electronics

MDPI

Article

Tikhonov-Tuned Sliding Neural Network Decoupling Control for an Inverted Pendulum

Yi-Jen Mon

Department of Electronic Engineering, Ming-Chuan University, Guei-Shan District, Taoyuan City 333, Taiwan; yjm@mail.mcu.edu.tw

Abstract: This paper introduces the concept of intelligent control using Tikhonov regularization for nonlinear coupled systems. This research is driven by the increasing demand for advanced control techniques and aims to explore the impact of Tikhonov regularization on these systems. The primary objective is to determine the optimal regularization term and its integration with other control methods to enhance intelligent control for nonlinear coupled systems. Tikhonov regularization is a technique employed to adjust neural network weights and prevent overfitting. Additionally, the incorporation of ReLU activation function in the neural network simplifies thearchitecture, avoiding issues like gradient explosion, and optimizes controller performance. Furthermore, sliding surfaces are designed to improve control system stability and robustness. The proposed Tikhonov-tuned sliding neural network (TSN) controller ensures both stability and superior system performance. The methodology emphasizes the importance of determining optimal neural network weights and regularization terms to prevent overfitting, facilitating accurate predictions in inverted pendulum control system applications. To assess the functionality and stability of TSN, this paper employs simulations and experimental implementations to control both the rotary inverted pendulum and the arm-driven inverted pendulum. The results indicate that the proposed TSN methodologies are effective and feasible.

Keywords: inverted pendulum; Tikhonov; sliding; LQR; fuzzy; neural network; decoupling

Citation: Mon, Y.-J. Tikhonov-Tuned Sliding Neural Network Decoupling Control for an Inverted Pendulum. *Electronics* **2023**, *12*, 4415. https://doi.org/10.3390/electronics12214415

Academic Editors: Olivier Sename, Tao Wang, Yimin Zhang, Liang Zhang and Junbin Liu

Received: 31 August 2023
Revised: 29 September 2023
Accepted: 25 October 2023
Published: 26 October 2023

1. Introduction

Tikhonov regularization theory, introduced by Andrey Nikolayevich Tikhonov in the 1950s, provides a systematic introduction to the theory and applications of regularization [1]. It addresses problems with unstable or no solutions, offering an effective mathematical framework to obtain stable and reliable solutions. The main idea is to introduce a regularization term into the cost function, balancing the trade-off between data fitting and solution stability. The regularization term, along with the data-fitting term, contributes to minimizing the cost function. The data-fitting term measures the difference between the solution and the observed data, while the regularization term constrains the solution's characteristics, such as smoothness, sparsity, or minimum energy. By adjusting the regularization term, a balance can be achieved between data fitting and solution stability [2–6].

The inverted pendulum is a highly typical nonlinear coupled system, and its study remains continuously relevant in research. Various fields, including intelligent systems, robotics, industry, and carbon reduction, can benefit from studying the inverted pendulum. This paper proposes the use of Tikhonov regularization theory combined with intelligent algorithms for the control of the inverted pendulum. The inverted pendulum is a classical nonlinear coupled system used to study oscillation and stability. It consists of a rod with a mass attached to one end, allowing it to rotate and swing along an arc path under the influence of gravity. The system's dynamics are driven by gravity and the constraint force of the rod, resulting in various dynamic phenomena such as robotic motion,

periodic motion, chaotic motion, and stability analysis. Due to its nonlinear nature, the inverted pendulum poses challenges for control and stabilization [7–11]. Recent studies have explored the decoupling control of nonlinear systems using Tikhonov regularization theory. By introducing a regularization term, the decoupling of coupling relationships between system state variables is achieved, enabling the design of separate controllers for decoupling control [12–15]. Intelligent control systems for the inverted pendulum often rely on combinations of adaptive control and robust control. The coupling nature of the inverted pendulum system, where the motion of the rod influences the pendulum's position and angle, makes controlling and stabilizing the system more challenging [16–27]. To address the challenges posed by the inverted pendulum system, this paper proposes the combination of a neural network with Tikhonov regularization theory for intelligent decoupling control. Sliding mode control (SMC) is employed for decoupling control in inverted pendulum systems to alleviate interactions among different variables. However, SMC encounters challenges, including computational burden [18], slower convergence [20], and singularity problems [22]. This paper introduces aTikhonov-tuned sliding neural network (TSN) controller, which leverages sliding control for decoupling and incorporates Tikhonov parameters to train a neural network. These parameters are then integrated into the sliding surface to ensure system stability.

The problem statement of this paper revolves around achieving intelligent decoupling control for the inverted pendulum by combining a neural network with Tikhonov regularization theory. The primary objective is to develop an intelligent control approach that effectively decouples the system while maintaining stability through the sliding neural network which is adjusted by the Tikhonov regularization term. This research combines theoretical advancements with practical implementation to contribute to the field of intelligent control systems for the inverted pendulum, with potential applications in various industries and fields.

The contributions of this paper build upon the related literature on Tikhonov regularization, establishing its connections with sliding mode control and neural networks by using simulations and experiments. These foundations contribute significantly to the paper's first contribution, as regularization in Tikhonov regularization and sliding mode control both aim to achieve robustness and stability in control systems. A critical aspect of this paper's approach involves the selection of appropriate regularization terms, numerical simulations, and practical operations. This plays a crucial role in optimizing the performance of sliding mode control using neural networks.

2. Tikhonov Regularization Theory

In a standard linear regression model, the objective is to minimize the prediction mean squared error. However, if the data are limited or the number of features is large, the model can become overly complex and start fitting noise in the training data. This can lead to overfitting, where the model performs well on the training data but poorly on new, unseen data. To address the overfitting issue, we employed Tikhonov regularization. In this approach, we aimed not only to minimize the sum of squared errors but also to incorporate a regularization term into the loss function. This regularization term penalized large coefficient values [1–6].

For example, let us consider a linear regression model with two features (x, y), where the model equation is as follows:

$$p = w_1 x + w_2 y + b \tag{1}$$

where w_1 and w_2 are the coefficients of the features, b is the intercept term, and p is the model output.

In Tikhonov regularization, the loss function will include an additional term proportional to the L2 norm of the coefficients:

$$loss = \frac{1}{N}\sum [(p\_pred - p\_actual)^2 + \lambda(w_1^2 + w_2^2)] \quad (2)$$

where N is the number of data points, *p_pred* is the predicted value, *p_actual* is the actual value, and λ is the regularization term. The purpose of this Tikhonov equation is to balance the accuracy of the predicted values with the complexity of the model. The first term, the mean squared difference, measures the discrepancy between the predicted and actual values, emphasizing the importance of accurate predictions. The second term, the regularization term, helps to control the model's complexity by penalizing large weight values. The regularization term λ determines the trade-off between fitting the data well and keeping the weights small. A higher λ value increases the regularization effect, promoting smaller weights and potentially reducing overfitting. Conversely, a lower λ value gives more weight to accurate predictions, allowing for larger weight values. By optimizing this Tikhonov loss function, the goal was to find the best combination of weight values that minimized the overall loss and achieved a balance between accurate predictions and model complexity.

By including the regularization term, the model was encouraged to keep the coefficients (w_1 and w_2) small while still fitting the data. The values for w_1 and w_2 in the Tikhonov equation were typically obtained through an optimization process. The specific approach to obtaining these values depended on a gradient descent approach, which is an iterative optimization algorithm that aims to find the minimum of a function. In the context of the Tikhonov equation, gradient descent can be used to update the values of w_1 and w_2 by iteratively adjusting them in the direction of the steepest descent of the loss function. The regularization term λ controls the degree of regularization applied. A larger λ value leads to smaller coefficient values and simpler models. In summary, Tikhonov regularization penalizes large coefficient values in a linear regression model, helping prevent overfitting and making the model more balanced, with better generalization ability.

To illustrate this concept, let us demonstrate the prediction of a sine function using the Tikhonov regularization method. First, we generated data for the sine function and defined a Tikhonov regularization function. This function utilized matrix operations to compute the regression parameters, denoted as w. Subsequently, we set the regularization term λ and constructed a design matrix X of size n. Utilizing Equation (2), we calculated the minimum Loss to use the Tikhonov regularization function for prediction and stored the results in y_pred. Finally, we plotted the original sine function and the predicted results of Tikhonov regularization as the blue and red curves, respectively (refer to Figure 1). Tikhonov regularization has been proven to be a valuable technique in control systems, offering stability and reliable solutions for problems with unstable or nonexistent solutions.

When performing polynomial fitting, as the polynomial degree n was set to 5 or higher, the polynomial fitting model became very flexible and could fit every point in the training data, including the noise. This led to the black sine function curve on the graph almost overlapping with the "λ = 0.01" red dash line, which is overfitting. However, such a model may have poor predictive capability on new data because it focuses too much on specific features of the training data and fails to capture the overall trend.

To address the overfitting issue, we could adjust the regularization term λ to a value of 10, as shown by the purple dashed-double-dot line at the bottom. A larger value of λ penalized the model's parameters more strongly, reducing the risk of overfitting. We could try adjusting the value of λ to find the optimal balance point, such as λ = 0.1, as shown by the blue dot line in the middle, which could achieve a balance between fitting the training data and the predictive performance on new data. Figure 1 shows the simulation results of adjusting the value of λ to find the optimal equilibrium point, avoiding overfitting and achieving the best prediction of the sine wave. The different λ yielded a different system performance, as demonstrated in Figure 1. By varying the λ value, we could observe

different levels of system stability, response time, overshoot, and other performance metrics. This allowed for fine-tuning of the system's behavior based on the specific requirements of the application. Therefore, Figure 1 serves as a visual representation of how different λ values impacted the system's performance, providing valuable insights for selecting the most suitable λ value in practical applications.

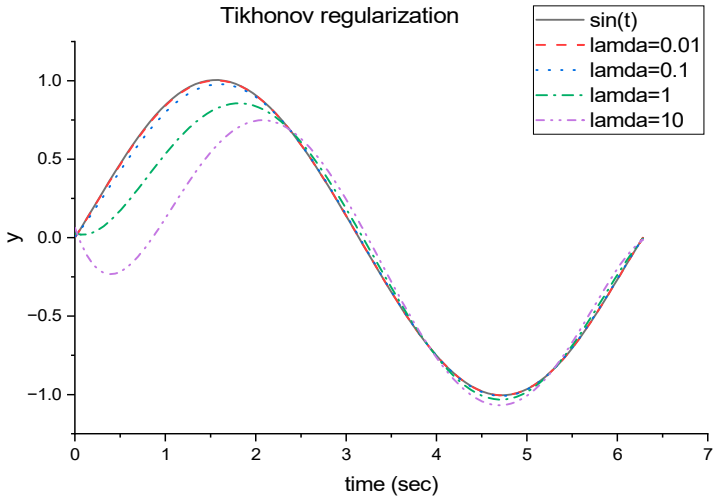


Figure 1. The comparison of different λ for Tikhonov regularization.

3. Intelligent Algorithm Design

The inverted pendulum represents a classic control system problem, wherein a pole is mounted onto a moving cart, and the objective is to keep the pole upright. The primary task of the controller is to generate control signals based on the system's state, including the pole's angle and the cart's position, to maintain balance.

Typically, we would establish a mathematical model to describe the dynamics of the inverted pendulum and use that model for controller design. Tikhonov regularization can be applied to the parameter estimation problem in controller design. In this case, the cost function could be formulated to minimize the error between the system output and the desired output, while incorporating a Tikhonov regularization term to constrain the size and complexity of the parameters. By adjusting the regularization term, a balance could be achieved between fitting the observed data and ensuring the smoothness of the control parameters. Therefore, the first step involved simplifying the nonlinear coupled system into a linear system for design purposes, while the remaining nonlinear characteristics were addressed by the sliding neural network controller. For the linear parts, LQR (linear quadratic regulator) control could be used. For the nonlinear components, Tikhonov regularization could be applied to tune the weights of the neural network. For sliding mode control, the sliding surfaces contributed to the improved performance and robustness of the inverted pendulum system. This enabled adaptation to uncertainties, noise, and model errors through parameter estimation and facilitated the generation of suitable control strategies to stabilize the inverted pendulum and enhance overall performance.

Therefore, this paper proposes the design method of the Tikhonov sliding neural network (TSN) controller, as shown in Figure 2, which combines Tikhonov regularization with a neural network to achieve intelligent control of the inverted pendulum. In this section, a step-by-step explanation is provided of how to combine and compute Tikhonov regularization, neural networks, and sliding mode control. The process starts by setting the goal of achieving intelligent control for nonlinear systems through the application of Tikhonov regularization within the framework of sliding mode control. A sliding mode controller is then established to control the system by sliding its state onto a specific sliding surface to achieve the desired control effect. Neural networks are introduced as a crucial component to monitor and adjust the regularization terms in Tikhonov regularization,

utilizing their learning and adaptive adjustment capabilities. Finally, the combination of Tikhonov regularization, neural network, and sliding mode control is realized through iterative computation and application of the controller, which involves integrating the system's inputs and outputs with the controller and adjusting the regularization terms and neural network weights based on the specific circumstances.

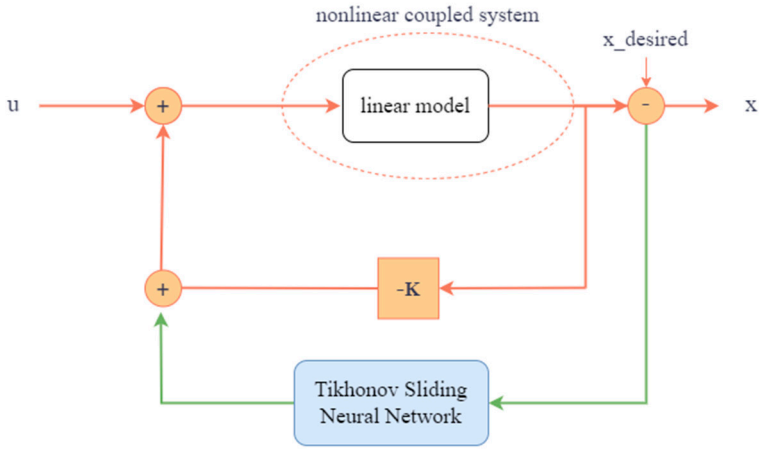


Figure 2. TSN design architecture.

3.1. Linear Model

This subsection indeed serves as a foundation for introducing our proposed nonlinear control method. In this subsection, we establish the groundwork by presenting classical optimal control techniques.

Consider a mathematical linear model presented as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{3}$$

where $\boldsymbol{A} \in R^{mxm}$ and $\boldsymbol{B} \in R^{m}$ are system dynamics matrices, while $\boldsymbol{x}$ and $\boldsymbol{u}$ are all m-dimensional vectors of time t. To introduce the controller into the system, we can represent the control input as $\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}$, where $\boldsymbol{K} \in R^{mxm}$ is the sought controller gain matrix. Substituting u back into the dynamics model, we obtain the following:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{x} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x} \tag{4}$$

Below, we will briefly introduce one method to derive the solution for $\boldsymbol{K}$. Suppose our goal is to design a state feedback controller that keeps the angle of the inverted pendulum near the target angle. According to the LQR theory [8,25], our control objective can be defined as minimizing the cost function J:

$$J = \int [\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}]dt \tag{5}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ represent the state weight matrix and control weight matrix, respectively, both having a square dimension of m. We want to adjust the controller gain matrix K to minimize J. The optimal solution can be derived through algebraic or differential methods in optimal control theory. One common approach is to solve the algebraic Riccati equation [26]. By solving the following Riccati equation, we can obtain the optimal controller gain matrix $\boldsymbol{K}$:

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} = 0 \tag{6}$$

where $\boldsymbol{P}$ is the symmetric positive definite state feedback gain matrix with a square dimension of 'm'. Once it is determined, the controller gain matrix $\boldsymbol{K}$ can be selected using the following formula, ensuring that the system described by Equation (6) has negative

eigenvalues to maintain the stability of Equation (4). The detailed derivation process is presented in Appendix A.

$$\boldsymbol{K}_{LQR} = \boldsymbol{R}^{-1}\boldsymbol{B}^{T}\boldsymbol{P} \tag{7}$$

Thus, the optimal state feedback controller can be expressed as:

$$\boldsymbol{u}_{LQR} = -\boldsymbol{K}_{LQR}\boldsymbol{x} \tag{8}$$

3.2. Nonlinear Model

From Equation (5), this approach optimizes the controller's performance by minimizing the cost function J and guiding the system's state towards the desired stable point. However, in certain cases, it may be necessary to introduce a regularization term into the controller design to address specific issues. For example, when the system's model contains uncertainties or noise, a sliding mode neural network can be employed to estimate the controller's parameters, thereby improving robustness and performance.

A canonical nonlinear system is represented as below:

$$\dot{\boldsymbol{x}} = \boldsymbol{f} + \boldsymbol{G}\boldsymbol{u} + \boldsymbol{d} \tag{9}$$

where $\boldsymbol{x}$, $\boldsymbol{f}$, $\boldsymbol{u}$, and $\boldsymbol{d}$ are the state vector, dynamic with respect to $\boldsymbol{x}$; control input vector; and disturbance vector, respectively. Each of these vectors has a dimension of 'm' and varies over time. $\boldsymbol{G}$ is an m-dimensional square system matrix. The unknown disturbance is assumed to have a known limited norm bound, i.e., $\|\boldsymbol{d}\| \leq d_b$. The control law can be deduced from Equation (9), and the expression is as follows:

$$\boldsymbol{u} = \boldsymbol{G}^{-1}(\dot{\boldsymbol{x}} - \boldsymbol{f} - \boldsymbol{d}) \tag{10}$$

Then, according to the universal approximation theorem [25], there should be an optimal control law, which is expressed as follows:

$$\boldsymbol{u}^{*} = \boldsymbol{G}^{-1}(\dot{\boldsymbol{x}}_{\boldsymbol{d}} - \boldsymbol{f} - \boldsymbol{d} - \boldsymbol{K}\boldsymbol{e}); \tag{11}$$

by defining the tracking error vector and the error dynamic as below:

$$\boldsymbol{e}\Delta\boldsymbol{x}_{d} - \boldsymbol{x}. \tag{12}$$

$$\dot{\boldsymbol{e}} = \boldsymbol{K}\boldsymbol{e} + \boldsymbol{G}(\boldsymbol{u}^{*} - \boldsymbol{u}), \tag{13}$$

where $\boldsymbol{e} \in R^{m}$.

If we assume that $\boldsymbol{u}^{*}$ is equal to $\boldsymbol{u}$ (as indicated in Equation (13)), we can then select an appropriate value of $\boldsymbol{K}$ as defined in Equation (8) to ensure the convergence of $\boldsymbol{e}$.

However, in practice, the $\boldsymbol{f}$ and $\boldsymbol{d}$ are always unknown, making the design $\boldsymbol{u}^{*}$ of Equation (12) challenging. Many recent studies have attempted to address this problem, but they encounter issues with complex manipulations and inadequate performance. In this paper, we propose a simple approach to designing the controller. We utilize a Tikhonov-tuned sliding mode neural network (TSN) controller to handle model-free problems, ensuring that the states closely approach the desired region. Simultaneously, a sliding mode controller is employed to maintain the system's states on the sliding surface, eventually allowing $\boldsymbol{u}$ to approximate $\boldsymbol{u}^{*}$, leading to improved performance and making sure that the states are converged. The designed controller is as follows:

$$\boldsymbol{u} = \boldsymbol{u}_{LQR} + \boldsymbol{u}_{TSN} \tag{14}$$

where $\boldsymbol{u}_{LQR}$ represents an LQR linear controller discussed in Section 3.1, while $\boldsymbol{u}_{TSN}$ denotes a TSN controller. The proposed TSN control system is introduced in Figures 2 and 3 and will be further elaborated in the following sections.

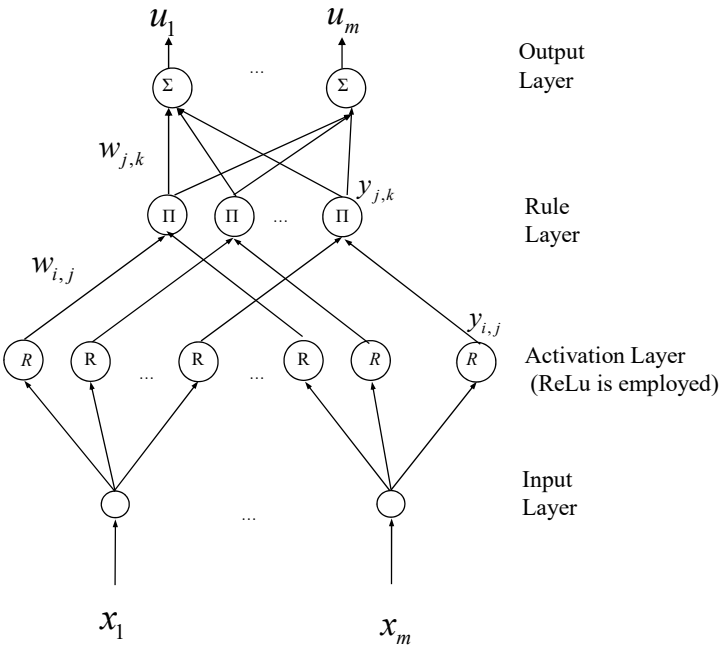


Figure 3. Tikhonov-tuned neural network (TNN) diagram.

3.3. Tikhonov-Tuned Neural Network (TNN) Architecture

Figure 3 depicts a four-layer Tikhonov-tuned neural network (TNN). It comprises the input, activation, rule, and output layers. The energy function is minimized using the gradient descent method to obtain optimal weights for the TNN. Signal propagation and basic functions in each layer of the TNN are introduced as follows:

(a) Input Layer: In this layer, each node is associated with net input and net output values, denoted as x_i and $y_{i,j}$, respectively.

$$net_i = x_i \tag{15}$$

$$y_{i,j} = f_{i,j}(net_i) = net_i,\ i = 1, 2, \cdots, m, \text{ and } j = 1, 2, \cdots, m \tag{16}$$

where x_i represents the i-th input to the node of Layer 1 and m is the number of input variables. The link weights in this layer are all set to unity.

(b) Activation Layer: Each node in this layer performs an activation using the ReLU (rectified linear unit) function, which is defined as $\mathrm{Re}Lu(x) = \max(0, x)$ [28]. The derivative of ReLU is 1 when the input number is greater than zero; otherwise, it is equal to 0. ReLU will be used as the activation function for the j-th node of the i-th input.

$$net_{ij} = \max(0, y_{i,j}) \tag{17}$$

(c) Rule Layer: The cardinality of this layer determines the count of fuzzy rules, and its neuron count profoundly influences the rule complexity. Each individual node, represented by $\prod$, plays a pivotal role as it multiplies the incoming signal, giving birth to the emergent output specific to the j-th rule node. This dynamic interaction fosters diverse rule behaviors, leading to intricate and versatile system responses. The formal expression is given by the following equation:

$$y_{jk} = \prod_{ij} net_{ij},\ i = j = 1, 2, \ldots, m \tag{18}$$

(e) Output Layer: This comprises multiple nodes, each referred to as an output node, responsible for performing the crucial defuzzification operation. Among these nodes, we

have a particular one labeled as Σ which calculates the overall output by summing all incoming signals. This summation process is denoted by the following equations:

$$net_k = \sum_j w_{jk}\, y_{jk} \tag{19}$$

$$u_k = f_k(net_k) = net_k,\ k = 1, 2, \cdots, m \tag{20}$$

In Equations (19) and (20), we can further describe the components involved in this calculation:

w_{jk}: The link weight, representing the output action strength of the j-th column output associated with the k-th rule.

y_{jk}: The j-th input to the node of Layer 4.

m: The total number of output nodes.

u_k: The k-th column output of the TNN controller.

ReLU activation has the property of avoiding the vanishing gradient problem and promoting faster training by allowing gradients to flow when inputs are positive. As a result, the TNN architecture can handle zero input values and propagate gradients effectively, even in the absence of bias neurons.

Furthermore, Tikhonov regularization factors are incorporated into the TNN to ensure stability. These regularization factors help prevent overfitting and improve the generalization capability of the network.

3.4. Neural Network Controller Design

According to the gradient theory, we consider a single neuron to derive the equations in the neural network. To tune the weights of the TNN, the energy function and cost function of the Tikhonov are defined as follows to employ the gradient operation:

$$E_1(u, w) = \frac{1}{2}(u^* - u)^2 \tag{21}$$

$$E_2(w) = \frac{\lambda}{2} w^2 \tag{22}$$

where λ is the Tikhonov regularization term and $E_2(w)$ is the cost function of the Tikhonov function. Thus, the total energy function can be presented as follows:

$$E(u, w) = E_1(u, w) + E_2(w) \tag{23}$$

The gradient calculus of E with respect to w can be obtained as follows:

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = -\eta\left(\frac{\partial E_1}{\partial u} * \frac{\partial u}{\partial w_j} + \frac{\partial E_2}{\partial w_j}\right) = (u^* - u) * y_k + \lambda w_j \tag{24}$$

where η is the learning rate. Similarly, the activation layer update law can be as follows:

$$\Delta w_i = -\eta \sum_i \frac{\partial E}{\partial w_i} = -\eta\left(\frac{\partial E}{\partial y_j} * \frac{\partial y_j}{\partial w_j}\right) = y_j \sum_j w_j * (u^* - u) * y_j + \lambda w_i \tag{25}$$

Up until this point, the derivations are complete, and the entire TNN design is now finalized. The weight update laws for the activation layer and rule layer are presented in Equations (24) and (25), respectively. These two layers play a crucial role in the TNN. Additionally, the Tikhonov regularization term is incorporated to prevent overfitting. Moreover, the model demonstrates the capability to predict the next state of the system and generate appropriate control forces to address uncertainties and disturbances.

In the derivation process, the variable $\boldsymbol{x}$ is used. However, in the subsequent practical application, the error variable $\boldsymbol{e}$ is substituted for $\boldsymbol{x}$ in calculations.

3.5. Sliding Mode Controller Design

From Equation (12), according to the universal approximation theorem [25], suppose there exists an optimal $\boldsymbol{u}^*$ such that

$$\boldsymbol{u}^* = \boldsymbol{u}_{LQR} + \boldsymbol{\varepsilon} \tag{26}$$

where the term $\boldsymbol{\varepsilon}$ denotes the approximation error. $\boldsymbol{u}_{LQR}$ is the LQR controller discussed in the previous section.

By the substitution of (26) and (14) into (13), the error dynamics become

$$\boldsymbol{e}\,. = \boldsymbol{K}\boldsymbol{e} + \boldsymbol{G}(\boldsymbol{\varepsilon} - \boldsymbol{u}_{TSN}) \tag{27}$$

where $\boldsymbol{\varepsilon}$ is assumed to be bounded by $0 \leq \|\boldsymbol{\varepsilon}\| \leq \|\boldsymbol{\zeta}\|$, where $\boldsymbol{\zeta}$ is a carefully selected positive constant vector. The sliding mode controller aims to mitigate the influence of the approximation error. In the forthcoming section's demonstration, the TSN controller can be achieved in the following manner:

$$\boldsymbol{u}_{TSN} = \|\boldsymbol{\zeta}\| sign(\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e}) \tag{28}$$

where '*sign*' represents the sign function, which can be replaced with a saturation function to mitigate chattering effects. Furthermore, $\boldsymbol{P} \in R^{m\times m}$ is a symmetric positive definite matrix.

3.6. Stability Analysis

According to the stability theorem of nonlinear systems [25], let us define the Lyapunov function as follows:

$$V = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e} \tag{29}$$

The derivative of the Lyapunov function can be expressed as follows:

$$\dot{V} = \frac{1}{2}(\,\dot{\boldsymbol{e}}^T\boldsymbol{P}\boldsymbol{e} + \boldsymbol{e}^T\boldsymbol{P}\,\dot{\boldsymbol{e}}). \tag{30}$$

From Equations (27) and (30), we arrive at the following equation:

$$\begin{aligned}\dot{V} &= \tfrac{1}{2}[\boldsymbol{e}^T\boldsymbol{K}^T\boldsymbol{P}\boldsymbol{e} + \boldsymbol{\varepsilon}^T\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e} - \boldsymbol{u}_{TSN}^T\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e}] + \tfrac{1}{2}[\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{K}\boldsymbol{e} + \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{G}\boldsymbol{\varepsilon} - \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{G}\boldsymbol{u}_{TSN}] \\ &= -\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + \boldsymbol{\varepsilon}^T\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e} - \boldsymbol{u}_{TSN}^T\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e}\end{aligned} \tag{31}$$

where $\boldsymbol{Q}$ is a positive definite symmetric matrix and $\boldsymbol{K}$ is a Hurwitz matrix. Matrix $\boldsymbol{Q}$ can be utilized to represent the common part of $\boldsymbol{K}^T\boldsymbol{P}$ and $\boldsymbol{P}\boldsymbol{K}$ in Equation (31), which is given by

$$\boldsymbol{Q} = -\frac{1}{2}(\boldsymbol{K}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{K}). \tag{32}$$

Substituting Equation (28) into (31) will result in the following:

$$\begin{aligned}\dot{V} &= -\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + \varepsilon^T\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e} - \boldsymbol{u}_{TSN}^T\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e} \\ &= -\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + \varepsilon^T\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e} - \|\zeta\|\mathrm{sgn}(\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e})^T\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e} \\ &\leq -\|\boldsymbol{e}\| mineig(\boldsymbol{Q})\|\boldsymbol{e}\| + \|\boldsymbol{\varepsilon}\|\left\|\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e}\right\| - \|\zeta\|\left\|\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e}\right\| \\ &\leq -\|\boldsymbol{e}\| mineig(\boldsymbol{Q})\|\boldsymbol{e}\| + (\|\boldsymbol{\varepsilon}\| - \|\boldsymbol{\zeta}\|)\left\|\boldsymbol{G}^T\boldsymbol{P}\boldsymbol{e}\right\| \leq 0;\end{aligned} \tag{33}$$

where $mineig(\boldsymbol{Q})$ is the minimum eigenvalue of the matrix $\boldsymbol{Q}$. We can select a large eigen diagonal matrix and the positive vector ζ in the context of Lyapunov stability. The choice of a large eigen diagonal matrix is indeed a key factor in ensuring the Lyapunov stability of the system. In summary, the choice of a large eigen diagonal matrix is a deliberate strategy to ensure the Lyapunov stability of the system, and the selection of the positive vector ζ is a

critical part of this analysis, but it may vary depending on the context and specific stability requirements of the problem at hand. Since the error vector $\boldsymbol{e}$ is measured using the 2-norm, the right-hand side of Equation (27) remains bounded, ensuring a stable behavior for our system. Additionally, based on Equations (29) and (33), we can find the Lyapunov function V using the 2-norm and $\dot{V}$ infinity norm, respectively. Applying Barbalat's Lemma, which guarantees the convergence of states, we confidently conclude that the error vector $\boldsymbol{e}$ will approach zero, demonstrating the stability of the system.

The aforementioned design methods are summarized in Figure 4, with each block marked with its corresponding equations.

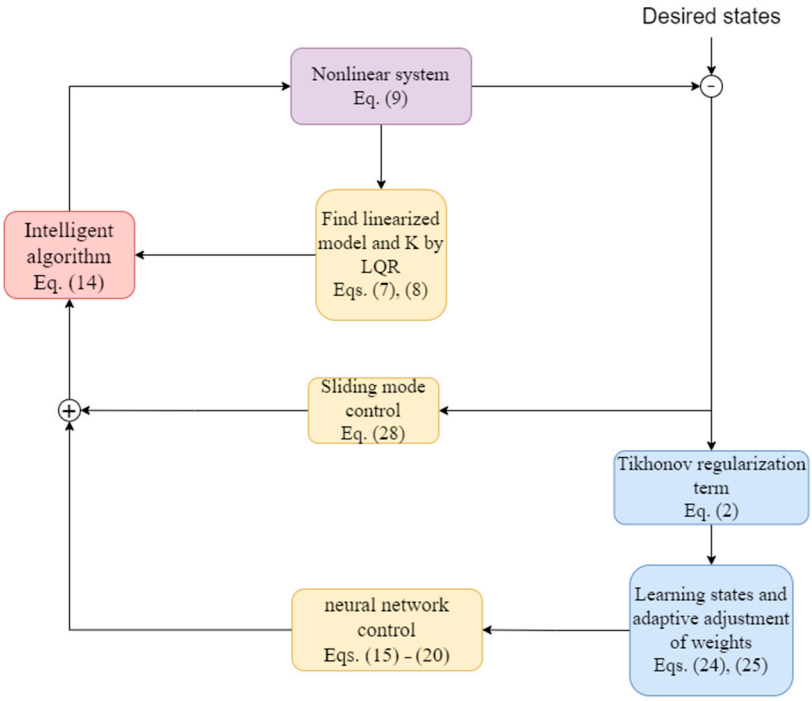


Figure 4. TSN design block diagram.

4. Simulation Result

4.1. Example 1: Cart Inverted Pendulum

By analyzing the linear quadratic regulator (LQR) theory, we could determine the stability of the linear model and obtain the controller K. Consistently observing a negative derivative in Lyapunov analysis indicates stability in the nonlinear system. Introducing Tikhonov regularization λ allowed us to fine-tune the weights of the TNN in order to prevent overfitting and achieve accurate predictions to compensate for the control forces of the nonlinear system, especially in cases of coupled systems. The training objective was to drive the sliding surface, influenced by various errors, to converge towards zero.

Consider the model of the inverted pendulum as below: [25]

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{g\sin(\theta) - aml\dot{\theta}^2\sin(2\theta)/2 - a\cos(\theta)u}{4l/3 - aml\cos^2(\theta)} \end{aligned} \tag{34}$$

where $a = (M+m)^{-1}$, $x_1 = \theta$ is the angle from the vertical, and u is the control force (Newton). The simulation model for an inverted pendulum was utilized as described below and adhered to the specifications outlined in Table 1.

Table 1. Simulation parameters of cart inverted pendulum.

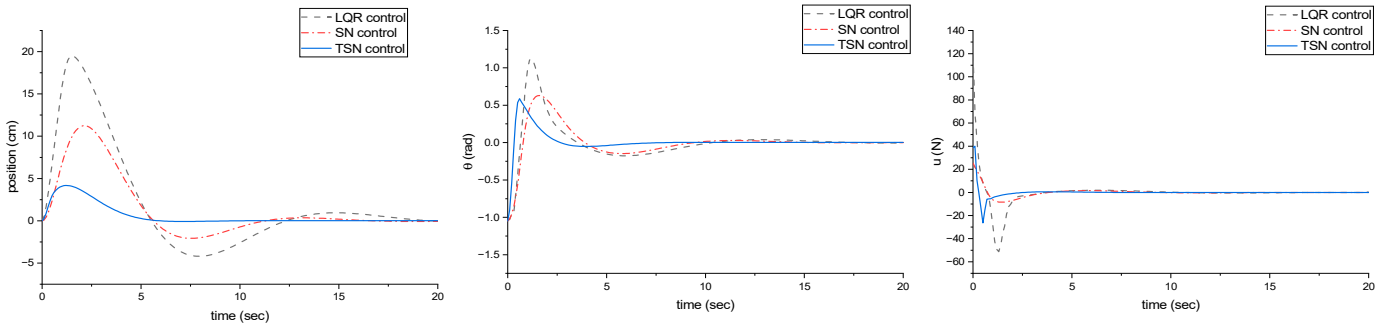
l	M	m	g	θ	Position
Length of pendulum	Mass of cart	Mass of pendulum	Acceleration due to gravity	Initial angle from the vertical	Initial position
20 cm	1 kg	0.2 kg	9.8m/s^2	-1.05 radians	0 cm

Based on the procedure outlined in Section 3.1, the linearized model around the original point couldbe obtained as follows:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 17.3 & 0 \end{bmatrix}$$
$$\boldsymbol{B} = \begin{bmatrix} 0 \\ -0.21 \end{bmatrix} \tag{35}$$

Subsequently, the values of $\boldsymbol{K}$ couldbe determined as $\boldsymbol{K}$= $[-258.12 \ -170.35]$; then, the eigenvalue of $\boldsymbol{A} - \boldsymbol{BK}$ was $[-1.0632 \ -34.7103]$. As shown in Figure 2, the control system employed a sliding neural network to handle the nonlinear coupling characteristics and dynamically adjust the weights of the TNN to ensure both stability and predictability. After comparing the LQR, sliding neuralnetwork (SN), and Tikhonovsliding neural network (TSN) methods, as depicted in Figure 3, under the condition of requiring both the position and the angle to be zero, it becameevident that the TSN method excelled in decoupling. The weight factor values, presented in matrix format, were derived from the aforementioned sections. These values were stored in a matrix, where each row represented a distinct factor or variable, and each column corresponded to the relevant weight factor value. The optimal regularization term λ was set to 0.1, and initial weightings were all set to 0.15. According to Equation (28), the matrics of $\boldsymbol{P} = diag\begin{bmatrix}10 & 10\end{bmatrix}$, $\boldsymbol{G} = diag\begin{bmatrix}1 & 1\end{bmatrix}$, and $\zeta = diag\begin{bmatrix}10 & 10\end{bmatrix}$.

Performance indices serve as quantitative measures to evaluate the performance of the simulation of an inverted pendulum system. These indices provide insights into various aspects, including stability, tracking accuracy, control effort, settling time, and energy efficiency. Figure 5 clearly demonstrates that the TSN controller outperformed both the LQR and theSN controllers.



(**a**) (**b**) (**c**)

Figure 5. (**a**) Position result; (**b**) angle result; (**c**) control force.

4.2. Example 2: Rotary Inverted Pendulum

The rotary inverted pendulum represents a unique variant of the inverted pendulum, distinct from the cart inverted pendulum, which is governed by the linear back-and-forth motion of a cart. To assemble the rotary inverted pendulum, we affixed a connecting pivot arm to a servo motor's shaft, employing rotational torque to manage another connecting pendulum fastened to its trailing end. When the connecting pendulum remained motionless, it hung vertically due to gravity. Upon applying a driving torque, it overcame gravity,

assuming an upright vertical position. Consequently, it did not follow a linear path but instead executed a circular rotation. The control principles for this system resemble the control laws detailed in this paper. Both systems are interconnected, nonlinear systems, where a single input governs two outputs. The connecting pivot arm's position transitioned from 0 degrees and eventually returned to 0 degrees. The pendulum's position oscillated from 0 degrees to either positive or negative 180 degrees, contingent on its initial rotation direction, whether clockwise or counterclockwise. Figure 6 illustrates the architectural diagram.

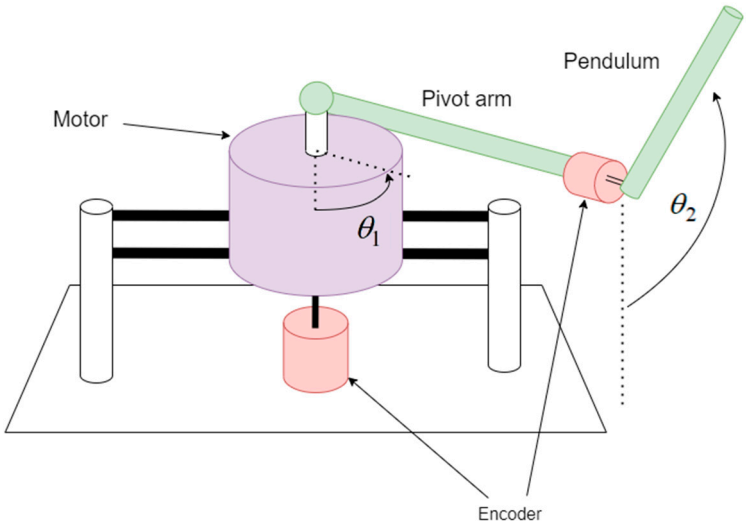


Figure 6. The architecture diagram of rotary inverted pendulum.

The equation representing the rotary inverted pendulum model is provided below [29]:

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= \tfrac{-0.5\sin 2x_2}{\Delta}(P_2P_4x_4^2 + P_4P_5x_3x_4 + P_2P_3x_3^2\cos x_2 + P_3P_6) \\
&\quad + \tfrac{-0.5P_4}{\Delta}(-2P_3x_4^2\sin x_2 + 2P_7x_3 - 2P_8) \\
\dot{x}_4 &= \tfrac{-0.5\sin 2x_2}{\Delta}(-P_3x_4^2 + P_2P_3x_4^2\cos x_2 + P_3P_5x_3x_4\cos x_2 + P_1P_2x_3^2 + P_2^2x_3^2\sin 2x_2) \\
&\quad + \tfrac{-\cos x_2}{\Delta}(P_3P_7x_3 - P_3P_8u) + \tfrac{-P_6\sin x_2}{\Delta}(P_1 - P_2\sin^2 x_2)
\end{aligned} \tag{36}$$

where $\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T = \begin{bmatrix} \theta_1 & \theta_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T$, $P_1 = m_2l_1^2 + J_1$, $P_2 = m_2c_2^2$, $P_3 = m_2l_1c_2$, $P_4 = m_2c_2^2 + J_2$, $P_5 = m_2l_1^2$, $P_6 = m_2c_2g$, $P_7 = k_b^2/R_m$, $P_8 = k_t/R_m$, and $\Delta = P_1P_4 + P_2P_4\sin^2 x_2 - P_3^2\cos^2 x_2$. For a comprehensive list of simulation and physical parameters, please refer to Table 2.

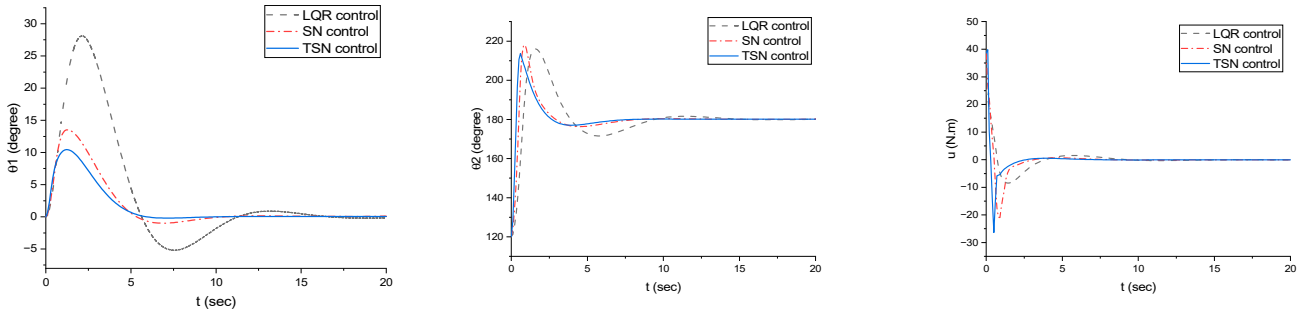
Table 2. Simulation parameters of rotary inverted pendulum.

l_1,l_2	m_1,m_2	C_1,C_2	J_1,J_2	K_b,K_t	R_m	x_1,x_2	x_3,x_4
Length of arm and pendulum	Mass of arm and pendulum	Distance to center of mass of arm and pendulum	Inertial of arm and pendulum	Constants of back-emf and torque	Armature resistance	Angular displacement of arm and pendulum	Angular velocity of arm and pendulum
16 cm	56 g and 22 g	8 cm	1.57 and 1.79 kg × cm^2	0.01826	2.56 Ω	0°, 120°	0°/s, 0°/s

As shown in Example 1, following the procedure outlined in Section 3.1, we couldobtain the linearized model around the original point in the following manner:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -7.12 & -0.09 & 0 \\ 0 & 59.33 & 0.07 & 0 \end{bmatrix}$$
$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 7 & -5.4 \end{bmatrix}^T. \tag{37}$$

Utilizing the identical approach as illustrated in Example 1, we could obtain the feedback gain as $\boldsymbol{K}$= [−9.8 −114.4 −6.22 −15.48]. The optimal regularization parameter was configured at λ=0.05, and initial weightings were all set as 0.1.According to Equation (28), the matrics of $\boldsymbol{P} = diag\begin{bmatrix} 20 & 20 & 15 & 15 \end{bmatrix}$, $\boldsymbol{G} = diag\begin{bmatrix} 2 & 2 & 3 & 3 \end{bmatrix}$, and $\boldsymbol{\zeta} = diag\begin{bmatrix} 1 & 1 & 5 & 5 \end{bmatrix}$. The simulation began with initial conditions set to $\theta_1 = 0^\circ$ and $\theta_2 = 120^\circ$, and the results are compared to those of LQR and SN in Figure 7.



(**a**) (**b**) (**c**)

Figure 7. (**a**) θ_1 result; (**b**) θ_2 result; (**c**) control force.

4.3. Example 3: Arm-Driven Inverted Pendulum

The arm-driven inverted pendulum is a robotic system consisting of two freely rotating links. These links operate on a common plane, with the lower motor applying torque to rotate the arm, which, in turn, drives the upper link. The primary objective of this system is to maintain equilibrium, ensuring both poles remain upright and balanced without toppling over. Due to the influence of gravity on both connecting rods and the fact that only one force propels them, nonlinear characteristics interplay and mutually affect each other. Consequently, controlling this system proves to be more challenging compared to a rotary inverted pendulum, as it represents a highly coupled nonlinear system. It necessitates not only preventing the driving link from falling but also ensuring the arm remains stable. In this paper, we propose a method to address the issue of coupling in this system. The architecture diagram of arm-driven inverted pendulum is shown in Figure 8.

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= \tfrac{1}{\Delta}[(P_2\Delta_1 - (P_2 + P_3\cos x_2)\Delta_2] \\ \dot{x}_4 &= \tfrac{1}{\Delta}[P_1 + P_2 + 2P_3\Delta_2\cos x_2 - (P_2 + P_3\cos x_2)\Delta_1] \end{aligned} \tag{38}$$

where

$$\begin{aligned} \Delta &= (P_1 + P_2 + 2P_3\cos x_2)P_2 - (P_2 + P_3\cos x_2)^2 \\ \Delta_1 &= P_3x_3x_4\sin x_2 - \tfrac{k_b^2}{R_m}x_3 + P_3x_4(x_3 + x_4)\sin x_2 - P_4\cos x_1 - P_5\cos(x_1 + x_2) + \tfrac{k_b}{R_m}e \ ; \\ \Delta_2 &= -P_3x_3^2\sin x_2 - P_5\cos(x_1 + x_2) \end{aligned} \tag{39}$$

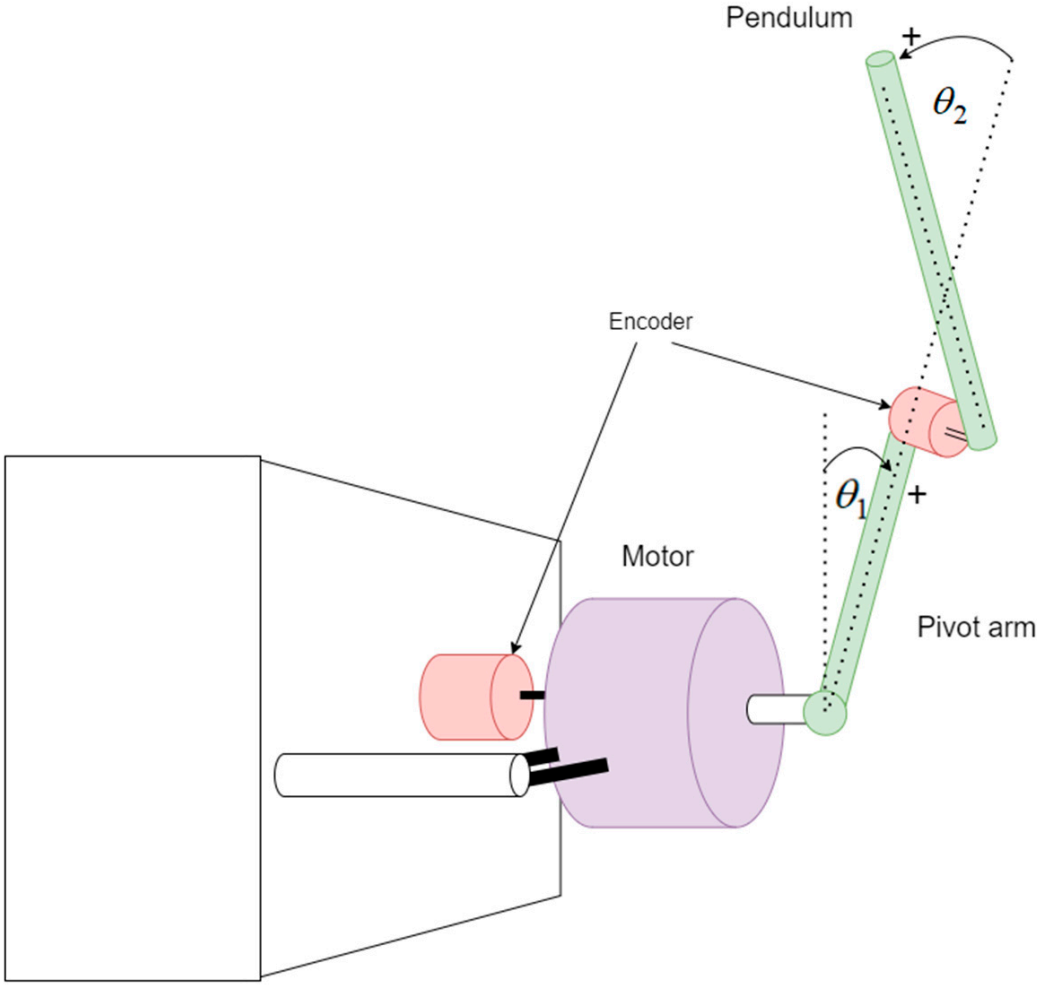
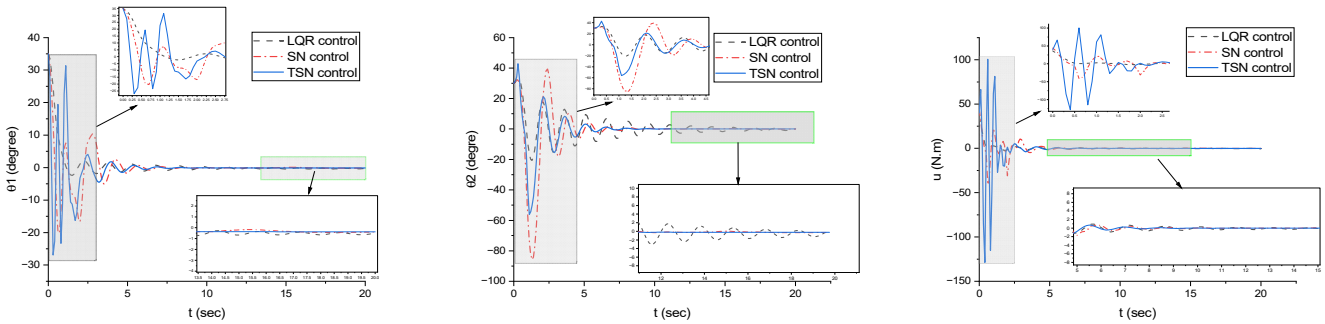


Figure 8. The architecture diagram of arm-driven inverted pendulum.

$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T = \begin{bmatrix} \theta_1 & \theta_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T$, $P_1 = m_1 c_1^2 + m_2 l_1^2 + J_1$, $P_2 = m_2 c_2^2 + J_2$, $P_3 = m_2 l_1 c_2$, $P_4 = (m_1 c_1 + m_2 l_1) g$, and $P_5 = m_2 c_2 g$. All parameters were identical to those in example 2, specifically in Table 2. Following the same procedure as in example 2, we linearized the model around the original point as shown in Equation (40).

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -19.9 & 9.8 & -0.03 & 0 \\ 74.4 & 100.5 & -0.03 & 0 \end{bmatrix} \\ \boldsymbol{B} &= \begin{bmatrix} 0 & 0 & 2.16 & 1.89 \end{bmatrix}^T. \end{aligned} \tag{40}$$

The feedback gain was determined as follows: $\boldsymbol{K}$= [69.16, 141.81, 1.44, 9.78]. We set the optimal regularization parameter to λ= 0.03, and initial weightings were all set as 0.1. According to Equation (28), the matrics of $\boldsymbol{P} = diag\begin{bmatrix} 30 & 30 & 50 & 50 \end{bmatrix}$, $\boldsymbol{G} = diag\begin{bmatrix} 20 & 20 & 30 & 30 \end{bmatrix}$, and $\boldsymbol{\zeta} = diag\begin{bmatrix} 10 & 10 & 25 & 25 \end{bmatrix}$. The simulation was initiated with initial conditions $x_1 = \theta_1$,$x_2 = \theta_2$, $x_3 = \dot{\theta}_1$, and $x_4 = \dot{\theta}_2$, which were defined as in Figure 8 and set to $\theta_1 = 35^\circ$ and $\theta_2 = 30^\circ$; the results were compared to those obtained using the LQR and SN methods, as shown in Figure 9.



(**a**) (**b**) (**c**)

Figure 9. (**a**) Results of θ_1; (**b**) results of θ_2; (**c**) control force.

The simulation results reveal that both the cart and rotary inverted pendulums exhibit low coupling, which makes it difficult to distinctly demonstrate the advantages of the TSN control law when compared to the other two control laws. However, in the case of the arm-driven inverted pendulum system, which is a highly coupled nonlinear system, the superiority of TSN becomes evident.

During the arm-driven inverted pendulum simulation, we observed that the LQR control exhibited oscillations before converging to the steady state. In contrast, the SN control law displayed a longer settling time and higher overshot in transient response, making it less effective than the TSN control law. The key differentiator lies in TSN's utilization of Tikhonov's regularization term, which enhanced the decoupling performance by adjusting the weight of the neural network to improve prediction.

In the following section, we will conduct practical verifications of two types of inverted pendulums: rotary and arm-driven. Subsequently, we will demonstrate the robustness and decoupling performance of TSN, compared to implementing the other two methods, LQR and SN. The LQR and SN controllers face difficulties in real empirical experiments due to factors such as vibration and the precision of encoders. These factors can indeed render the system unstable in the steady state.

5. Experimental Operation

5.1. Establishing the Experiment Setup Environment

First, the driver software usb20emurst 2.0 needed to be installed and the environment needed to be set up. The actual setup utilized a mechanical module of a rotary inverted pendulum (shown in Figure 10) produced by TeraSoft Inc, Taipei, Taiwan. [29], control board, power supply, and emulator. Its control principle was the same as the simulation, with the only difference being an initial swing-up motion. Once it reached the controllable stage, which was when the angle transmitted by the encoder fell between controllable regions, the proposed TSN decoupled control law was taken over by the controller. The software used included TI (Texas Instruments, Dallas, TX, USA) Inc. CCS 2 (Code Composer Studio 2) [30] and MathWorks Inc., The Natick Mall, MA, USA. MATLAB R14 RTW (Real-Time Workshop R14) and Simulink R14 [31], while the hardware consisted primarily of a driver board powered by the TI F2812 DSP chip. After powering on, the JTAG emulator connection was reset for normal operation, as shown in Figure 11. Then, the Simulink program interface was opened, as depicted in Figure 12. The system, through RTW, automatically burned the code into the TI F2812 chip, as depicted in Figure 13.

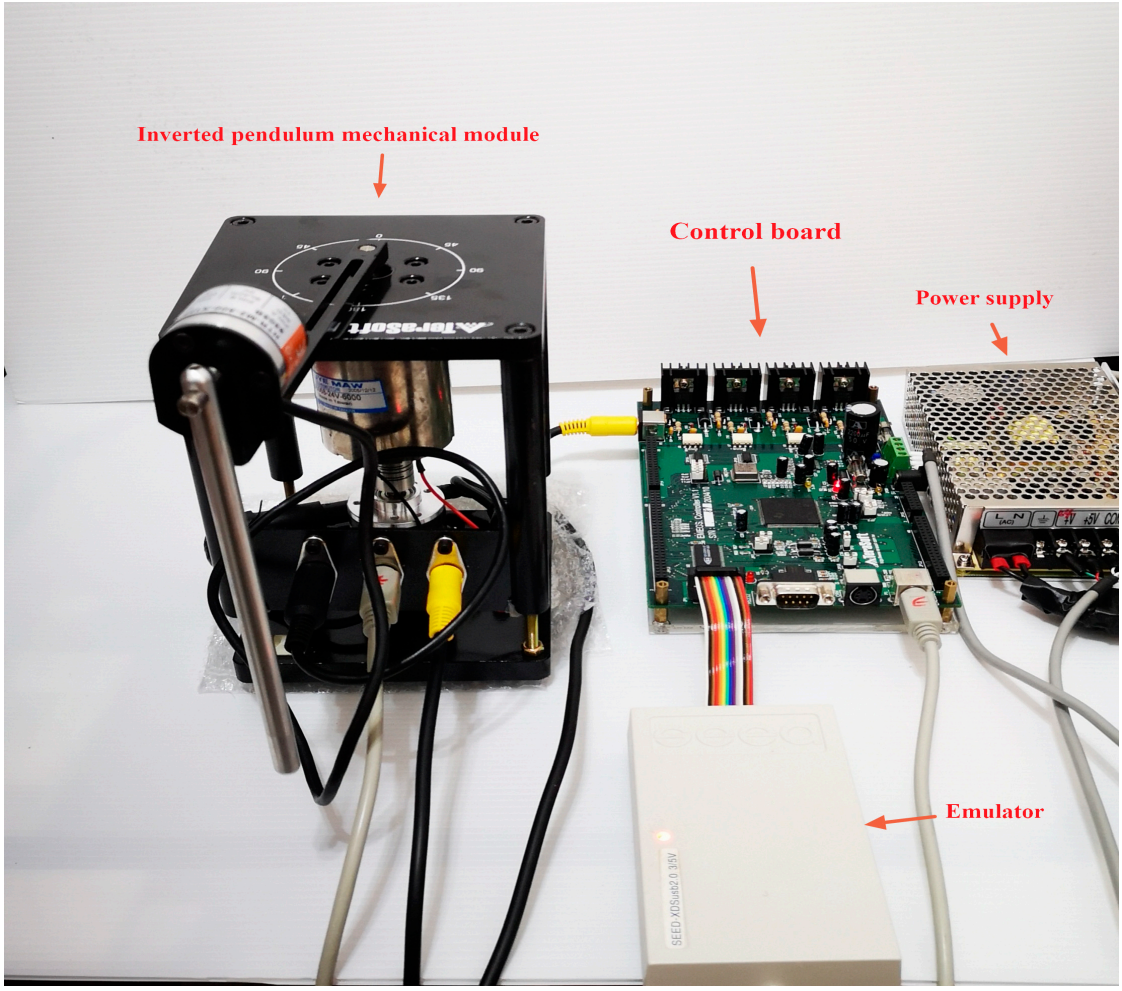


Figure 10. Photograph of inverted pendulum system.

Figure 11. JTAG emulatorresetnormal diagram.

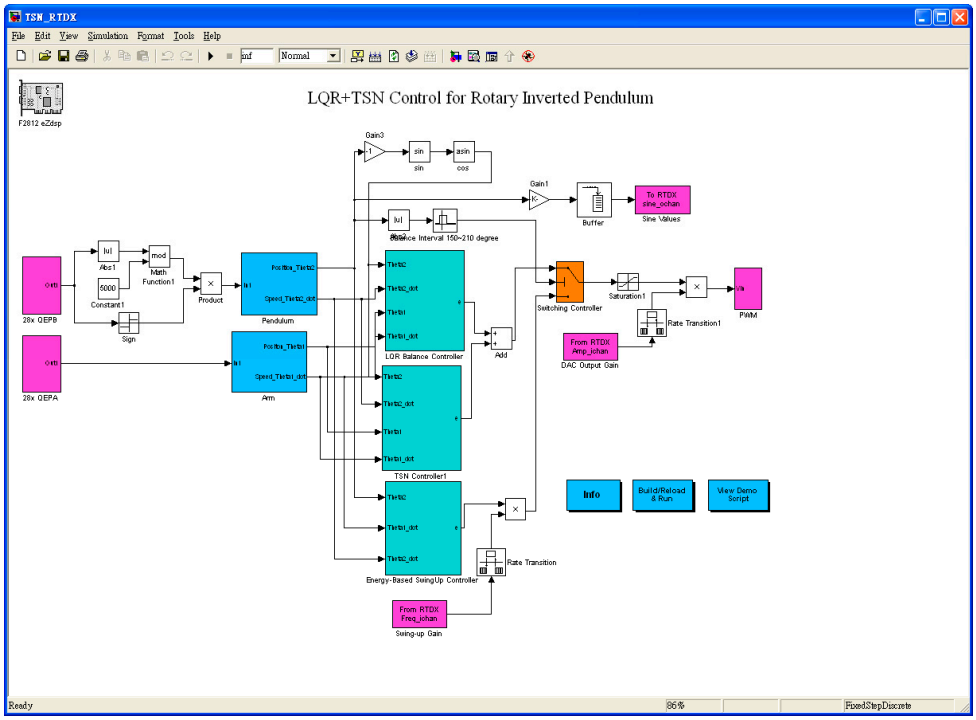


Figure 12. Simulink program interface diagram.

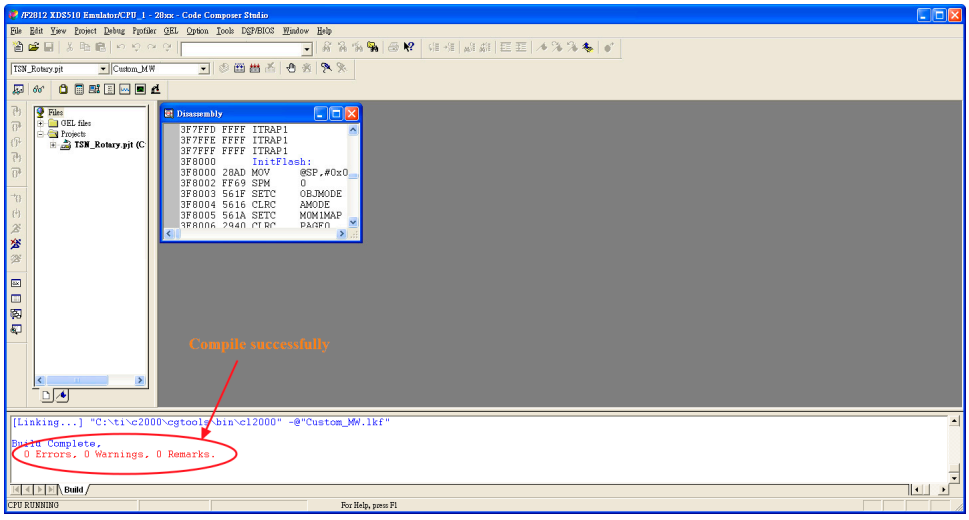
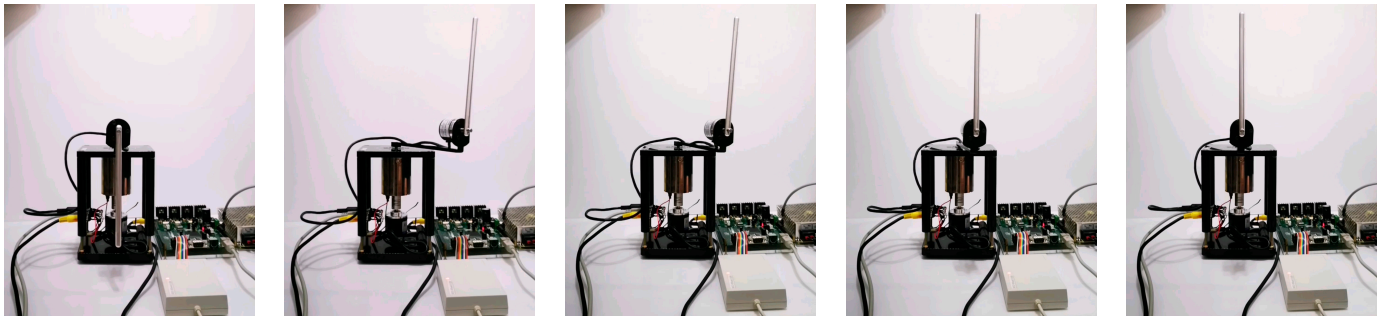


Figure 13. CCS burned the code into the TI F2812 interface diagram.

5.2. Experiment of Rotary Inverted Pendulum

In this experiment, we commenced our investigation by examining the lower coupling effect on the rotary inverted pendulum. Upon activation of the inverted pendulum, it rapidly stabilized, returning to its zero position and angle, as visually depicted in Figure 14. Figure 14 comprises five subfigures: the leftmost one shows the initial photograph, while the middle sections illustrate various stages of the pendulum's motion, and the rightmost one showcases its stable endpoint. Our results obtained from the TSN controller reveal that the performance of the inverted pendulum control system is notably superior to that of both the SN and the LQR controllers.



(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 14. Photographs illustrating the progression of rotary inverted pendulum experiments with TSN control: (**a**) initial states; (**b**) pendulum swing-up; (**c**) convergence to zero states; (**d**) stable two-state configuration; (**e**) stable steady-states achieved.

While the differences in simulations were relatively small, in real experiments, the disparities in performance are significant enough to be easily discerned through human observation. We primarily focused on three key metrics: settling time, overshoot, and steady-state error, all of which vividly demonstrated the effectiveness of our controller. Table 3 conveniently summarizes these metrics, unequivocally highlighting the superiority of our controller. It is worth noting, however, that the LQR controller for the lower coupling system still encountered challenges when dealing with disturbances and uncertainties. Even though LQR is a popular control method, it is most suitable for non-coupling linear systems to maintain robustness, as its fixed gains are based on a linear model. In contrast, SN and TSN controllers outperform LQR due to their adaptive and predictive capabilities, while still retaining the advantages of LQR. TSN ranks as the top performer, followed by SN and LQR.

Table 3. Comparative analysis of rotary inverted pendulum with LQR, SN, and TSN Controllers.

Controller	Overshoot	Settling Time	Steady-State Error
LQR controller	high	15 s	5 degrees
SN controller	medium	10 s	near 0 degrees
TSN controller	low	6 s	0 degrees

5.3. Experiment of Arm-Driven Inverted Pendulum

In the context of arm-driven inverted pendulum control, it is evident from Section 4.3 that the LQR controller exhibited oscillations, while the SN controller displayed significant overshoot. During the implementation phase, it became clear that the LQR controller was inadequate for executing arm-driven inverted pendulum control, resulting in continuous pendulum rotation and rendering the system uncontrollable. On the other hand, the SN controller also struggled to attain a stable steady state and eventually lost control after a prolonged period, causing the pendulum to rotate continuously. This issue could be attributed to the limited robustness of the steady-state component, leading to divergence in the presence of minor disturbances. These observations highlight the limitations of conventional sliding mode control, with possible factors being the neural network's lower predictive accuracy or the hardware's insufficient computational capability. Therefore, it is imperative to address these issues when utilizing the SN controller.

In this paper, we propose a solution to these challenges by incorporating a Tikhonov regularization term into the prediction of neural network weights, which effectively mitigates these problems. Additionally, we adopted the ReLu activation function, which significantly reduced the computational load on the hardware. For a more comprehensive understanding of arm-driven inverted pendulum operation, please refer to Figure 15, which provides a visual representation. Furthermore, Table 4 presents a comparative analysis of the LQR, SN, and TSN controllers.

(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 15. Photographs depicting the stages of arm-driven inverted pendulum experiments with TSN control: (**a**) initial states; (**b**) pendulum decoupling initiation; (**c**) convergence to zero states; (**d**) stable two-state configuration; (**e**) achievement of stable steady-states.

Table 4. Comparative analysis of arm-driven inverted pendulum with LQR, SN, and TSN controllers.

Controller	Overshoot	Settling Time	Steady-State Error
LQR controller	fail	fail	fail
SN controller	large	unstable	unstable
TSN controller	low	16 s	near 0 degrees

In conclusion, the feasibility and superiority of the TSN control methodology have been confirmed through three simulations and two empirical experiments conducted in real-world experimental operations in Sections 4 and 5.

In the two previous experiments, we measured the encoder's angle change using the RTDX sampling time. As illustrated in Figure 16, we observed the pendulum's steady-state behavior as it remained at a stable value of −180 degrees.

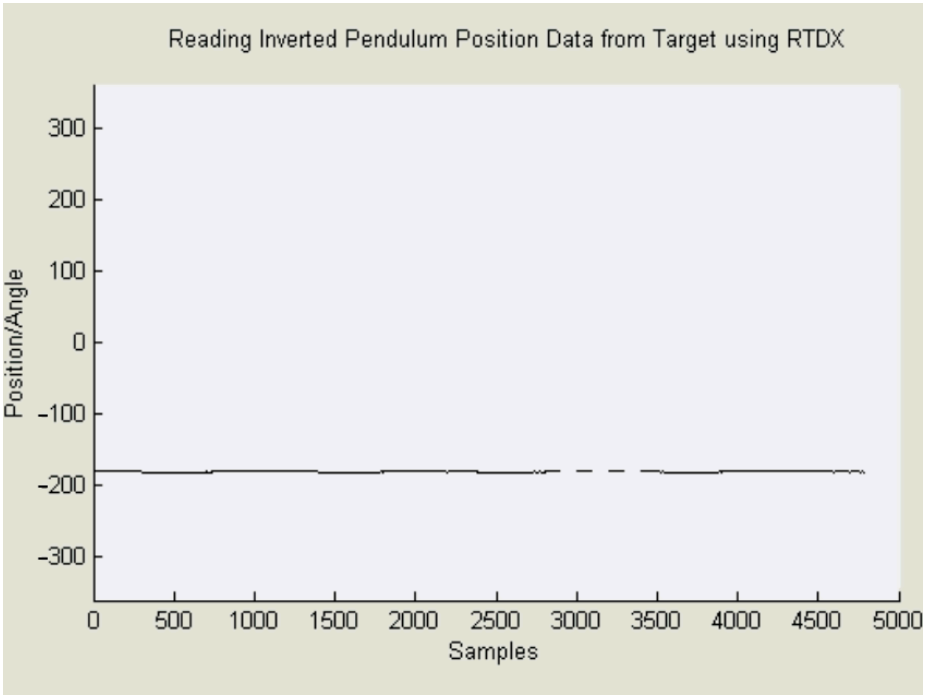


Figure 16. Photographs depicting the pendulum's steady-state behavior at a stable −180 degrees.

Remark 1. *Extensive simulation and practical experiments have confirmed the robust anti-interference capabilities of the TSN controller when applied to the rotary pendulum. In cases where weak coupling is present, relatively lax initial conditions are permissible. Conversely, when handling the arm-driven pendulum, which exhibited significantly stronger coupling, it becomes essential to establish initial conditions within a manageable range to achieve effective decoupling. Additionally, this system is highly sensitive to interference, and excessive interference is not tolerable.*

6. Conclusions

This paper proposed a method that combines the Tikhonov regularization algorithm with sliding neural network control for nonlinear coupled systems. The purpose was to explore methods for stabilizing an inverted pendulum, enabling it to exhibit AI (artificial intelligence)-like stability and predictive capabilities in articulated actuators. Through mathematical analysis, simulations, and practical implementations, the results demonstrate compliance with stability and performance requirements, thus proving the feasibility and superiority of this approach.

The Tikhonov regularization, sliding surface, and neural network methodology proposed in this work differs from recent research in several aspects. Firstly, it provides a comprehensive and integrated approach by combining multiple control methodologies, leading to improved system performance and stability. Secondly, the use of the Tikhonov regularization term helps in effectively adjusting the weights of neural networks and allows for adaptive learning and real-time adjustments to enhance decoupling ability. Thirdly, the incorporation of sliding mode control achieves desired control objectives and enhances the controller's capability of handling uncertainties and disturbances. Overall, this proposed methodology offers a novel and effective solution for decoupling control in the inverted pendulum system, showcasing its superiority and reliable robustness in decoupling.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The author is grateful to the editors and reviewers for their constructive comments and help, which have significantly improved this work.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. Derivation of the LQR

According to the LQR theory, the following derivations are given for defining and minimizing the cost function J as follows:

$$J = \int [\boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R}\boldsymbol{u}]dt \tag{A1}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ represent the state weight matrix and control weight matrix, respectively, both having a square dimension of m. By solving the following Riccati equation, we can obtain the optimal controller gain matrix $\boldsymbol{K}$, and Equation (A1) becomes as follows:

$$\begin{array}{rl} J & = \int [\boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{K}^T \boldsymbol{R}\boldsymbol{K}\boldsymbol{x}]dt \\ & = \int [\boldsymbol{x}^T (\boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R}\boldsymbol{K})\boldsymbol{x}]dt \end{array} \tag{A2}$$

given that $\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x}$ and assuming the existence of a positive definite matrix $\boldsymbol{P}$ such that:

$$\begin{array}{rl} \frac{d}{dt}\boldsymbol{x}^T \boldsymbol{P}\boldsymbol{x} = & \boldsymbol{x}\,.^T \mathrm{Px} + \boldsymbol{x}^T \boldsymbol{P}\,\boldsymbol{x}\,. = \boldsymbol{x}^T (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})^T \boldsymbol{P}\boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{P}(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x} \\ & = \boldsymbol{x}^T \left(\boldsymbol{A}^T \boldsymbol{P} - \boldsymbol{K}^T \boldsymbol{B}^T \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{K}\right)\boldsymbol{x} \end{array} \tag{A3}$$

Combining (A2) and (A3), we arrive at the following:

$$\boldsymbol{x}^T \left(\boldsymbol{A}^T \boldsymbol{P} - \boldsymbol{K}^T \boldsymbol{B}^T \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{K}\right)\boldsymbol{x} = -\boldsymbol{x}^T (\boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R}\boldsymbol{K})\boldsymbol{x} \tag{A4}$$

This leads to the following equality:

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{K} = \boldsymbol{K}^T\boldsymbol{B}^T\boldsymbol{P} - \boldsymbol{K}^T\boldsymbol{R}\boldsymbol{K} \tag{A5}$$

Now, to eliminate the need for the term $\boldsymbol{R}$ in Equation (A5), let us introduce the following:

$$\boldsymbol{K} = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} \tag{A6}$$

Substituting (A6) into (A5), we derive the following Riccati equation:

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} = 0 \tag{A7}$$

References

1. Tikhonov, A.N.; Arsenin, V.Y. *Solutions of Ill-Posed Problems*; Winston & Sons: Hoboken, NJ, USA, 1977.
2. Engl, H.W.; Hanke, M.; Neubauer, A. Regularization of inverse problems. In *Regularization of Inverse Problems*, 2nd ed.; Springer: Berlin, Germany, 1996.
3. Hansen, P.C. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 1998.
4. Buccini, A.; Pasha, M.; Reichel, L. Generalized singular value decomposition with iterated Tikhonov regularization. *J. Computa. Appl. Mathe.* **2020**, *373*, 112276. [CrossRef]
5. Jiang, J.; Tang, H.; Mohamed, M.S.; Luo, S.; Chen, J. Augmented Tikhonov Regularization Method for Dynamic Load Identification. *Appl. Sci.* **2020**, *10*, 6348. [CrossRef]
6. Yuliansyah, D.R.; Pan, M.-C.; Hsu, Y.-F. Sensor-to-Image Based Neural Networks: A Reliable Reconstruction Method for Diffuse Optical Imaging of High-Scattering Media. *Sensors* **2022**, *22*, 9096. [CrossRef] [PubMed]
7. Qi, J.; Ping, Y.; Wang, M.; Wu, C. Online Trajectory Planning Method for Double-Pendulum Quadrotor Transportation Systems. *Electronics* **2022**, *11*, 50. [CrossRef]
8. Maraslidis, G.S.; Kottas, T.L.; Tsipouras, M.G.; Fragulis, G.F. Design of a Fuzzy Logic Controller for the Double Pendulum Inverted on a Cart. *Information* **2022**, *13*, 379. [CrossRef]
9. Wu, Y., Zhu, P. Fuzzy Control for the Swing-Up of the Inverted Pendulum System. In *Intelligent Computing and Information Science. ICICIS 2011. Communications in Computer and Information Science*; Chen, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 135. [CrossRef]
10. Yang, X.; Zheng, X. Swing-Up and Stabilization Control Design for an Underactuated Rotary Inverted Pendulum System: Theory and Experiments. *IEEE Trans. Ind. Electron.* **2018**, *65*, 7229–7238. [CrossRef]
11. Park, M. S.; Chwa, D. Swing-Up and Stabilization Control of Inverted-Pendulum Systems via Coupled Sliding-Mode Control Method. *IEEE Trans. Ind. Electron.* **2009**, *56*, 3541–3555. [CrossRef]
12. Wang, C.; Jaidaa, A.; Wang, Z.; Lu, L. An Effective Decoupling Control with Simple Structure for Induction Motor Drive System Considering Digital Delay. *Electronics* **2021**, *10*, 3048. [CrossRef]
13. Carpio, M.; Saltaren, R.; Viola, J.; Calderon, C.; Guerra, J. Proposal of a Decoupled Structure of Fuzzy-PID Controllers Applied to the Position Control in a Planar CDPR. *Electronics* **2021**, *10*, 745. [CrossRef]
14. Kang, R.; Liatsis, P.; Kyritsis, D.C. Emission Quantification via Passive Infrared Optical Gas Imaging: A Review. *Energies* **2022**, *15*, 3304. [CrossRef].
15. Kim, K. K. K.; Shen, D. E.; Nagy, Z. K.; Braatz, R. D. Wiener's Polynomial Chaos for the Analysis and Control of Nonlinear Dynamical Systems with Probabilistic Uncertainties [Historical Perspectives]. *IEEE Control. Syst. Mag.* **2013**, *33*, 58–67. [CrossRef]
16. Mon, Y. J.; Lin, C. M. Double Inverted Pendulum Decoupling Control by Adaptive Terminal Sliding-mode Recurrent Fuzzy Neural Network. *J. Intelli. Fuzzy Syst.* **2014**, *26*, 1723–1729. [CrossRef]
17. Pan, J.; Qi, S.; Wang,Y. Flatness based active disturbance rejection control for cart inverted pendulum and experimental study. In Proceedings of the 2015 American Control Conference (ACC), Chicago, IL, USA, 1–3 July 2015; pp. 4868–4873. [CrossRef]
18. Liu, K.; Yang, P.; Wang, R.; Jiao, L.; Li, T.; Zhang, J. Observer-Based Adaptive Fuzzy Finite-Time Attitude Control for Quadrotor UAVs. *IEEE Trans. Aerospace Electron. Syst.* **2023**, 1–17. [CrossRef]
19. Chiu, C. –H.; Hung, Y. –T.; Peng, Y. –F. Design of a Decoupling Fuzzy Control Scheme for Omnidirectional Inverted Pendulum Real-World Control. *IEEE Access* **2021**, *9*, 26083–26092. [CrossRef]
20. Liu, K.; Wang, R. Antisaturation Adaptive Fixed-Time Sliding Mode Controller Design to Achieve Faster Convergence Rate and Its Application. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 3555–3559. [CrossRef]
21. Hou, M.; Zhang, X.; Chen, D.; Xu, Z. Hierarchical Sliding Mode Control Combined with Nonlinear Disturbance Observer for Wheeled Inverted Pendulum Robot Trajectory Tracking. *Appl. Sci.* **2023**, *13*, 4350. [CrossRef]
22. Liu, K.; Wang, R.; Zheng, S.; Dong, S.; Sun, G. Fixed-time disturbance observer-based robust fault-tolerant tracking control for uncertain quadrotor UAV subject to input delay. *Nonlinear Dyn.* **2022**, *107*, 2363–2390. [CrossRef]
23. Tomasiello, S.; Pedrycz, W.; Loia, V. On Fractional Tikhonov Regularization: Application to the Adaptive Network-Based Fuzzy Inference System for Regression Problems. *IEEE Trans. Fuzzy Syst.* **2022**, *30*, 4717–4727. [CrossRef]

24. Molina, C.; Martinez, J.; Giraldo, E. Dynamic Tikhonov State Forecasting Based on Large-Scale Deep Neural Network Constraints. *Eng. Proc.* **2023**, *39*, 28. [CrossRef]
25. Slotine, J.J.; Li, W. *Applied Nonlinear Control*; Prentice Hall: Upper Saddle River, NJ, USA, 1991.
26. Burqan, A.; Sarhan, A.; Saadeh, R. Constructing Analytical Solutions of the Fractional Riccati Differential Equations Using Laplace Residual Power Series Method. *Fractal Fract.* **2023**, *7*, 14. [CrossRef]
27. Hassan, K. K. *Nonlinear Systems*, 3rd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2002.
28. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on Machine Learning (ICML), Haifa, Israel, 21–24 June 2010.
29. TeraSoft Inc. *EMECS (Electro-Mechanical Engineering Control System) User's Manual*; TeraSoft Inc.: Taipei, Taiwan, 2004.
30. Texas Instruments. *TMS F28xx Technical Reference Book*; Texas Instruments: Dallas, TX, USA, 2004.
31. Mathworks Inc. *MATLAB R14 User's Manual*; Mathworks Inc.: Netic, MA, USA, 2004.