A.; Hashimoto, W.; Shen, X.; Hashimoto, K. Safe exploration in reinforcement learning: A generalized formulation and algorithms. In Proceedings of the 37th International Conference on Neural Information Processing Systems, New Orleans, LA, USA, 2024.
24. Du, H.; Hao, B.; Zhao, J.; Zhang, J.; Wang, Q.; Yuan, Q. A path planning approach for mobile robots using short and safe Q-learning. *PLoS ONE* **2022**, *17*, e0275100. [CrossRef]
25. Xu, H.; Zhan, X.; Zhu, X. *Constraints Penalized Q-Learning for Safe Offline Reinforcement Learning*; Association for the Advancement of Artificial Intelligence (AAAI): Washington, DC, USA, 2022; pp. 8753–8760.
26. Ge, Y.; Zhu, F.; Ling, X.; Liu, Q. Safe Q-learning method based on constrained markov decision processes. *IEEE Access* **2019**, *7*, 165007–165017. [CrossRef]
27. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. 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]
28. Dong, S.; Li, Z. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Comput. Methods Appl. Mech. Eng.* **2021**, *387*, 114129. [CrossRef]
29. Jagtap, A.D.; Kharazmi, E.; Karniadakis, G.E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **2020**, *365*, 113028. [CrossRef]
30. Zhang, R.; Liu, Y.; Sun, H. Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling. *Eng. Struct.* **2020**, *215*, 110704. [CrossRef]

31. Zhu, Y.; Zabaras, N.; Koutsourelakis, P.-S.; Perdikaris, P. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *J. Comput. Phys.* **2019**, *394*, 56–81. [CrossRef]
32. Sun, L.; Wang, J.-X. Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theor. Appl. Mech. Lett.* **2020**, *10*, 161–169. [CrossRef]
33. Ma, Y.; Xu, X.; Yan, S.; Ren, Z. A Preliminary study on the resolution of electro-thermal multi-physics coupling problem using physics-informed neural network (PINN). *Algorithms* **2022**, *15*, 53. [CrossRef]
34. Amini Niaki, S.; Haghighat, E.; Campbell, T.; Poursartip, A.; Vaziri, R. Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. *Comput. Methods Appl. Mech. Eng.* **2021**, *384*, 113959. [CrossRef]
35. Robinson, H.; Pawar, S.; Rasheed, A.; San, O. Physics guided neural networks for modelling of non-linear dynamics. *Neural Netw.* **2022**, *154*, 333–345. [CrossRef]
36. Hochreiter, S.; Younger, A.S.; Conwell, P.R. Learning to learn using gradient descent. In Proceeding of the Artificial Neural Networks-ICANN 2001, Vienna, Austria, 21–25 August 2001.
37. Younger, A.S.; Hochreiter, S.; Conwell, P.R. *Meta-Learning with Backpropagation*; IEEE: Washington, DC, USA, 2001; pp. 2001–2006.
38. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017.
39. Liu, X.; Zhang, X.; Peng, W.; Zhou, W.; Yao, W. A novel meta-learning initialization method for physics-informed neural networks. *Neural Comput. Appl.* **2022**, *34*, 14511–14534. [CrossRef]
40. Psaros, A.F.; Kawaguchi, K.; Karniadakis, G.E. Meta-learning PINN loss functions. *J. Comput. Phys.* **2022**, *458*, 111121. [CrossRef]
41. Penwarden, M.; Zhe, S.; Narayan, A.; Kirby, R.M. A metalearning approach for Physics-informed neural networks (PINNs): Application to parameterized PDEs. *J. Comput. Phys.* **2023**, *477*, 111912. [CrossRef]
42. Nicodemus, J.; Kneifl, J.; Fehr, J.; Unger, B. Physics-informed neural networks-based model predictive control for multi-link manipulators. *IFAC-PapersOnline* **2022**, *55*, 331–336. [CrossRef]
43. Gokhale, G.; Claessens, B.; Develder, C. Physics informed neural networks for control oriented thermal modeling of buildings. *Appl. Energy* **2022**, *314*, 118852. [CrossRef]
44. García-Cervera, C.J.; Kessler, M.; Periago, F. Control of partial differential equations via physics-informed neural networks. *J. Optim. Theory Appl.* **2022**, *196*, 391–414. [CrossRef]
45. Shi, G.; Azizzadenesheli, K.; O'Connell, M.; Chung, S.-J.; Yue, Y. Meta-adaptive nonlinear control: Theory and algorithms. In Proceedings of the 35th International Conference on Neural Information Processing Systems 34, Vancouver, BC, Canada, 6–14 December 2021; Available online: https://proceedings.neurips.cc/paper_files/paper/2021/hash/52fc2aee802efbad698503d28ebd3a1f-Abstract.html (accessed on 3 August 2025).
46. Richards, S.M.; Azizan, N.; Slotine, J.-J.; Pavone, M. Adaptive-control-oriented meta-learning for nonlinear systems. Robotics: Science and Systems. *arXiv* **2021**, arXiv:2103.04490. [CrossRef]
47. Nagabandi, A.; Clavera, I.; Liu, S.; Fearing, R.S.; Abbeel, P.; Levine, S.; Finn, C. Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning. In Proceedings of the 2nd Workshop on Meta-Learning, Montreal, QC, Canada, 8 December 2018. [CrossRef]
48. Belkhale, S.; Li, R.; Kahn, G.; McAllister, R.; Calandra, R.; Levine, S. Model-based meta-reinforcement learning for flight with suspended payloads. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1471–1478. [CrossRef]
49. Penwarden, M.; Jagtap, A.D.; Zhe, S.; Karniadakis, G.E.; Kirby, R.M. A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions. *J. Comput. Phys.* **2023**, *493*, 112464. [CrossRef]
50. Masood, A. *Automated Machine Learning: Hyperparameter Optimization, Neural Architecture Search, and Algorithm Selection with Cloud Platforms*; Packt Publishing Limited: Birmingham, UK, 2023.
51. Kushner, H.J. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *J. Fluids Eng.* **1964**, *86*, 97–106. [CrossRef]
52. Zhilinskas, A.G. Single-step Bayesian search method for an extremum of functions of a single variable. *J. Cybermetics Syst. Anal.* **1975**, *11*, 160–166. [CrossRef]
53. Močkus, J. *On Bayesian Methods for Seeking the Extremum*; Springer: Berlin/Heidelberg, Germany, 1975; pp. 400–404.
54. Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **1998**, *13*, 455–492. [CrossRef]
55. Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. In Proceedings of the Advances in Neural Information Processing Systems 25, Lake Tahoe, NV, USA, 3–6 December 2012; Available online: https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html (accessed on 3 August 2025).
56. Salemi, P.; Nelson, B.L.; Staum, J. *Discrete Optimization via Simulation Using Gaussian Markov Random Fields*; IEEE: Georgia, GA, USA, 2014; pp. 3809–3820.

57. Mehdad, E.; Kleijnen, J.P.C. Efficient global optimisation for black-box simulation via sequential intrinsic Kriging. *J. Oper. Res. Soc.* **2018**, *69*, 1725–1737. [CrossRef]
58. Booker, A.J.; Dennis, J.J.E.; Frank, P.D.; Serafini, D.B.; Torczon, V.; Trosset, M.W. A rigorous framework for optimization of expensive functions by surrogates. *J. Struct. Multidiscip. Optim.* **1999**, *17*, 1–13. [CrossRef]
59. Regis, R.G.; Shoemaker, C.A. Constrained global optimization of expensive black box functions using radial basis functions. *J. Glob. Optim.* **2016**, *31*, 153–171. [CrossRef]
60. Regis, R.G.; Shoemaker, C.A. Improved strategies for radial basis function methods for global optimization. *J. Glob. Optim.* **2006**, *37*, 113–135. [CrossRef]
61. Regis, R.G.; Shoemaker, C.A. Parallel radial basis function methods for the global optimization of expensive functions. *Eur. J. Oper. Res.* **2007**, *182*, 514–535. [CrossRef]
62. Audet, C.; Denni, J.; Moore, D.; Booker, A.; Frank, P. A surrogate-model-based method for constrained optimization. In Proceedings of the 8th Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, USA, 6–8 September 2000.
63. Mahendran, N.; Wang, Z.; Hamze, F.; Freitas, N.D. Adaptive MCMC with Bayesian optimization. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research*; Canary Island, Spain, 21–23 April 2021. Available online: https://proceedings.mlr.press/v22/mahendran12.html (accessed on 3 August 2025).
64. Meeds, E.; Welling, M. GPS-ABC: Gaussian process surrogate approximate Bayesian computation. In Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, Quebec City, QC, Canada, 23–27 July 2014.
65. Genton, M.G. Classes of kernels for machine learning: A statistics perspective. *J. Mach. Learn. Res.* **2001**, *2*, 299–312.
66. Sollich, P.; Urry, M.; Coti, C. Kernels and learning curves for Gaussian process regression on random graphs. In Proceedings of the Advances in Neural Information Processing Systems 22, Vancouver, BC, Canada, 7–10 December 2009. Available online: https://proceedings.neurips.cc/paper/2009/hash/92cc227532d17e56e07902b254dfad10-Abstract.html (accessed on 3 August 2025).
67. Wilson, A.; Adams, R. Gaussian Process Kernels for Pattern Discovery and Extrapolation. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2021; PMLR: New York, NY, USA, 2013; Volume 28, pp. 1067–1075. Available online: https://proceedings.mlr.press/v28/wilson13.html (accessed on 3 August 2025).
68. Duvenaud, D.; Lloyd, J.; Grosse, R.; Tenenbaum, J.; Zoubin, G. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning, Proceedings of Machine Learning Research*; Atlanta, GA, USA, 16–21 June 2023, Available online: https://proceedings.mlr.press/v28/duvenaud13.html (accessed on 3 August 2025).
69. Roman, I.; Santana, R.; Mendiburu, A.; Lozano, J.A. An experimental study in adaptive kernel selection for Bayesian optimization. *IEEE Access* **2019**, *7*, 184294–184302. [CrossRef]
70. Couckuyt, I.; Deschrijver, D.; Dhaene, T. Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. *J. Glob. Optim.* **2013**, *60*, 575–594. [CrossRef]
71. Chapelle, O.; Li, L. An Empirical Evaluation of Thompson Sampling. Curran Associates. In Proceedings of the Advances in Neural Information Processing Systems 24, Granada, Spain, 12–17 December 2011. Available online: https://proceedings.neurips.cc/paper/2011/hash/e53a0a2978c28872a4505bdb51db06dc-Abstract.html (accessed on 3 August 2025).
72. Duanyai, W.; Song, W.K.; Chitthamlerd, T.; Kumar, G. Meta-learning-based physics-informed neural network: Numerical simulations of initial value problems of nonlinear dynamical systems without labeled data and correlation analyses. *J. Nonlinear Model. Anal.* **2024**, *6*, 485–513. [CrossRef]
73. Nocedal, J. Updating quasi-Newton matrices with limited storage. *Math. Comput.* **1980**, *35*, 773–782. [CrossRef]
74. Byrd, R.H.; Nocedal, J.; Schnabel, R.B. Representations of quasi-Newton matrices and their use in limited memory methods. *Math. Program.* **1994**, *63*, 129–156. [CrossRef]
75. Liu, D.C.; Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **1989**, *45*, 503–528. [CrossRef]
76. Duanyai, W. *A Comparision of the Controllability of Model Reference Adaptive Control and Intelligent Control Systems for a Nonlinear SISO Plant*; King Mongkut Institute of Technology Ladkrabang: Bangkok, Thailand, 2024.