robotics

MDPI

Article

Closed-Form Continuous-Time Neural Networks for Sliding Mode Control with Neural Gravity Compensation

Claudio Urrea *, Yainet Garcia-Garcia and John Kern

Electrical Engineering Department, Faculty of Engineering, University of Santiago of Chile, Las Sophoras 165, Estación Central, Santiago 9170020, Chile; yainet.garcia@usach.cl (Y.G.-G.); john.kern@usach.cl (J.K.)
* Correspondence: claudio.urrea@usach.cl; Tel.: +56-2-27-183-350

Abstract: This study proposes the design of a robust controller based on a Sliding Mode Control (SMC) structure. The proposed controller, called Sliding Mode Control based on Closed-Form Continuous-Time Neural Networks with Gravity Compensation (SMC-CfC-G), includes the development of an inverse model of the UR5 industrial robot, which is widely used in various fields. It also includes the development of a gravity vector using neural networks, which outperforms the gravity vector obtained through traditional robot modeling. To develop a gravity compensator, a feedforward Multi-Layer Perceptron (MLP) neural network was implemented. The use of Closed-Form Continuous-Time (CfC) neural networks for the development of a robot's inverse model was introduced, allowing efficient modeling of the robot. The behavior of the proposed controller was verified under load and torque disturbances at the end effector, demonstrating its robustness against disturbances and variations in operating conditions. The adaptability and ability of the proposed controller to maintain superior performance in dynamic industrial environments are highlighted, outperforming the classic SMC, Proportional-Integral-Derivative (PID), and Neural controllers. Consequently, a high-precision controller with a maximum error rate of approximately 1.57 mm was obtained, making it useful for applications requiring high accuracy.

Keywords: closed-form continuous-time; gravity compensation; inverse model; sliding mode control; trajectory tracking

check for updates

Citation: Urrea, C.; Garcia-Garcia, Y.; Kern, J. Closed-Form Continuous-Time Neural Networks for Sliding Mode Control with Neural Gravity Compensation. *Robotics* **2024**, *13*, 126. https://doi.org/10.3390/robotics13090126

Academic Editor: Sunan Huang

Received: 4 July 2024
Revised: 9 August 2024
Accepted: 21 August 2024
Published: 23 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Precise control of industrial robots has become a critical factor in improving efficiency and quality in productive sectors, especially when aiming to gain autonomy. Modern robotic systems face increasingly complex, dynamic, and variable environments, requiring controllers capable of rapidly adapting and maintaining high precision under diverse operating conditions. Hence, the evolution of robotic control has led to the exploration of advanced techniques to address these challenges.

In this context, the use of trajectories occupies an important place, as it represents the translation of tasks that the robot must perform in the work environment. Trajectory tracking is a fundamental task that requires controllers capable of handling complex and nonlinear dynamics. This progression reflects the need to overcome the limitations of traditional approaches and leverage computational and machine learning advances to enhance robot performance in demanding industrial applications.

Related Work

The use of classical control strategies such as the Proportional-Integral-Derivative (PID) controller has been a standard solution in industry due to its simplicity, low cost, and easy implementation. However, these controllers, although effective in certain scenarios, present significant limitations when faced with variable conditions and unknown or highly nonlinear systems. Their precision decreases in the presence of changes in the operating

point, dynamic system variations, temporal delays, and other aspects that limit their performance [1]. These limitations have motivated the search for advanced approaches to robotic control. In this context, Sliding Mode Control (SMC) emerges as a promising solution.

SMC stands out as a robust, nonlinear control scheme that dynamically adapts its parameters to effectively handle uncertainties and disturbances in robotic systems. Unlike PID controllers, SMC can maintain high performance across a wide range of operating conditions, making it particularly suitable for robotic applications where precision and adaptability are crucial. However, despite its high performance demonstrated in academic and research settings, SMC presents as its main disadvantage the presence of high-frequency oscillations known as "chattering" in the control signal, which in practice can damage actuators, thus limiting its implementation at an industrial level [1].

To address this, improvements to SMC have been proposed to mitigate this phenomenon. Adaptive approaches have been addressed in various ways, as they allow adjustment to changing conditions and appropriate responses to model uncertainties. In [2], Fuzzy Logic Control (FLC) and SMC are combined with the aim of smoothing the control action and reducing chattering, reporting better performance than conventional SMC. In [3], an SMC controller, Fractional-Order Proportional Integral Derivative (FOPID) controller, and FLC controller are integrated. while maintaining robustness.

Shen et al. [4] have proposed several solutions for the control of Neutral-Point-Clamped (NPC) converters using SMC. In [4], they developed a cascade control scheme combining Adaptive Sliding Mode Control (ASMC) and a Nonlinear High-Gain Observer (HGO), thus improving immunity to measurement noise and disturbance rejection. In [5], they propose an Adaptive Second-Order SMC using an adaptive-gain Generalized Super-Twisting Algorithm (GSTA) and a modified Super-Twisting Observer (STO), reporting better disturbance rejection compared with the use of linear observers as well as a good compromise between dynamic performance and chattering reduction. In [6], several relevant aspects are proposed to improve convergence rate, dynamic performance, abrupt disturbance rejection, and robustness to parameter variations in NPC converter systems, such as a Higher Order Sliding-Mode Observer (HOSMO)-based varying exponent gain Super-Twisting Algorithm (STA) (HOSMO-VEGSTA) control scheme.

In [7], Integral SMC (ISMC) is combined with an Adaptive Super Twisting Observer (ASTO) to improve the control of robotic manipulators for trajectory tracking in the presence of uncertainties and disturbances. The ASTO allows estimation and compensation of uncertainties in finite time, and its adaptive gains enable suppression of the chattering phenomenon compared with fixed-gain observers. In [8], they combine SMC with an Extended State Observer (ESO) for disturbances. Their approach integrates a PID sliding surface with a saturation function-based ESO to estimate and reject disturbances, achieving reduced tracking errors and sensitivity to low-frequency disturbances.

Abadi et al. [9] implement a Sliding Active Disturbance Rejection Control for wheeled mobile robots. To mitigate the effects caused by wheel slip uncertainties and wind disturbances, an SMC control with a boundary layer around the switching surface is applied, implementing a continuous control law. Additionally, Active Disturbance Rejection Control (ADRC) is integrated to estimate and eliminate uncertainties within the feedback loop using an ESO. A similar strategy is proposed in [10]. Another aspect addressed by researchers is systems with time-varying delays, which can occur, for example, in teleoperated systems such as those used in mining or robotic surgery. Chang et al. [11] developed an adaptive sliding mode control scheme that can handle variable delays and unknown bounded disturbances.

In [12], an Adaptive Sliding-Mode-Assisted Disturbance Observer (ASMADO) is proposed for inertially stabilized platforms with a spherical mechanism. This combines ISMC, an adaptive disturbance observer, and gravity compensation. These strategies allow for estimation and compensation of unknown disturbances and model uncertainties, reduction of the magnitude of discontinuous control action, and adequate mitigation of chattering effects while maintaining control robustness. Gravity compensation is a

frequently applied strategy, as it leads to smoother and more efficient system control, reducing actuator effort. In [13], a Model Predictive Control (MPC) was complemented with gravity compensation, which was implemented using the neural inverse model of belt drive transmissions and motors, as gravity compensation based on the gravity vector obtained through robot modeling proved too aggressive for the actuators.

As can be observed, appropriate modeling and identification of disturbance dynamics and the system in general are crucial for obtaining efficient, robust control, with extensive use of ESO and other identification techniques being evident. In [14], they developed a control scheme comprising a super-twisting sliding mode controller and a nonlinear input estimator for a soft robot based on data-driven sparse identification. Meanwhile, in [15], they present a model-free terminal sliding mode control method, incorporating a neural network to approximate the robot's dynamics. In this regard, data-driven controllers are particularly useful in robotic applications, where operating conditions can vary significantly and obtaining an exact model is challenging.

Generally, the inclusion of neural networks has provided efficient solutions in robotic control due to their ability to approximate complex functions and adapt to uncertainties. In [16], a Fuzzy Neural Network Sliding Mode Controller (FNNSMC) is applied. This controller integrates a Radial Basis Function (RBF) neural network to estimate model errors in real-time, improving control precision, and a fuzzy gain adjustment to optimize the response and minimize chattering. RBF networks are also employed in [17,18] in conjunction with SMC and other techniques for robust control in trajectory tracking.

The control scheme proposed in [19] combines a Double-Loop Recurrent Neural Network (DLRNN) with a type-2 fuzzy and SMC to achieve efficient and robust control of uncertain robotic systems. In [20], an intelligent Genetic Algorithm (GA)-optimized Adaptive Fuzzy Fractional-Order Sliding Mode Controller (AFFOSMC) is formed with type-2 fuzzy sets. This approach improves the precision of manipulator trajectory tracking with errors less than 2 mm, a critical precision for surgical operations.

Khan et al. [21] present an approach that combines adaptive SMC with an ESO and reinforcement learning for industrial robots. In [22], the design of an adaptive neural network controller using backstepping techniques is presented, incorporating an integral sliding mode surface to reduce steady-state error. In [23], artificial neural network controllers are designed to replace classical velocity controllers in a UR5 robot. Although their neural controller achieved trajectory tracking performance comparable to PID control, the authors highlight the potential for better generalization and adaptation to variations between individual robotic units.

In general, neural network-based approaches show increasing potential for high-precision robotic control, especially in handling nonlinear dynamics and uncertainties in industrial manipulators. However, traditional neural networks present limitations in terms of training data quantity and valuable computational and energy resources, which complicates their applicability in various fields and real-time applications. In response to these limitations, there has been an increase in the study of neural networks inspired by the behavior of biological neurons and the use of Ordinary Differential Equation (ODE) solvers, among other approaches.

However, the training and inference of ODE-based neural networks are slow, which worsens as the complexity of data and tasks increases, as occurs with medical data processing, physical simulations, among others [24]. In this regard, Hasani et al. propose in [24] Closed-Form Continuous-Time (CfC) neural networks as an efficient solution for sequential data processing and time series with short training times, high precision, and low memory consumption. The present work addresses the development of a robust data-driven Sliding Mode Controller using neural networks capable of effectively handling the nonlinearities and uncertainties of the UR5 robot for trajectory tracking. The main contributions of the study are as follows:

- Modeling of UR5 nonlinear dynamics: A data-driven approach was developed that captures the robot's dynamic complexities, solving difficulties associated with model uncertainties and nonlinearities.
- Effective compensation of gravitational effects: An adaptive method was implemented to counteract gravitational forces, improving robot control performance on the vertical axis.
- Chattering reduction: The proposed approach allows mitigation of the chattering problem, thus improving actuator lifespan and the smoothness of robot movements.
- Robustness to external disturbances: A control strategy was designed capable of maintaining stable performance against load variations of up to 1 kg and torque disturbances of 5 Nm, solving the problem of sensitivity to changing conditions in dynamic industrial environments.
- Improved precision in complex trajectory tracking: A maximum error in Cartesian space of approximately 1.57 mm is achieved, thus addressing the challenge of precision in applications requiring high accuracy.
- Real-time performance optimization: The efficient integration of advanced machine learning techniques such as Closed-Form Continuous-Time neural networks allowed the implementation of a control system that adapts to changing conditions and solves problems of computational efficiency and adaptability.

This article is structured as follows: Section 2 describes the theoretical foundations of the implemented neural networks; Section 3 describes the system under study; Section 4 specifies the methodology adopted for the design of the proposed controller; Section 5 presents the experiments and simulations carried out in different scenarios; Section 6 discusses the results obtained; and Section 7 shows the conclusions, limitations, and future lines of research.

2. Neural Networks for Automatic Control Systems

Neural networks are applied in controller design and/or system identification using a black-box or gray-box approach. Feedforward Neural Networks (Multi-Layer Perceptrons, MLP) have been widely used to learn complex input-output mappings, such as the inverse kinematics of robotic arms. However, they present limitations in working with sequential data and time series, which affects their performance in dynamic systems. In contrast, Recurrent Neural Networks (RNNs) are effective in capturing long-term dependencies in sequential data, making them suitable for tasks involving temporal sequences. Nevertheless, they face difficulties in accurately representing the continuous evolution of time and complex nonlinear dynamics.

In response to these limitations, Hasani et al. proposed in [25] the Liquid Time-Constant (LTC) networks, which are characterized by the introduction of "liquid" time constants that adapt to input data, allowing for a more natural and flexible representation. LTCs are limited by their computational cost and prolonged training times, as they depend on numerical solvers of ODE. For this reason, Hasani et al. proposed in [24] Closed-Form Continuous-Time neural networks. CfCs address this limitation by providing a Closed-Form approximation of the LTC solution, eliminating the need for numerical solvers. This work employs both traditional and cutting-edge approaches, such as MLP and CfC neural networks, whose fundamental aspects are addressed below.

2.1. Multi-Layer Perceptron

Artificial neural networks are inspired by the human brain and mimic its functioning through a layered structure, with the aim of performing tasks in a similar manner. The input layer receives the features to be considered from processes or systems, while the output layer corresponds to the desired outputs. Neural networks can have a determined number of intermediate layers, known as hidden layers. Hidden layers can enhance the algorithm, allowing for more complex connections and improving the network's processing capacity and performance. Figure 1a represents the diagram of a MLP neural network,

while Figure 1b outlines the structure of the CfC (https://github.com/raminmh/CfC, accessed on 1 June 2024), inspired by the architecture proposed in [24].

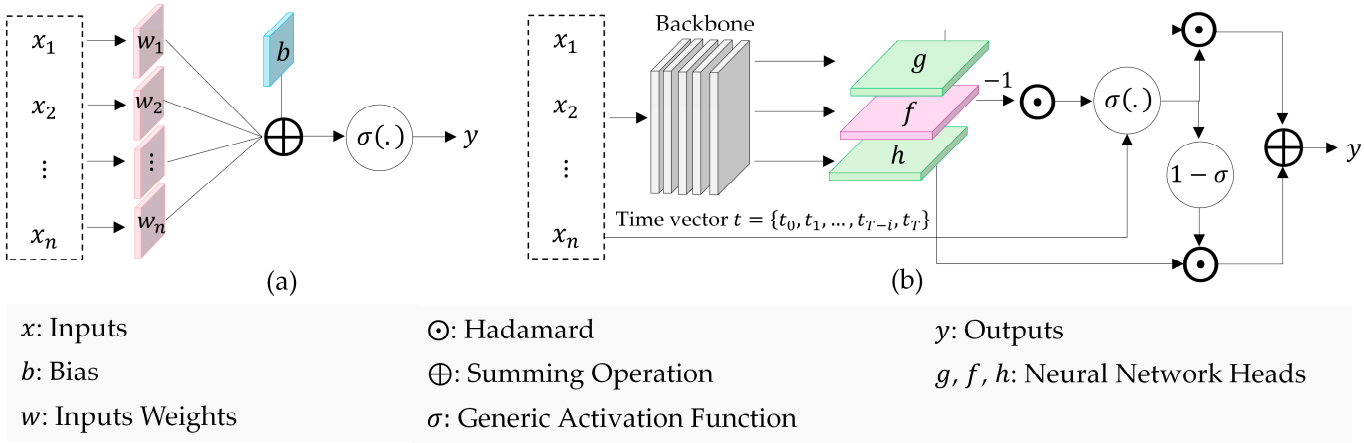


Figure 1. Schematic representation of implemented neural networks. (**a**) Multi-Layer Perceptron. (**b**) Closed-Form Continuous-Time neural networks.

In the MLP, connections are forward, indicating a feedforward network. The output of the neural network is defined as in Equation (1).

$$y = \sigma\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right) \tag{1}$$

2.2. Closed-Form Continuous-Time Neural Networks

The basis of LTC functioning lies in the use of ODE solvers to compute their outputs. The state of a Neural ODE (NODE) can be defined as in Equation (2) [26]:

$$\frac{dx(t)}{dt} = f(x(t), \mathrm{I}(t), t, \Phi) \tag{2}$$

where $x(t) \in \mathbb{R}^D$ denotes the hidden state of the network, $\mathrm{I}(t)$ is the input, and f is parameterized by Φ.

Continuous-Time Recurrent Neural Networks (CT-RNN) introduce a formulation where the dynamics of the hidden state are influenced by a time constant ϱ that regulates the speed at which the system adapts to changes in the hidden state. This is represented by the inclusion of the term $-x(t)/\varrho$, as expressed in (3), which acts as a stability mechanism that helps the system reach an equilibrium state.

$$\frac{dx(t)}{dt} = -\frac{x(t)}{\varrho} + f(x(t), \mathrm{I}(t), t, \Phi) \tag{3}$$

Combining the concepts of NODEs and the time constant ϱ, Hasani et al. propose an alternative formulation called CT-RNN in [25]. The dynamics of the hidden state are defined as in (4), where $\mathrm{S}(t)$ represents the following nonlinearity defined as in (5) with parameters Φ and A.

$$\frac{dx(t)}{dt} = -\frac{x(t)}{\varrho} + \mathrm{S}(t) \tag{4}$$

$$\mathrm{S}(t) = f(x(t), \mathrm{I}(t), t, \Phi)(A - x(t)) \tag{5}$$

where $\mathrm{S}(t) \in \mathbb{R}^J$ indicates that at each time instant t, the signal S is a vector of J real numbers.

With the introduction of S(t), the formulation of a CT-RNN with a variable "liquid" time constant, coupled to its hidden state, termed Liquid Time-Constant Recurrent Neural Networks, is obtained, as defined in Equation (6):

$$\frac{dx(t)}{dt} = -\left[\frac{1}{\varrho} + f(x(t), \mathrm{I}(t), t, \Phi)\right] x(t) + f(x(t), \mathrm{I}(t), t, \Phi) A \tag{6}$$

LTCs are distinguished by their neural function f, which determines both the derivative of the hidden state and a liquid time constant dependent on the input, allowing for dynamic adaptation to the changing characteristics of real-time data. LTCs are flexible in their implementation, compatible with various ODE solvers. This unique structure provides them with greater expressiveness and stability compared with traditional ODE models, making them particularly suitable for time-series modeling tasks.

However, LTCs present limitations regarding computational cost and prolonged training times due to their reliance on numerical solvers for operation. For this reason, Hasani et al. propose an evolution of LTCs, called Closed-Form Continuous-Time neural networks, in [24]. Building on the formulation presented in (6), the hidden state of an LTC recurrent neural network is determined by the general expression (7):

$$\frac{dx(t)}{dt} = -\left[w_\varrho + f(x, \mathrm{I}, \Phi)\right] \odot x(t) + A \odot f(x, \mathrm{I}, \Phi) \tag{7}$$

where $x^{D\times 1}(t)$ defines the hidden state of an LTC layer at a time step t with D cells, $\mathrm{I}^{m\times 1}(t)$ is an exogenous input to the system with m features, ${w_\varrho}^{D\times 1}$ is a time-constant parameter vector, $A^{D\times 1}$ is a bias vector, f is a neural network parameterized by Φ, and $\odot$ is the Hadamard product that allows element-wise operations on multidimensional tensors. The dependence of $f(\cdot)$ on $x(t)$ denotes the possibility of having recurrent connections.

A closed-form approximation of Equation (7) was sought, arriving at Expression (8):

$$x(t) \approx (x_0 - \mathrm{A}) e^{-[w_\varrho + f(\mathrm{I}(t), \Phi)]t} \, f(-\mathrm{I}(t), \Phi) + A \tag{8}$$

To improve gradient properties and training stability, the following modifications were made:

- The exponential decay term $e^{-[w_\varrho + f(\mathrm{I}(t), \Phi)]t}$ was replaced by $\sigma\left(-f\left(x, \mathrm{I}; \Phi_f\right)t\right)$, where σ is an inverted sigmoidal function. This substitution provides a smoother transition and enhances gradient properties during training.
- Two new functions, g and h, were introduced to replace and expand the terms $(x_0 - A)$ and A, respectively. This allows for greater flexibility in modeling system dynamics. Here, $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are neural networks parameterized by Φ_f, Φ_g, and Φ_h, respectively.
- A continuous-time gating mechanism was implemented using $\sigma\left(-f\left(x, \mathrm{I}; \Phi_f\right)t\right)$ and its complement $\left[1 - \sigma\left(-\left[f\left(x, \mathrm{I}; \Phi_f\right)\right]t\right)\right]$, such that the sigmoidal temporal decay function acts as a gating mechanism interpolating between the two limits of $t \to -\infty$ and $t \to \infty$ of the ODE trajectory, controlling the flow of information over time.

These modifications resulted in the CfC formulation present in (9):

$$y = \sigma\left(-f\left(x, \mathrm{I}; \Phi_f\right)t\right) \odot g(x, \mathrm{I}; \Phi_g) + \left[1 - \sigma\left(-\left[f\left(x, \mathrm{I}; \Phi_f\right)\right]t\right)\right] \odot h(x, \mathrm{I}; \Phi_h) \tag{9}$$

The CfC architecture implements a shared backbone that branches into the functions f, g, and h. This design allows the network to learn shared representations while independently exploring temporal and structural dependencies. With this architecture, it retains the universal approximation capability of LTCs while improving training capacity and model flexibility without the need for numerical ODE solvers, significantly accelerating both training and inference. Empirical studies have demonstrated that CfCs can be up

to 100 times faster in training and inference compared with LTCs while maintaining or even improving accuracy in time series modeling tasks. This CfC formulation represents a significant advancement in the field of Continuous-Time neural networks, offering a balance between the expressiveness of ODE-based models and the computational efficiency required for large-scale practical applications [24].

The use of activation functions and loss functions in the training of neural networks plays a crucial role in their performance. Activation functions enable the network to learn and model nonlinear relationships between input and output data. Commonly used activation functions in neural networks include the hyperbolic tangent sigmoid function, defined as in (10). MLPs frequently employ the pure linear function (purelin) in the output layer, defined in (11), which is a linear activation function suitable for regression tasks where a continuous and unbounded output is desired.

$$\text{tansig}(x) = \tanh(x) = \frac{2}{(1 - e^{-2x})} - 1 \tag{10}$$

$$\text{purelin}(x) = x \tag{11}$$

The swish activation function is widely used in image work and has also shown good performance in training various types of neural networks in general, particularly in its Sigmoid Linear Unit (SiLU) variant, as defined in (12) [27]. In the case of loss functions commonly used in neural network training, the Mean Squared Error (MSE) is defined in (13), which measures the average squared difference between the values predicted by the network and the actual values.

$$\text{SW}(x) = \frac{x}{(1 + e^{-x})} \tag{12}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

where n represents the number of examples in the dataset, y_i is the actual value of the i-th example, and $\hat{y}_i$ is the value predicted by the network for the i-th example.

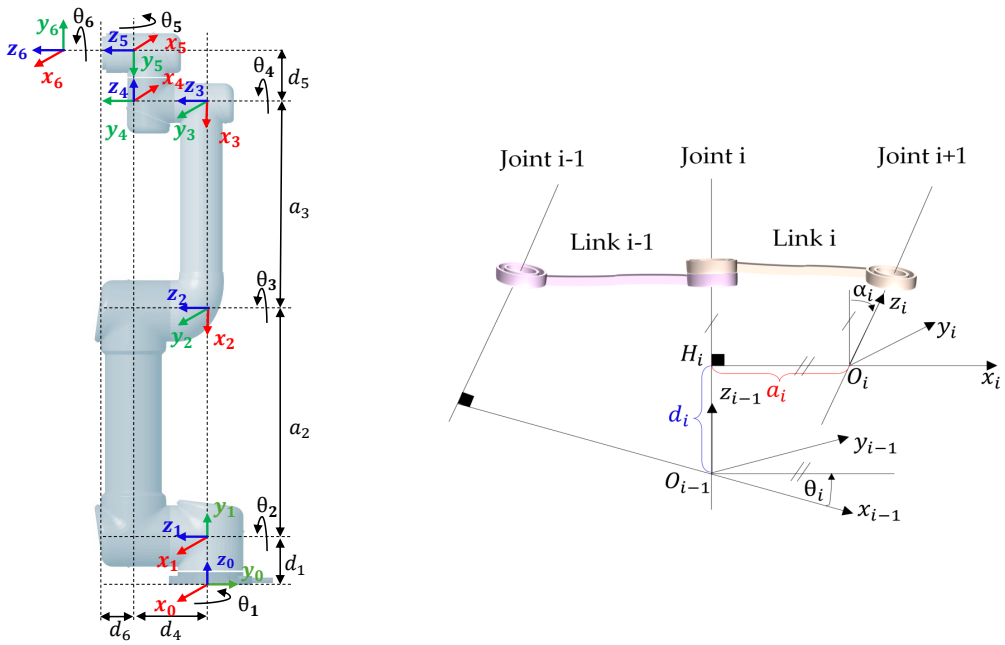
3. Description of the System under Study

This study utilizes MatLab® R2024a software and Python 3.9 on a Lenovo AMD Ryzen 9 5900HX 3.3 GHz laptop with 32 GB of RAM and an NVIDIA GeForce RTX 3080 GPU as the hardware. The robot selected to validate the proposed control strategy is the UR5. This is a collaborative robot from Universal Robots, which has been used in numerous applications due to its features and capabilities. The UR5 has 6 Degrees of Freedom (DoF), an open structure specially adapted for flexible use, various orientations, and extensive communication possibilities with external systems. It can operate in demanding work environments as well as in clean rooms, such as medical settings. It has a payload capacity of 5 kg, a reach of 850 mm, and a repeatability of ± 0.1 mm [28].

3.1. UR5 Kinematic Model

Figure 2a shows the dimensions of the UR5 robot and the reference frames of each link according to the Denavit-Hartenberg (D–H) algorithm for the home position (0, −90, 0, −90, 0, 0), while Figure 2b displays the D–H representation.

Table 1 presents the D–H parameters. In this context, i represents the joint number, θ_i represents the angle to rotate around the z-axis to make two consecutive x-axes parallel, d_i represents the distance measured along the z-axis to align two consecutive x-axes, a_i represents the distance measured along the x-axis that must be moved to align the new $\{O_{i-1}\}$, with $\{O_i\}$, while α_i denotes the angle to rotate around the x-axis to fully align the new $\{O_{i-1}\}$ with $\{O_i\}$.



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

Figure 2. UR5 robot modeled in Simscape. (**a**) Robot reference frames according to the D–H algorithm; (**b**) D–H representation.

Table 1. D–H parameters of the UR5 robot.

Kinematics	θ_i [rad]	d_i [m]	a_i [m]	α_i [rad]
Joint 1	θ_1	0.0892	0	$\pi/2$
Joint 2	θ_2	0	−0.4250	0
Joint 3	θ_3	0	−0.3923	0
Joint 4	θ_4	0.1092	0	$\pi/2$
Joint 5	θ_5	0.0947	0	$-\pi/2$
Joint 6	θ_6	0.0823	0	0

Through the transformation matrices T_i^{i-1}, both the position and orientation of the system i relative to the system $i-1$ are related, which is obtained using Equation (14). The general homogeneous transformation matrix relating the end-effector to the robot base in terms of the joint coordinates is determined by Equation (15).

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cdot \cos\alpha_i & \sin\theta_i \cdot \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cdot \cos\alpha_i & -\cos\theta_i \cdot \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

$$T_6^0 = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdot T_4^3 \cdot T_5^4 \cdot T_6^5 \tag{15}$$

Inverse Kinematics

Determining the inverse kinematics of the robot involves finding the joint coordinates that position and orient the end-effector according to a desired spatial location, allowing for the inverse calculation of each joint angle θ_i. The inverse kinematics solution implemented in this work is based on the one presented in [29]. The coordinate origin vector from joint axis v to joint axis u can be defined as in (16). Determining the transformation matrix allows

for the calculation of the end-effector position relative to the base coordinates P_6^0, with P_5^0 being determined as in Equation (17).

$$P_v^u = \begin{bmatrix} (P_v^u)_x \\ (P_v^u)_y \\ (P_v^u)_z \\ 1 \end{bmatrix} \tag{16}$$

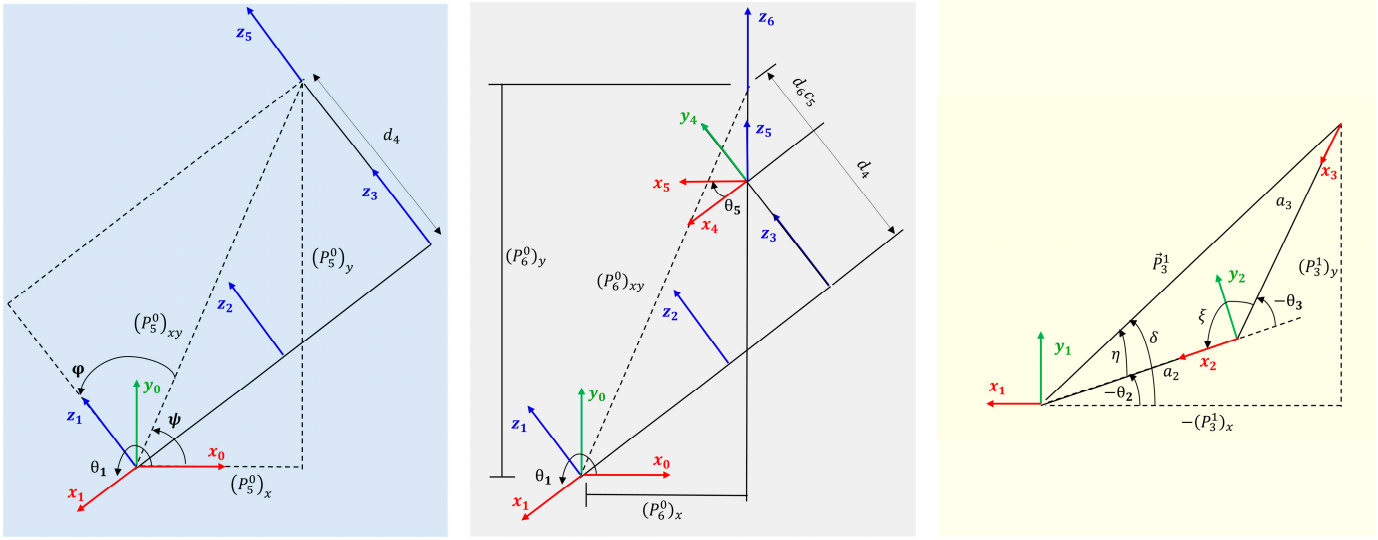
$$P_5^0 = P_6^0 + T_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} \tag{17}$$

The parameter θ_1 is determined by Equation (18), with the variables ψ and φ defined in Equations (19) and (20), respectively, as shown in Figure 3a.

$$\theta_1 = \varphi + \psi + \frac{\pi}{2} \tag{18}$$

$$\psi = \text{atan } 2\left(\frac{\left(P_5^0\right)_y}{\left(P_5^0\right)_x}\right) \tag{19}$$

$$\varphi = \pm\cos^{-1}\left(\frac{d_4}{\sqrt{\left(\left(P_5^0\right)_x\right)^2 + \left(\left(P_5^0\right)_y\right)^2}}\right) \tag{20}$$



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

Figure 3. Planar projection of the axis coordinates. (**a**) Planar projection of the fifth axis coordinate to the base coordinate; (**b**) Planar projection of the sixth axis coordinate to the base coordinate; (**c**) Planar projection of the fifth axis to the first axis joint coordinate [29].

The location of the parameter θ_5 can be seen in Figure 3b and is determined by Equation (21), with P_6^1 obtained as in Equation (22).

$$\theta_5 = \pm\cos^{-1}\left(\frac{\left(P_6^1\right)_z - d_4}{d_6}\right) \tag{21}$$

$$P_6^1 = \left(T_6^0\right)^{-1} P_6^0 \tag{22}$$

The matrix T_6^1 can be obtained as in (23) and allows for the calculation of θ_6 as in Equation (24). Parameters θ_3 and θ_2 are determined using Equations (25) and (26), respectively, as shown in Figure 3c, with P_3^1 defined in Equation (27).

$$T_6^1 = \left(T_1^0\right)^{-1} T_6^0 = \begin{bmatrix} . & . & . & . \\ . & . & . & . \\ \alpha & \beta & . & . \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

$$\theta_6 = \text{atan } 2\left(\frac{-\beta}{\alpha}\right) \tag{24}$$

$$\theta_3 = \pm\cos^{-1}\left(\frac{\left\|P_3^1\right\|^2 - (a_2)^2 - (a_3)^2}{2 \cdot a_2 \cdot a_3}\right) \tag{25}$$

$$\theta_2 = -\text{atan } 2\left(\frac{\left(P_3^1\right)_y}{-\left(P_3^1\right)_x}\right) + \sin^{-1}\left(\frac{a_3 \cdot \sin\theta_3}{\left\|P_3^1\right\|}\right) \tag{26}$$

$$P_3^1 = P_4^1 + T_4^1 \cdot \begin{bmatrix} 0 \\ -d_4 \\ 0 \\ 1 \end{bmatrix} \tag{27}$$

The parameter θ_4 is determined by Equation (28), with ρ and μ being elements of the matrix T_4^3 defined in (29).

$$\theta_4 = \text{atan } 2\left(\frac{\mu}{\rho}\right) \tag{28}$$

$$T_4^3 = \left(T_3^1\right)^{-1} T_4^1 = \begin{bmatrix} \rho & 0 & . & 0 \\ \mu & 0 & . & 0 \\ 0 & 1 & 0 & . \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{29}$$

3.2. UR5 Dynamic Model

The dynamics governing the behavior of an n-DoF robotic manipulator can be formulated as in Equation (30) [13].

$$\tau = M(q)\ddot{q} + C\left(q, \dot{q}\right) + G(q) + F\left(\dot{q}\right) \tag{30}$$

where τ represents the torque vector with dimensions $(n \times 1)$, M represents the inertia matrix with dimensions $(n \times n)$, $C\left(q, \dot{q}\right)$ is the vector of centrifugal and Coriolis forces with $(n \times 1)$, $G(q)$ is the gravitational forces vector $(n \times 1)$, and $F\left(\dot{q}\right)$ represents the friction. The variables q, $\dot{q}$, and $\ddot{q}$ represent the joint position, velocity, and acceleration, respectively. The dynamic model of the UR5 robot is available at the following link (https://drive.google.com/drive/folders/1Or-8sECzHhKaKkuDFRKfJH5vES0eoyC4?usp=drive_link). It is based on the modeling presented in [30], and the corresponding code is available at (https://github.com/kkufieta/ur5_modeling_force_estimate, accessed on 1 June 2024).

In the case of the UR5 robot implementation used in the study, it was modeled in SolidWorks and introduced into the Simscape simulation environment of MatLab®. Table 2 summarizes the robot's dynamics and physical parameters defined in its design.

Table 2. UR5 design parameter [28].

Dynamics	Mass [kg]	Center of Mass [m]	Motion Range	Maximum Speed	Torque
Link 1	3.7	[0, −0.02561, 0.00193]	±360°	±180°/s	150 Nm
Link 2	8.393	[0.2125, 0, 0.11336]	±360°	±180°/s	150 Nm
Link 3	2.33	[0.15, 0, 0.0265]	±360°	±180°/s	150 Nm
Link 4	1.219	[0, −0.0018, 0.01634]	±360°	±180°/s	28 Nm
Link 5	1.219	[0, −0.0018, 0.01634]	±360°	±180°/s	28 Nm
Link 6	0.1879	[0, 0, −0.001159]	±360°	±180°/s	28 Nm

4. Design of Control Strategy

The following section outlines the theoretical aspects of the methodology used for designing the proposed control strategy.

4.1. Sliding Mode Control Based on Closed-Form Continuous-Time Neural Networks with Gravity Compensation

Sliding Mode Control involves forcing the system trajectories to remain on a sliding surface, which allows avoiding undesired behaviors and provides stability. The sliding surface defined in this study is represented in (31).

$$s = \lambda \cdot e + \dot{e} \tag{31}$$

where s is the sliding surface, $\lambda > 0$ is a positive constant that adjusts the convergence of the system, e represents the position error calculated as shown in (32), while $\dot{e}$ represents the velocity error, and can be obtained using Equation (33).

$$e = q_d - q \tag{32}$$

$$\dot{e} = \dot{q}_d - \dot{q} \tag{33}$$

where q_d is the desired joint position vector, q is the current joint position vector, $\dot{q}_d$ is the desired joint velocity vector, and $\dot{q}$ represents the current joint velocity vector.

On the established sliding surface, considering e ensures that the position converges, while considering $\dot{e}$ allows controlling the convergence rate and smoothing the system's response. From Equation (31), it can be observed that if $s \to 0$, then $e \to 0$ and $\dot{e} \to 0$. The final torque vector τ to be applied to the robot can be obtained using the relationship (34).

$$\tau = \tau_{sw} + \tau_{eq} + \tau_g \tag{34}$$

where τ_{sw} is the SMC control torque vector, τ_{eq} is the equivalent torque vector obtained using the robot's inverse model, while τ_g is the gravity compensation torque vector.

The torque τ_{sw} is determined as in Expression (35). For this, the saturation function defined in (36) is implemented, which smooths the control action near the sliding surface, helping to reduce chattering while maintaining the robustness of the controller. The variable K represents the positive gain matrix and ϕ represents the boundary layer thickness.

$$\tau_{sw} = -\mathrm{K} sat\left(\frac{s}{\phi}\right) \tag{35}$$

$$\mathrm{sat}\left(\frac{s}{\phi}\right) \begin{cases} \mathrm{sgn}(s), & |s| > \phi \\ \frac{s}{\phi}, & |s| \leq \phi \end{cases} \tag{36}$$

The function $\mathrm{sgn}(s)$ is defined as in (37). The torque τ_{eq}, since it is obtained through a neural network, can be defined in a simplified way as in (38) [31].

$$\operatorname{sgn}(s)\begin{cases} 1, & s > 0 \\ 0, & s = 0 \\ -1, & s < 0 \end{cases} \tag{37}$$

$$\tau_{eq} = \mathrm{W}^{T}\Theta(\mathrm{X}) \tag{38}$$

where X is the input vector, W is a learnable weight matrix of the CfC network, and $\Theta(\mathrm{X})$ is the internal state vector of the network.

This formulation allows the CfC network to learn and adapt to the robot's full dynamics, including the effects of inertia, Coriolis, centrifugal, and other nonlinearities. The torque τ_g can be obtained as in (39).

$$\tau_g = \mathrm{V}^{T}\sigma\left(\mathrm{U}^{T}q + b\right) + c \tag{39}$$

where V is the output layer weight matrix of the MLP network, U is the input-to-hidden layer weight matrix, b is the hidden layer bias vector, c is the output layer bias vector, and $\sigma(\cdot)$ is the generic activation function.

Stability Analysis

Consider the candidate Lyapunov function:

$$\mathrm{L} = \left(\frac{1}{2}\right)s^{\mathrm{T}}\mathrm{M}(\mathrm{q})\mathrm{s} + \left(\frac{1}{2}\right)\mathrm{t_r}\left\{\widetilde{\mathrm{W}}^{\mathrm{T}}\Gamma^{-1}\widetilde{\mathrm{W}}\right\} + \left(\frac{1}{2}\right)\mathrm{t_r}\left\{\widetilde{\mathrm{V}}^{\mathrm{T}}{\Gamma_{\mathrm{V}}}^{-1}\widetilde{\mathrm{V}}\right\} + \left(\frac{1}{2}\right)\mathrm{t_r}\left\{\widetilde{\mathrm{U}}^{\mathrm{T}}{\Gamma_{\mathrm{U}}}^{-1}\widetilde{\mathrm{U}}\right\} \tag{40}$$

In this equation, the following are true:

- $\mathrm{M}(\mathrm{q})$ is the robot's inertia matrix;
- $\widetilde{\mathrm{W}} = \mathrm{W}^{*} - \mathrm{W}$, with W being the weight matrix of the CfC network;
- $\widetilde{\mathrm{V}} = \mathrm{V}^{*} - \mathrm{V}$, $\widetilde{\mathrm{U}} = \mathrm{U}^{*} - \mathrm{U}$, with V and U being the weight matrices of the MLP network;
- Γ, Γ_V, and Γ_U, are positive gain matrices.

The time derivative of $\mathrm{L}(\dot{L})$ is expressed as in (41):

$$\dot{\mathrm{L}} = s^{\mathrm{T}}\mathrm{M}(\mathrm{q})\dot{s} + \left(\frac{1}{2}\right)s^{\mathrm{T}}\dot{\mathrm{M}}(\mathrm{q})s - \mathrm{t_r}\left\{\widetilde{\mathrm{W}}^{\mathrm{T}}\Gamma^{-1}\dot{\widetilde{\mathrm{W}}}\right\} - \mathrm{t_r}\left\{\widetilde{\mathrm{V}}^{\mathrm{T}}{\Gamma_{\mathrm{V}}}^{-1}\dot{\widetilde{\mathrm{V}}}\right\} - \mathrm{t_r}\left\{\widetilde{\mathrm{U}}^{\mathrm{T}}{\Gamma_{\mathrm{U}}}^{-1}\dot{\widetilde{\mathrm{U}}}\right\} \tag{41}$$

Substituting the robot dynamics and the control laws, the Expression (42) is obtained:

$$\dot{\mathrm{L}} \leq -s^{\mathrm{T}}\mathrm{Ksat}\left(\frac{\mathrm{s}}{\phi}\right) + s^{\mathrm{T}}\widetilde{\mathrm{W}}^{\mathrm{T}}\Theta(\mathrm{X}) + s^{\mathrm{T}}\left[\widetilde{\mathrm{V}}^{\mathrm{T}}\sigma\left(\mathrm{U}^{\mathrm{T}}\mathrm{q} + \mathrm{b}\right) + \widetilde{\mathrm{U}}^{\mathrm{T}}\mathrm{V}^{\mathrm{T}}\sigma'\left(\mathrm{U}^{\mathrm{T}}\mathrm{q} + \mathrm{b}\right)\mathrm{q}\right] + \varepsilon \tag{42}$$

where ε represents the bounded approximation errors and disturbances of the neural networks used.

The proposed adaptation laws for the networks are defined in (43) and (44) for the CfC and MLP, respectively.

$$\dot{\widetilde{\mathrm{W}}} = -\Gamma\Theta(\mathrm{X})s^{\mathrm{T}} \tag{43}$$

$$\dot{\widetilde{\mathrm{V}}} = -\Gamma_{\mathrm{V}}\sigma\left(\mathrm{U}^{\mathrm{T}}\mathrm{q} + \mathrm{b}\right)s^{\mathrm{T}}\ \dot{\widetilde{\mathrm{U}}} = -\Gamma_{\mathrm{U}}\mathrm{q}\left[\mathrm{V}^{\mathrm{T}}\sigma'\left(\mathrm{U}^{\mathrm{T}}\mathrm{q} + \mathrm{b}\right)\right]^{\mathrm{T}}s^{\mathrm{T}} \tag{44}$$

Substituting these adaptation laws in (41), the expression (45) is obtained:

$$\dot{\mathrm{L}} \leq -s^{\mathrm{T}}\mathrm{Ksat}\left(\frac{s}{\phi}\right) + \varepsilon \tag{45}$$

If the gain matrix K is chosen large enough to dominate ε, then $\dot{\mathrm{L}} < 0$ implies that s converges to a neighborhood of the origin, whose size depends on ε and can be made

arbitrarily small by increasing K, which must be sufficient to dominate not only ε, but also the terms involving $\tilde{W}$, $\tilde{V}$, and $\tilde{U}$ in (42). This analysis demonstrates the practical stability of the closed-loop system, considering the CfC and MLP neural networks.

The methodology for designing, testing, and validating the controller is summarized in Table 3:

Table 3. Experiment methodology pseudocode.

1:	***Procedure***	
2:	Data acquisition UR5 inverse model identification Input data : $q,\ \dot{q},\ \ddot{q},\ \tau(k-1)$ Output data : $\tau(k)$	Data collection for the UR5 robot using chirp signals. The data include joint position, velocity, and acceleration, as well as previous and current torque. This allows capturing the robot's dynamics for training the inverse model of the robot.
3:	Analysis and comparison of system identification techniques using neural networks (MLP, CfC)	Evaluate MLP and CfC neural network architectures to determine which is more effective in identifying the robot's inverse model.
4:	Obtaining the inverse model of the UR5 robot	Train the selected neural network (CfC) with the acquired data to obtain an inverse model of the UR5 robot.
5:	Data acquisition to obtain the gravity vector Signal $(q) \rightarrow \mathrm{G}(q) = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} \rightarrow \tau_g(k) \quad \tau_g(k) = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix}$	Collect specific data to model the effect of gravity in different robot configurations using stacked ellipse trajectories, rose curve trajectories, sinusoidal trajectories, and circular trajectories.
6:	Training and comparison of neural networks for gravity compensation	Train and compare MLP and CfC networks to determine which is more effective in gravity compensation.
7:	Gravity vector of the UR5	Implement the selected neural network (MLP) to estimate the gravity vector in real-time.
8:	Design of the control strategy	Develop the Closed-Form Continuous-Time Neural Networks with Neural Gravity Compensation (SMC-CfC-G) controller by integrating the inverse model and gravity compensation for trajectory tracking.
9:	Definition of control parameters $(s,\ \lambda\ ,\ \mathrm{K},\ \phi)$	Adjust the parameters of the SMC controller, including gains and boundary layer thickness, to reduce chattering.
10:	Implementation of the control strategy (SMC-CfC-G) $\tau = \tau_{sw} + \tau_{eq} + \tau_g$	- In each iteration, read the current position, velocity, and acceleration of the robot. - Estimate the equivalent torque using the CfC inverse model. - Calculate gravity compensation using the MLP. - Determine the control law. - Apply the final control torque to the robot.
11:	Controller performance evaluation	Conduct trajectory tracking and disturbance rejection tests to evaluate controller performance.
12:	Analysis of results and adjustment of parameters, if necessary	Analyze performance indices (ISE, ITAE, and RMSE) and adjust controller parameters as necessary to optimize performance.
13:	***End***	

4.2. System Identification

The uncertain factors in robot dynamics are divided into structural and non-structural uncertainties. Parameter identification errors introduce structural uncertainty. Many factors, such as non-linear joint damping, friction model design methods, and noise, are non-structural uncertainties, and expressing uncertain factors through physical modeling methods is challenging. On the other hand, many manufacturers do not provide, or only partially provide, the robot's dynamic parameters. Simultaneously, due to factors such as manufacturing errors and uneven material distribution, the dynamic parameters of industrial robots of the same model may differ.

Therefore, developing reasonable and applicable torque prediction methods is important for industrial robot modeling. However, due to the complexity of most robots, it is impractical to measure physical parameters directly. Experiments remain the most effective way to obtain dynamic parameters [32]. The use of neural networks, while often a complex and iterative process of trial and error, can achieve adequate identification values, all closely related to the complexity of the plant to be identified. Generally, the success of training depends on the quantity and quality of the data. Some of the frequently suggested excitation signals are sinusoidal signals, pseudo-random or noise-type signals, as well as the use of Fourier series, among others. Figure 4 shows the scheme applied in this research for system identification.

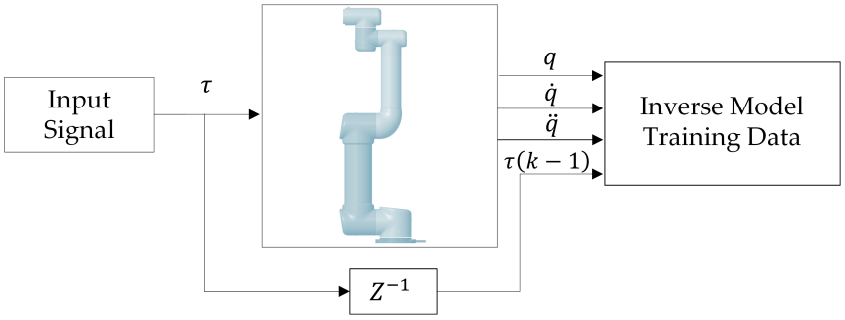


Figure 4. UR5 identification procedure schemes.

The UR5 is a collaborative robot designed for fast and precise movements. Capturing its high-frequency dynamics is essential for robust control in demanding applications. In the context of SMC control, which is susceptible to chattering, an exact inverse model incorporating these dynamics can optimize compensation, alleviating the load on the discontinuous part of the control and attenuating chattering, thus improving the overall performance of the SMC. In response to this, the chirp signal is used to identify the inverse model of the UR5 robot. This signal performs a frequency sweep, allowing a wide spectrum to be stimulated in a single experiment. This approach facilitates a more refined tuning of the SMC, allowing higher gains without compromising stability, which translates into more precise and reliable control. The torque signals applied to each of the joints are shown in Figure 5.

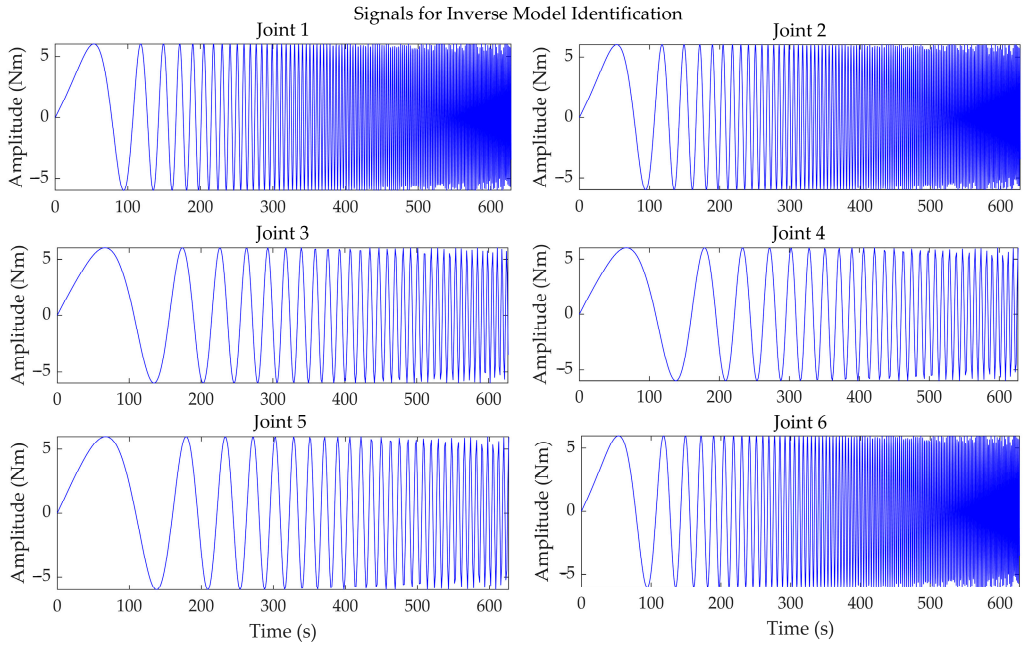


Figure 5. Chirp-type signals used for UR5 identification.

The obtained data are divided into three sets, corresponding to 70% (initial data for training), 20% (intermediate data for testing), and 10% (final data for validation). The sampling time for identification was set at 0.01 s, with the aim of efficiently capturing the plant's dynamics. The simulation time was set at 600 s, taking a total of 60,000 training samples. The MLP neural network training was performed using the Neural Network Start application in MatLab®, which can be accessed with the nnstart command. The CfC neural network was trained in Python 3.9 software.

4.3. Gravity Compensation

Gravity compensation in robotics helps to correct overshoots and asymmetric transient behaviors, improving position control in multi-DoF robots. It is especially beneficial for robotic systems with small actuators that generate less torque [33]. This improves movement precision, reduces actuator effort, and increases the system's energy efficiency. Additionally, it facilitates trajectory tracking and robot stability in various positions. Implementing this compensation neuronally offers additional benefits, such as adaptability to variations in robot parameters, efficient handling of uncertainties and non-linear effects, and greater computational efficiency. Neural networks also allow for smoother integration with data-based approaches, have the potential for continuous improvement, and can be more easily generalized to other robot models or complex systems, such as soft robots.

To counteract the gravitational forces acting on the UR5, a neural gravity compensator is applied. For this, an MLP neural network and a CfC neural network were trained with data obtained by applying an input signal to the gravity matrix of the UR5 robot. For training the gravity compensation network, four trajectories were used, which can reflect specific tasks to be developed by the robot. Trajectories in the form of stacked ellipses, rose curve trajectory, sinusoid, and circular trajectory were implemented. The selected trajectories cover the robot's workspace and provide a good variety of movements, including cyclical movements, height changes, variable orientations, and complex curves. This is important to capture a wide range of gravitational effects. The sampling time was set at 0.01 s, the simulation time at 10 s and obtaining a total of 4000 training samples.

Table 4 summarizes the main training parameters established, both for training the inverse model of the robot and for training the neural gravity compensator.

Table 4. Training parameters of the neural networks.

Parameters	MLP	CfC
Epochs	5000	1000
Initial Learning Rate	–	1.0×10^{-3}
Mini-Batch Size	8	8
Activation Function	tansig, purelin	SiLU
Optimizer	Bayesian Regularized (trainbr)	Adam
Number of Neurons	3 Hidden Layers (19; 9; 6 neurons)	1 Hidden Layer (128 neurons)
Loss Function	MSE	MSE

The control scheme implemented for the Sliding Mode Control based on SMC-CfC-G is shown in Figure 6.

4.4. Cartesian Trajectories

To verify the performance of the proposed control strategy, the tracking of two trajectories is evaluated. Trajectory 1 corresponds to a three-petal flower, whose mathematical formulation is defined in (46) and is visualized in Figure 7a.

$$\begin{cases} r = 0.1\sin(3t) \\ x = 0.2 + r \cdot \sin(t) \\ y = 0.2 + r \cdot \cos(t) \\ z = 0.4 \end{cases} \tag{46}$$

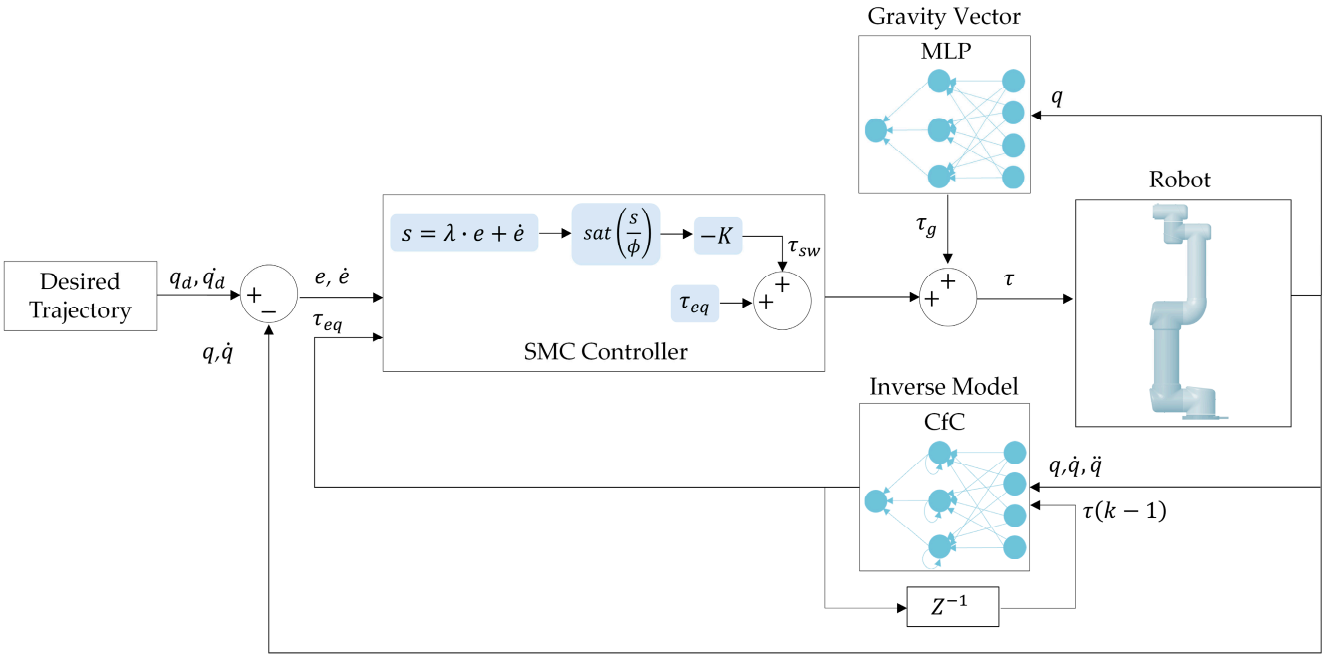
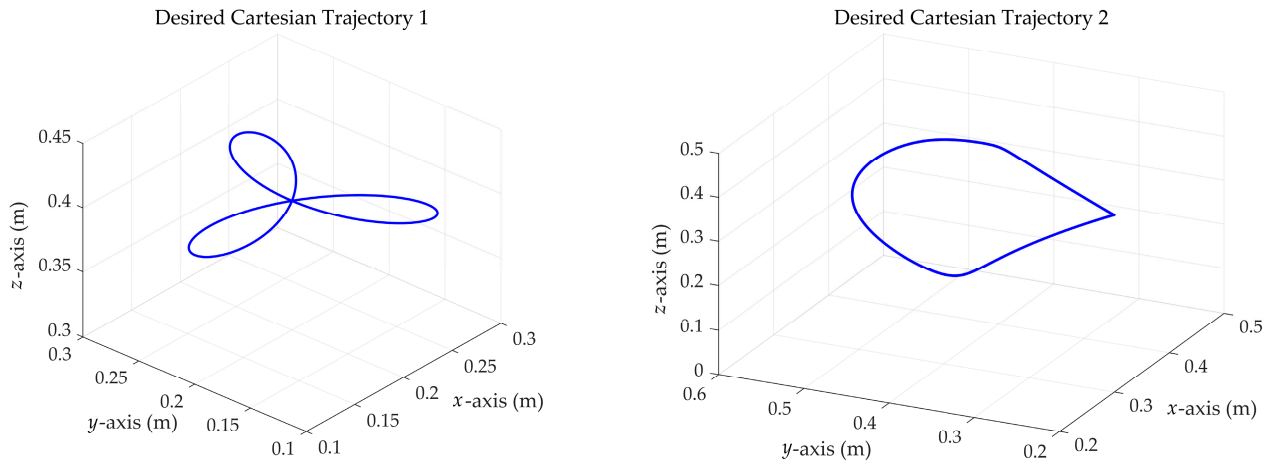


Figure 6. Proposed control scheme (SMC-CfC-G).



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

Figure 7. Trajectories in Cartesian space. (**a**) Three-petal flower; (**b**) Combination of lines and curves.

For trajectory 2, curves and straight lines are combined, as shown in Figure 7b. For its formation, the minjerkpolytraj function from MatLab® was used, which takes as input the number of samples, the waypoints, and the time to move from one waypoint to another.

These trajectories have been chosen for their ability to challenge the controller in multiple aspects. The three-petal flower, with its smooth curves and continuous changes in direction, allows for evaluating the precision and smoothness of the controller in complex movements. The transition between a curve and a straight line can involve abrupt changes in velocity and acceleration, which can cause vibrations or unwanted movements if the controller does not handle these changes adequately, providing an ideal scenario to test the robustness and adaptability of the controller.

5. Simulation Results

The training performed to develop the inverse model of the robot using MLP achieved an MSE of 4.95×10^{-6}, while the CfC achieved an MSE of 2.0×10^{-1}. The error values

reached during the training of the network used to compensate for gravity effects were 3.9×10^{-9} for the MLP and 6.2×10^{-2} for the CfC. Both neural networks were evaluated as an inverse model and as a gravity compensator, with the CfC performing best for modeling the robot's behavior and the MLP for gravity compensation. Table 5 summarizes the adjusted values for each joint in the implementation of the SMC-CfC-G controller.

Table 5. Adjusted values for SMC-CfC-G controller.

Joint	K	λ	ϕ
q_1	80	6	0.5
q_2	180	8	0.25
q_3	100	6	0.5
q_4	30	5	0.5
q_5	20	5	0.5
q_6	20	5	0.5

The selection of the SMC parameters for the UR5 robot controller was carried out using a systematic experimental approach. First, conservative values were set for the sliding surface coefficient (λ), the SMC gain (K), and the boundary layer thickness (ϕ), based on theoretical estimates. Then, an iterative tuning process was carried out, evaluating the robot's performance in terms of tracking accuracy, robustness to disturbances, and smoothness of movement. This process involved the execution of several tests with representative trajectories, analyzing the results of various performance indices, control effort, and the level of chattering.

The parameters were gradually refined for each joint, seeking a balance between response speed, precision, and stability. Finally, the selected parameters were validated with a set of trajectories not used during the tuning and under different load conditions. This approach allowed obtaining a set of parameters well-adapted to the specific dynamic characteristics of the UR5 and the specific purposes of the application, ensuring robust and efficient controller performance. The implementation of an adaptive (ϕ) is considered for future research, which favors real-time adjustments according to changing conditions during the robot's operation.

5.1. Performance Indexes

For a quantitative evaluation of the results of each controller in terms of response quality and error values in tracking the desired trajectory, performance indexes (PIs) are calculated. The PIs considered are: Integral Square Error (ISE), Integral of Time-Weighted Absolute Error (ITAE), and Residual Mean Square Error (RMSE), which are mathematically defined in Equations (47)–(49) [1,34]. The ISE penalizes larger error values to a greater extent; a small ISE is indicative of better overall reference tracking.

$$\text{ISE} = \int_0^T e^2\,(t)dt \;\; [\text{rad}] \tag{47}$$

$$\text{ITAE} = \int_0^T t|e(t)|dt \;\; [\text{rad}\cdot\text{s}] \tag{48}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \;\; [\text{rad}] \tag{49}$$

On the other hand, the ITAE penalizes late errors to a greater extent, and the RMSE measures the average magnitude of errors regardless of their sign. In all cases, the best controller performance is evidenced by ISE, ITAE, and RMSE values close to zero.

5.2. Trajectory Tracking

In addition to the controller proposed in the study, a classical SMC controller was designed with the same adjustment parameters as the proposed controller, as shown in Table 5, as well as a PID and a Neural controller (ANN), which allowed for comparison of the performance of the designed controller. Figure 8 shows the result for trajectory tracking by the designed controllers.

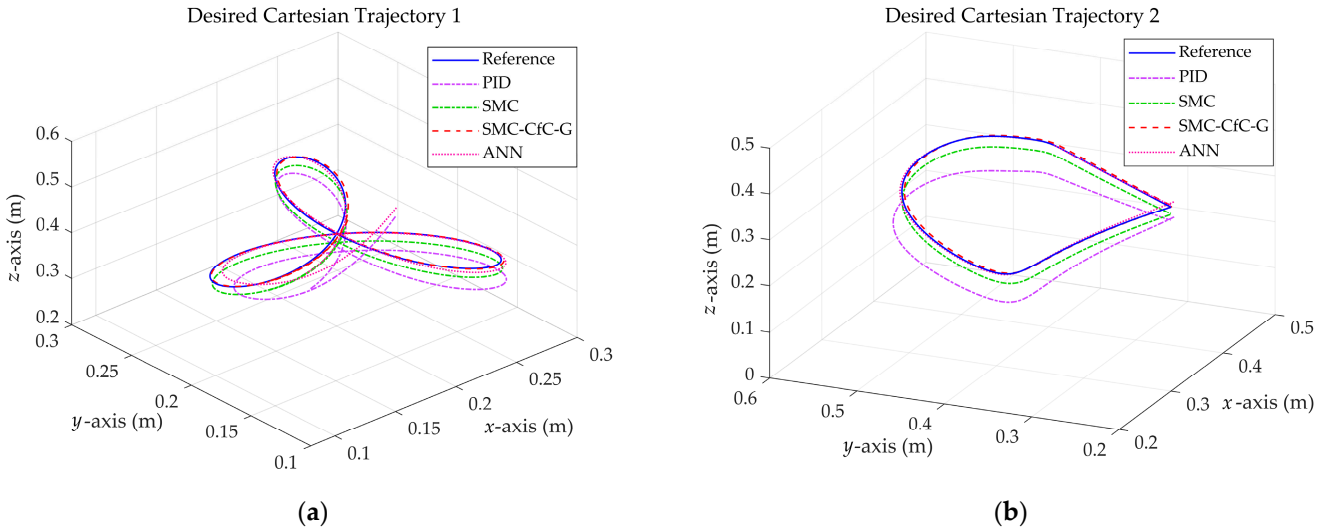


Figure 8. Trajectory tracking by the designed controllers. (**a**) Desired Cartesian trajectory 1 (three-petal flower); (**b**) Desired Cartesian trajectory 2 (combination of lines and curve).

As observed, all controllers performed the desired trajectory in its entirety and in an acceptable manner. However, a phase shift in the z-axis is evident, mainly from the PID and SMC controllers. According to this figure, the behavior of the neural controller is good, although less precise than that achieved by the SMC-CfC-G. For a closer look at the behavior of these controllers, Figures 9–11 show the results in Cartesian space (x, y, z axis) in trajectory tracking. In the first column is the positioning for trajectory 1, and in the second column is the positioning for trajectory 2.

The view of trajectory tracking in the previous axes does not present considerable differences, although it is distinguished that the PID controller presents the worst response, while the rest have a similar behavior, highlighting the performance of the SMC-CfC-G controller in all cases as the most precise. When analyzing the behavior on the z-axis, greater differences are observed, mainly influenced by gravity effects.

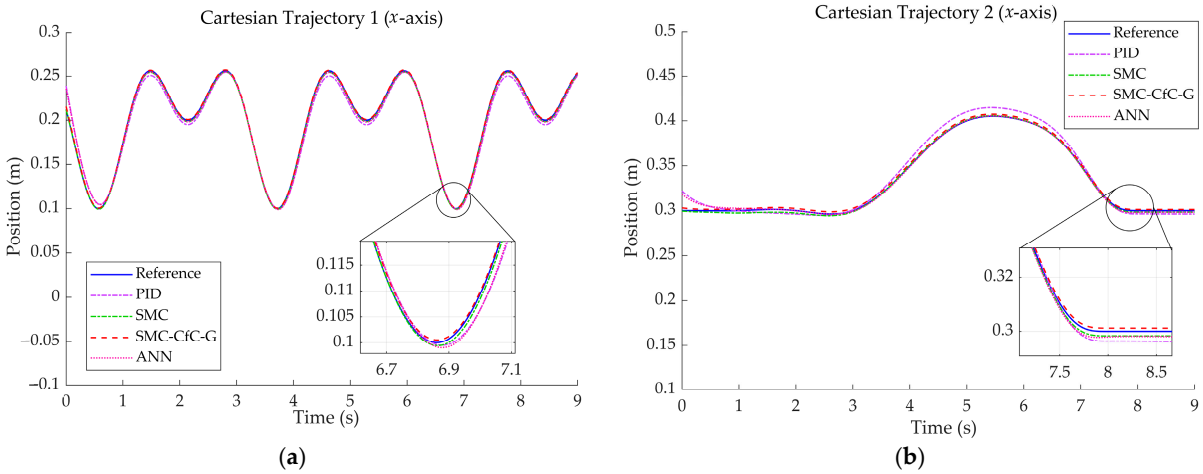
The PID controller is evidently strongly influenced by gravity effects, as is the SMC, albeit to a lesser extent. Visually, the neural controller is the second-best performer, with the SMC-CfC-G being the superior performer.

5.3. External Disturbance Rejection

An important characteristic of a control system is the ability to achieve good rejection of external disturbances. This ability is crucial for maintaining system stability and precision in dynamic and variable environments. To test this quality in the proposed controller, two types of disturbances were introduced that simulate adverse conditions that could occur in real applications.

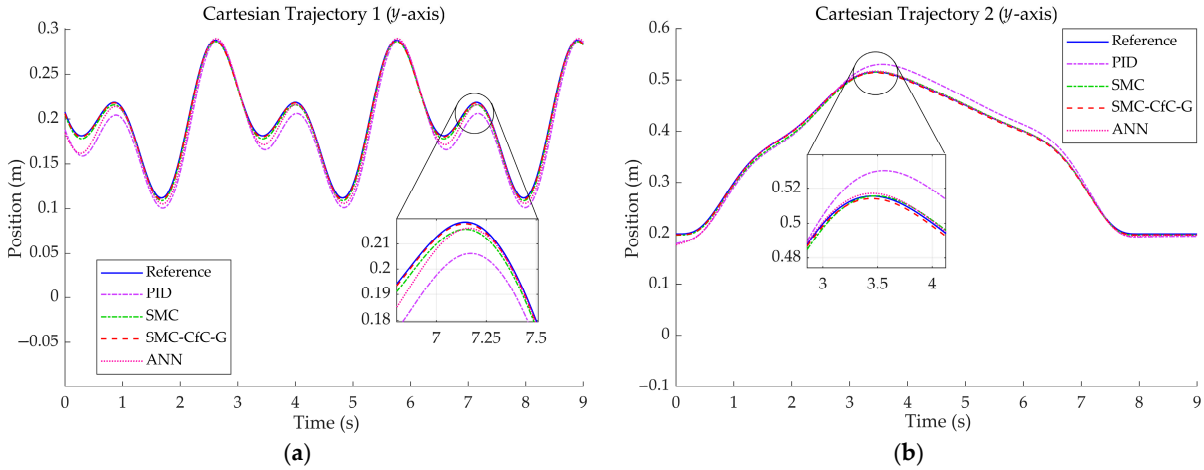
The first disturbance consisted of adding a 1 kg load to the end effector at 1.5 s of simulation, a load that was maintained for the rest of the trajectory. A 1 kg load applied to the end effector can be representative of a pick-and-place operation or fluctuations in tool weight, among other scenarios that represent a change in the robot's operating conditions, which is common in industrial applications. The second disturbance consisted of applying a torque of 5 Nm to the end effector from 1.2 s after starting its movement until 2.2 s in all

three axes. This disturbance can be representative of interactions with complex mechanisms such as gears, screwing operations, assembly, or environmental disturbances.



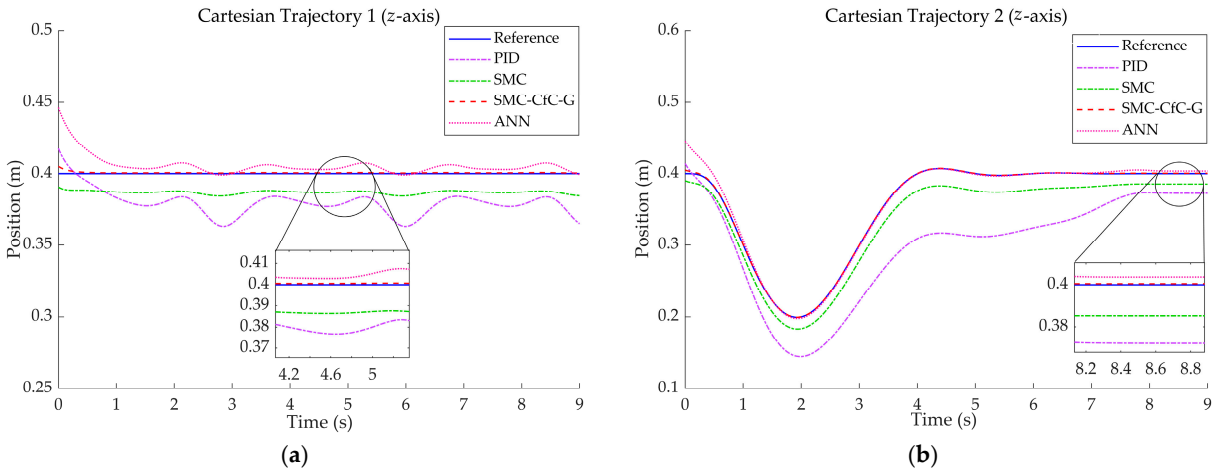
(**a**) (**b**)

Figure 9. Simulation results for the implemented control strategies. (**a**) *x*-axis tracking trajectory 1; (**b**) *x*-axis tracking trajectory 2.



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

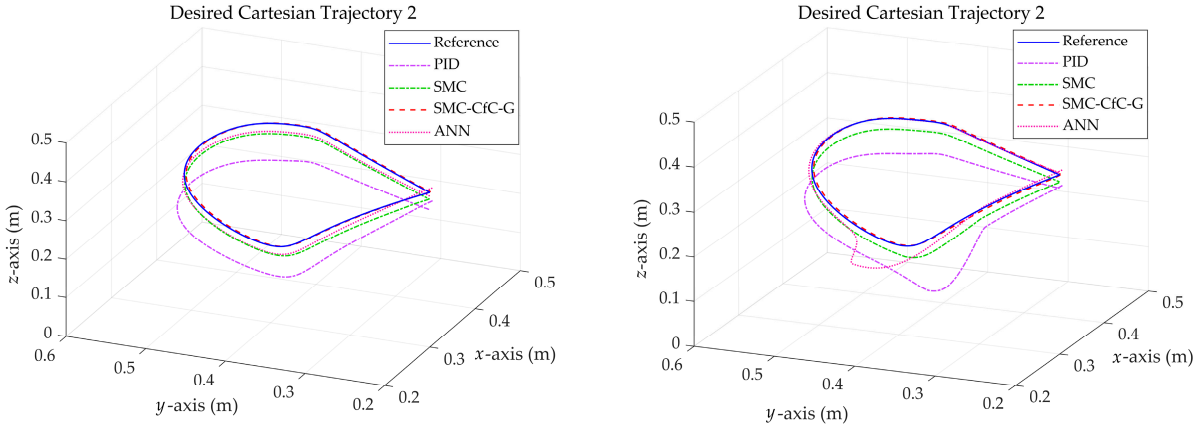
Figure 10. Simulation results for the implemented control strategies. (**a**) *y*-axis tracking trajectory 1; (**b**) *y*-axis tracking trajectory 2.



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

Figure 11. Simulation results for the implemented control strategies. (**a**) *z*-axis tracking trajectory 1; (**b**) *z*-axis tracking trajectory 2.

Figure 12 shows the results of tracking trajectory 2 by the designed controllers. The selection of this trajectory is motivated by the fact that it combines lines and curves, which can be more complicated for an articulated robot. Figure 12a shows the tracking under the 1 kg load, and Figure 12b reflects the performance achieved when applying the torque disturbance. As can be observed, both disturbances cause changes in trajectory tracking.



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

Figure 12. Tracking of Cartesian trajectory 2 in the presence of external disturbances. (**a**) Increase in end effector load; (**b**) Multidirectional torque.

When applying the mass, the main effect noted was a deviation in the z-axis; however, they maintained the ability to follow the trajectory. The controller most affected by this disturbance was the PID. The SMC and ANN maintained similar behavior, with a slight deviation in the z-axis. In the case of the SMC-CfC-G, the effects of this disturbance are not visually evident, with satisfactory control being observed. In contrast, the application of multidirectional torque resulted in more pronounced deviations from the desired trajectory, especially in the case of the PID controller, followed by the neural controller, which was particularly sensitive to this disturbance. Both the SMC and SMC-CfC-G maintained the ability to perform the desired trajectory, with the latter being the least affected by the disturbance, with no evident effects noted.

For a better analysis of the performance of the control strategies, the performance indices achieved by the implemented controllers for both trajectories, considering the different scenarios analyzed, are shown below. Table 6 contains all the performance indices for tracking trajectory 1 without considering disturbances.

Table 6. Performance indexes for trajectory 1 without disturbances.

PIs	Controller	$\mathbf{q_1}$	$\mathbf{q_2}$	$\mathbf{q_3}$	$\mathbf{q_4}$	$\mathbf{q_5}$	$\mathbf{q_6}$
ISE	PID	17.0843	0.0036	0.0088	1.94×10^{-4}	4.58×10^{-5}	4.13×10^{-6}
	ANN	0.0033	9.90×10^{-4}	0.0019	2.32×10^{-4}	4.58×10^{-5}	2.85×10^{-6}
	SMC	$\mathbf{2.09 \times 10^{-5}}$	8.57×10^{-5}	0.0026	9.12×10^{-5}	$\mathbf{1.09 \times 10^{-5}}$	$\mathbf{4.85 \times 10^{-7}}$
	SMC-CfC-G	8.54×10^{-5}	$\mathbf{6.26 \times 10^{-7}}$	$\mathbf{1.95 \times 10^{-5}}$	$\mathbf{4.68 \times 10^{-5}}$	2.92×10^{-5}	3.74×10^{-5}
ITAE	PID	13.5075	0.1570	0.3292	0.0403	0.0038	0.0055
	ANN	0.0530	0.0767	0.0284	0.0373	0.0038	0.0040
	SMC	**0.0125**	0.0248	0.1692	0.0326	$\mathbf{4.50 \times 10^{-4}}$	$\mathbf{5.79 \times 10^{-4}}$
	SMC-CfC-G	0.0277	**0.0020**	**0.0128**	**0.0178**	0.0109	0.0205
RMSE	PID	0.0315	0.0322	0.0502	0.0075	0.0036	0.0011
	ANN	0.0307	0.0168	0.0231	0.0082	0.0035	9.02×10^{-4}
	SMC	**0.0024**	0.0050	0.0274	0.0051	**0.0018**	$\mathbf{3.72 \times 10^{-4}}$
	SMC-CfC-G	0.049	$\mathbf{4.22 \times 10^{-4}}$	**0.0024**	**0.0037**	0.0029	0.0033

Best performance is denoted in bold.

In general, the SMC controllers obtained the best performance, with similar behavior based solely on the calculated quantitative parameters. The following tables reflect the performance indices achieved for trajectory 2. In this case, three scenarios are considered: the PIs calculated Without Disturbances (ND), with Load Disturbance (LD), and the scenario where Torque Disturbance (TD) is applied. Table 7 shows the ISE results for all controllers, with the best results achieved for each joint in the different scenarios highlighted.

Table 7. Performance index ISE for trajectory 2.

Controller	Scenario	q_1	q_2	q_3	q_4	q_5	q_6
PID	ND	0.0016	0.0484	0.0203	3.35×10^{-4}	4.07×10^{-5}	5.24×10^{-6}
	LD	0.0016	0.0722	0.0319	0.0070	4.06×10^{-5}	5.23×10^{-6}
	TD	0.0031	0.0516	0.0225	0.0394	0.0105	0.0473
ANN	ND	0.0015	0.0484	0.0203	3.26×10^{-4}	4.07×10^{-5}	5.08×10^{-6}
	LD	0.0015	0.0722	0.0319	0.0084	4.06×10^{-5}	8.59×10^{-6}
	TD	0.0032	0.0516	0.0225	0.0341	0.0105	0.0616
SMC	ND	$\mathbf{1.15 \times 10^{-6}}$	0.0014	0.0057	2.21×10^{-4}	$\mathbf{1.11 \times 10^{-5}}$	$\mathbf{1.80 \times 10^{-6}}$
	LD	$\mathbf{1.30 \times 10^{-6}}$	0.0018	0.0072	5.13×10^{-4}	$\mathbf{1.12 \times 10^{-5}}$	$\mathbf{1.81 \times 10^{-6}}$
	TD	$\mathbf{1.41 \times 10^{-5}}$	0.0015	0.0060	8.66×10^{-4}	1.24×10^{-4}	5.03×10^{-4}
SMC-CfC-G	ND	1.47×10^{-4}	$\mathbf{2.23 \times 10^{-6}}$	$\mathbf{3.94 \times 10^{-5}}$	$\mathbf{1.21 \times 10^{-4}}$	4.21×10^{-4}	5.57×10^{-4}
	LD	1.46×10^{-4}	$\mathbf{9.89 \times 10^{-6}}$	$\mathbf{9.18 \times 10^{-6}}$	$\mathbf{1.13 \times 10^{-4}}$	4.22×10^{-5}	8.69×10^{-5}
	TD	1.98×10^{-4}	$\mathbf{3.01 \times 10^{-6}}$	$\mathbf{3.43 \times 10^{-5}}$	$\mathbf{1.21 \times 10^{-4}}$	$\mathbf{4.03 \times 10^{-5}}$	$\mathbf{7.44 \times 10^{-5}}$

Best performance is denoted in bold.

As observed, although the rest of the controllers achieve small values, the SMC and SMC-CfC-G controllers perform best in terms of ISE in all cases. The SMC provides better results for q_1, q_5, and q_6, while the SMC-CfC-G offers better performance in q_2, q_3, and q_4. However, in the presence of disturbance when applying torque, the SMC-CfC-G controller shows better performance, demonstrating greater robustness and adaptability. Table 8 contains the ITAE.

When analyzing the ITAE, the result obtained is similar to the previous one, with SMC controllers being superior to the rest. Table 9 contains the calculated RMSE values. Despite similar values between the SMC controllers, improvements to the proposed controller over the classic SMC controller are verified here. The calculated metrics reflect the effects of disturbances on the different controllers, with the adverse effects of applying torque disturbance being notable on most occasions. In the case of increased load on the robot's end effector, no considerable increase in performance indices is observed compared with the same parameter without disturbance.

Table 8. Performance index ITAE for trajectory 2.

Controller	Scenario	q_1	q_2	q_3	q_4	q_5	q_6
PID	ND	0.0511	2.3132	1.7039	0.1880	0.0032	0.0016
	LD	0.0570	2.9050	2.1495	0.9814	0.0016	7.15×10^{-4}
	TD	0.1330	2.3448	1.7410	0.5715	0.2093	0.4382
ANN	ND	0.0488	2.3132	1.7039	0.1801	0.0032	0.0133
	LD	0.0555	2.9050	2.1495	1.0801	0.0016	0.0265
	TD	0.1455	2.3448	1.7410	0.5245	0.2093	0.5104
SMC	ND	**0.0081**	0.4030	0.8688	0.1716	$\mathbf{4.94 \times 10^{-4}}$	$\mathbf{3.34 \times 10^{-4}}$
	LD	**0.0089**	0.4539	0.9902	0.2645	$\mathbf{9.65 \times 10^{-4}}$	$\mathbf{3.51 \times 10^{-4}}$
	TD	**0.0142**	0.4064	0.8779	0.2095	**0.0208**	**0.0428**
SMC-CfC-G	ND	0.1347	**0.0153**	**0.0680**	**0.1186**	0.2274	0.2692
	LD	0.1346	**0.0349**	**0.0188**	**0.1137**	0.0549	0.1076
	TD	0.1382	**0.0156**	**0.0665**	**0.0982**	0.0544	0.1046

Best performance is denoted in bold.

Table 9. Performance index RMSE for trajectory 2.

Controller	Scenario	q_1	q_2	q_3	q_4	q_5	q_6
PID	ND	0.0142	0.0778	0.0504	0.0065	0.0023	8.09×10^{-4}
	LD	0.0142	0.0950	0.0631	0.0297	0.0023	8.08×10^{-4}
	FD	0.0197	0.0803	0.0530	0.0702	0.0362	0.0769
ANN	ND	0.0137	0.0059	0.0145	0.0064	0.0022	7.97×10^{-4}
	LD	0.0137	0.0180	0.0180	0.0325	0.0023	0.0010
	FD	0.0198	0.0395	0.0155	0.0652	0.0580	0.0878
SMC	ND	$\mathbf{3.79 \times 10^{-4}}$	0.0133	0.0266	0.0053	**0.0012**	$\mathbf{4.75 \times 10^{-4}}$
	LD	$\mathbf{4.04 \times 10^{-4}}$	0.0148	0.0301	0.0080	**0.0012**	$\mathbf{4.75 \times 10^{-4}}$
	FD	**0.0013**	0.0135	0.0273	0.0104	0.0039	0.0079
SMC-CfC-G	ND	0.0043	$\mathbf{5.28 \times 10^{-4}}$	**0.0022**	**0.0039**	0.0073	0.0084
	LD	0.0043	**0.0011**	**0.0011**	**0.0038**	0.0023	0.0033
	FD	0.0050	$\mathbf{6.13 \times 10^{-4}}$	**0.0021**	**0.0039**	**0.0022**	**0.0030**

Best performance is denoted in bold.

6. Discussion

The main motivation of this study is to apply a robust control strategy based on data under the gray box principle with high precision in trajectory tracking. The proposed SMC-CfC-G controller presents several advantages that justify its good performance. Firstly, the SMC control law focuses on reducing error over time and minimizing it quickly. In the specific case of the implemented controller, a maximum error of approximately 2.2 mm is achieved on the *x*-axis, approximately 1.7 mm on the *y*-axis, and approximately 0.8 mm on the *z*-axis.

Another important strength of the proposed controller is gravity compensation. When verifying the errors, especially on the *z*-axis, the effectiveness of this methodology and the neural network dedicated to this function is observed. The MLP outperformed the CfC in developing the gravity compensator, which may be largely influenced by a small number of training samples. However, both neural networks showed better performance than that offered by applying the gravity vector obtained through robot modeling, so this methodology is considered efficient in controller design.

It is worth noting that we worked with a fairly realistic configuration of the UR5 robot through its implementation in Simscape, where the movement ranges established by the manufacturer were considered, as well as the maximum speed and torque values established for each joint, which provides greater reliability of the results obtained in the study. The performance of the inverse model developed using the CfC neural network was decisive in the final result, as it was able to adequately model the system dynamics. The CfC neural network managed to determine the robot parameters with a chirp signal; this process can be improved with the implementation of other signals and a larger number of training data points.

To add more realism to the analysis, the results of trajectory tracking were evaluated qualitatively and quantitatively in different situations. Curved trajectories were implemented, which are more natural for the UR5 based on its morphology, as well as lines, which, despite being more challenging for the robot, were successfully performed by the proposed controller. The study was also complemented by the implementation of common external disturbances in industry with different degrees of complexity. A 1 kg load applied to the end effector allowed evaluation of the adaptability and robustness of the control algorithm. The application of a multidirectional torque of 5 Nm corresponds to a more complex and challenging scenario, providing a more rigorous test for the control system. This disturbance allowed testing the recovery capacity and maintenance of precision of the control system in variable load situations.

From these disturbances, high sensitivity of the PID to gravitational-type disturbances and low adaptability were evidenced; the neural controller and SMC were also affected

to a lesser extent, which was not the case for the SMC-CfC-G controller. The second disturbance caused considerable deviations in trajectory tracking by the PID and ANN controllers, impairing their ability to compensate for the effect of the disturbance and hindering immediate correction of trajectory tracking. The SMC controllers were able to correctly track the trajectory, although the classic SMC maintains a larger error on the z-axis. In contrast, the SMC-CfC-G error underwent minimal variations, maintaining high precision, which can be observed qualitatively and quantitatively.

Regarding the calculated performance indices, they allow verifying the better behavior of SMC control in general. However, it can be stated to some extent that RMSE and ISE were more accurate in differentiating the best-performing controller. Both indices focus on minimizing overall error without introducing time weight, which may be more suitable for systems where balanced and robust performance is sought throughout the trajectory, offering a clear and direct view of the system's overall precision. Based on the results obtained, a low average deviation between the desired values and the actual values obtained by the controller is evidenced, and consequently, high precision and good reference tracking.

7. Conclusions

This study presented a robust data-based controller approach for precise trajectory tracking. Neural networks were applied both for identifying the inverse model of the UR5 robot and for gravity compensation, resulting in precise, robust control and good rejection of external disturbances. As part of this research, the behavior of the recently proposed CfC neural network was studied, which has been applied in a small number of situations, thus demonstrating its efficiency and applicability in industrial robot control.

Regarding the limitations of this study, it can be highlighted that, given the complexity of adjusting parameters of CfC neural networks, it is considered that further experiments should be conducted with various parameter configurations, such as initial learning rate, activation function, and optimizers, among others, to achieve better performance and adjustment. It is also significant to evaluate the performance of other variants of the CfC architecture, such as the so-called CfC-mmRNN, where the CfC defines the memory state of an RNN, for example, a Long Short-Term Memory, which allows mitigating the vanishing gradient problem. On the other hand, although the inverse model of the robot was made using the CfC neural network, which allows efficient learning of temporal relationships, the use of chirp signals only in identification is recognized as a limitation of the study, which limited the modeling of gravitational effects to some extent. For this reason, it was considered to perform a neural gravity compensator to counteract.

In future research, the SMC-CfC-G controller will be applied to high-precision medical tasks, particularly in the field of robotic surgery. The proposed controller could be integrated as a low-level component within a hierarchical control system for surgical robots, taking advantage of its high precision and robustness to disturbances. This application could significantly improve accuracy in delicate surgical procedures, such as microsurgeries or minimally invasive interventions. In addition to improvements in precision, the controller's ability to adapt to different loads and efficiently compensate for gravity effects could be particularly useful in manipulating various surgical instruments. This application would not only demonstrate the practical utility of our approach in a highly specialized industrial context but could also contribute significantly to the advancement of robotic surgery, potentially improving outcomes for patients.

Author Contributions: Conceptualization, C.U. and Y.G.-G.; methodology, C.U. and Y.G.-G.; software, C.U. and Y.G.-G.; validation, C.U. and Y.G.-G.; formal analysis, C.U. and Y.G.-G.; investigation, C.U. and Y.G.-G.; resources, C.U. and Y.G.-G.; data curation, C.U. and Y.G.-G.; writing—original draft preparation, C.U. and Y.G.-G.; writing—review and editing, C.U. and Y.G.-G.; visualization, C.U. and Y.G.-G.; supervision, C.U. and J.K.; project administration, C.U. and J.K.; funding acquisition, C.U. and J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Faculty of Engineering of the University of Santiago of Chile and Agencia Nacional de Investigación y Desarrollo de Chile (ANID-Subdirección de Capital Humano/Doctorado Nacional/2022-21220266).

Data Availability Statement: All data are contained within this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Urrea, C.; Garcia-Garcia, Y. Design and Performance Analysis of Level Control Strategies in a Nonlinear Spherical Tank. *Processes* **2023**, *11*, 720. [CrossRef]
2. Qureshi, M.S.; Swarnkar, P.; Gupta, S. A supervisory on-line tuned fuzzy logic based sliding mode control for robotics: An application to surgical robots. *Robot. Auton. Syst.* **2018**, *109*, 68–85. [CrossRef]
3. Sachan, S.; Swarnkar, P. Robust Motion Planning in Robot-Assisted Surgery for Nonlinear Incision Trajectory. *Electronics* **2023**, *12*, 762. [CrossRef]
4. Shen, X.; Liu, J.; Lin, H.; Yin, Y.; Alcaide, A.M.; Leon, J.I. Cascade control of grid-connected NPC converters via sliding mode technique. *IEEE Trans. Energy Convers.* **2023**, *38*, 1491–1500. [CrossRef]
5. Shen, X.; Wu, C.; Liu, Z.; Wang, Y.; Leon, J.I.; Liu, J.; Franquelo, L.G. Adaptive-gain second-order sliding mode control of NPC converters via super-twisting technique. *IEEE Trans. Power Electron.* **2023**, *38*, 15406–15418. [CrossRef]
6. Shen, X.; Liu, J.; Liu, Z.; Gao, Y.; Leon, J.I.; Vazquez, S.; Wu, L.; Franquelo, L.G. Sliding Mode Control of Neutral-Point-Clamped Power Converters with Gain Adaptation. *IEEE Trans. Power Electron.* **2024**, *39*, 9189–9201. [CrossRef]
7. Shen, H.; Song, T.; Fang, L.; Wang, H.; Zhang, Y. Adaptive super twisting observer-based prescribed time integral sliding mode tracking control of uncertain robotic manipulators. *Int. J. Adapt. Control Signal Process.* **2024**, *38*, 2588–2616. [CrossRef]
8. Abbasi, S.J.; Lee, S. Enhanced Trajectory Tracking via Disturbance-Observer-Based Modified Sliding Mode Control. *Appl. Sci.* **2023**, *13*, 8027. [CrossRef]
9. Abadi, A.; Ayeb, A.; Labbadi, M.; Fofi, D.; Bakir, T.; Mekki, H. Robust Tracking Control of Wheeled Mobile Robot Based on Differential Flatness and Sliding Active Disturbance Rejection Control: Simulations and Experiments. *Sensors* **2024**, *24*, 2849. [CrossRef]
10. Jiang, S.; Huang, F.; Liu, W.; Huang, Y.; Chen, Y. A Double Closed-Loop Digital Hydraulic Cylinder Position System Based on Global Fast Terminal Sliding Mode Active Disturbance Rejection Control. *IEEE Access* **2024**, *12*, 80138–80152. [CrossRef]
11. Chang, Y.-H.; Yang, C.-Y.; Lin, H.-W. Robust Adaptive-Sliding-Mode Control for Teleoperation Systems with Time-Varying Delays and Uncertainties. *Robotics* **2024**, *13*, 89. [CrossRef]
12. Tian, D.; Wang, M.; Wang, F.; Xu, R. Adaptive sliding-mode-assisted disturbance observer-based decoupling control for inertially stabilized platforms with a spherical mechanism. *IET Control. Theory Appl.* **2022**, *16*, 1194–1207. [CrossRef]
13. Kern, J.; Urrea, C.; Verdejo, H.; Agramonte, R.; Becker, C. Trajectory Tracking and Disturbance Rejection Performance Analysis of Classical and Advanced Controllers for a SCORBOT Robot. *Robotics* **2024**, *13*, 48. [CrossRef]
14. Papageorgiou, D.; Sigurðardóttir, G.Þ.; Falotico, E.; Tolu, S. Sliding-mode control of a soft robot based on data-driven sparse identification. *Control. Eng. Pract.* **2024**, *144*, 105836. [CrossRef]
15. Truong, T.N.; Vo, A.T.; Kang, H.-J. A model-free terminal sliding mode control for robots: Achieving fixed-time prescribed performance and convergence. *ISA Trans.* **2024**, *144*, 330–341. [CrossRef]
16. Sang, H.; Yang, C.; Liu, F.; Yun, J.; Jin, G. A fuzzy neural network sliding mode controller for vibration suppression in robotically assisted minimally invasive surgery. *Int. J. Med. Robotics Comput. Assist. Surg.* **2016**, *12*, 670–679. [CrossRef]
17. Yuan, T.; Zhang, C.; Yi, F.; Lv, P.; Zhang, M.; Li, S. RBFNN-Based Adaptive Integral Sliding Mode Feedback and Feedforward Control for a Lower Limb Exoskeleton Robot. *Electronics* **2024**, *13*, 1043. [CrossRef]
18. Wu, H.; Zhang, X.; Song, L.; Zhang, Y.; Wang, C.; Zhao, X.; Gu, L. Parallel Network-Based Sliding Mode Tracking Control for Robotic Manipulators with Uncertain Dynamics. *Actuators* **2023**, *12*, 187. [CrossRef]
19. Li, L.J.; Chang, X.; Chao, F.; Lin, C.M.; Huỳnh, T.T.; Yang, L.; Shang, C.; Shen, Q. Self-Organizing Type-2 Fuzzy Double Loop Recurrent Neural Network for Uncertain Nonlinear System Control. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 1–15. [CrossRef]
20. Sachan, S.; Swarnkar, P. Intelligent fractional-order sliding mode optimised control of surgical manipulator for healthcare system. *Electr. Eng.* **2024**, *106*, 2131–2142. [CrossRef]
21. Khan, H.; Khan, S.A.; Lee, M.C.; Ghafoor, U.; Gillani, F.; Shah, U.H. DDPG-Based Adaptive Sliding Mode Control with Extended State Observer for Multibody Robot Systems. *Robotics* **2023**, *12*, 161. [CrossRef]
22. Li, T.; Zhang, G.; Zhang, T.; Pan, J. Adaptive Neural Network Tracking Control of Robotic Manipulators Based on Disturbance Observer. *Processes* **2024**, *12*, 499. [CrossRef]
23. Mystkowski, A.; Wolniakowski, A.; Kadri, N.; Sewiolo, M.; Scalera, L. Neural Network Learning Algorithms for High-Precision Position Control and Drift Attenuation in Robotic Manipulators. *Appl. Sci.* **2023**, *13*, 10854. [CrossRef]
24. Hasani, R.; Lechner, M.; Amini, A.; Liebenwein, L.; Ray, A.; Tschaikowski, M.; Teschl, G.; Rus, D. Closed-form continuous-time neural networks. *Nat. Mach. Intell.* **2022**, *4*, 992–1003. [CrossRef]

25. Hasani, R.; Lechner, M.; Amini, A.; Rus, D.; Grosu, R. Liquid time-constant networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual. 2–9 February 2021; Volume 35, pp. 7657–7666.
26. Chen, R.T.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K. Neural ordinary differential equations. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 1–19. [CrossRef]
27. Urrea, C.; Garcia-Garcia, Y.; Kern, J. Improving Surgical Scene Semantic Segmentation through a Deep Learning Architecture with Attention to Class Imbalance. *Biomedicines* **2024**, *12*, 1309. [CrossRef]
28. Universal Robots A/S. UR5 Especificaciones Técnicas No. Artículo 110105. Available online: https://www.universal-robots.com/media/50591/ur5_es.pdf (accessed on 16 June 2024).
29. Jian, B.-L.; Tsai, C.-S.; Kuo, Y.-C.; Guo, Y.-S. An image vision and automatic calibration system for universal robots. *J. Low Freq. Noise Vib. Act. Control.* **2021**, *40*, 347–366. [CrossRef]
30. Kufieta, K. Force Estimation in Robotic Manipulators: Modeling, Simulation and Experiment. Master's Thesis, NTNU Norwegian University of Science and Technology, Trondheim, Norway, 2014. Available online: http://folk.ntnu.no/tomgra/Diplomer/Kufieta.pdf (accessed on 16 June 2024).
31. Liu, J. *Sliding Mode Control Using MATLAB*; Academic Press: Cambridge, MA, USA, 2017.
32. Wang, S.; Shao, X.; Yang, L.; Liu, N. Deep learning aided dynamic parameter identification of 6-DOF robot manipulators. *IEEE Access* **2020**, *8*, 138102–138116. [CrossRef]
33. Bembli, S.; Haddad, N.K.; Belghith, S. A robust Model free terminal sliding mode with gravity Compensation control of a 2 DoF exoskeleton-upper limb system. *J. Control Autom. Electr. Syst.* **2021**, *32*, 632–641. [CrossRef]
34. Silaa, M.Y.; Bencherif, A.; Barambones, O. Indirect Adaptive Control Using Neural Network and Discrete Extended Kalman Filter for Wheeled Mobile Robot. *Actuators* **2024**, *13*, 51. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.