Integral Sliding Modes Generation via DRL-Assisted Lyapunov-Based Control for Robot Manipulators

Nikolas Sacchi, Gian Paolo Incremona and Antonella Ferrara

Abstract— **This paper proposes an enhanced version of the integral sliding mode (ISM) control, where a deep neural network (DNN) is first trained as a deep reinforcement learning (DRL) agent. Then, such a DNN is fine-tuned relying on a Lyapunov-based weight adaptation law, with the aim of compensating the lack of knowledge of the full dynamics in the case of robot manipulators. Specifically, a DRL agent is trained off-line on a reward depending on the sliding variable to estimate the unknown drift term of the robot dynamics. Such an estimate is then exploited to initialize and perform the *fine tuning* of the online adaptation mechanism based on the DNN. The proposal is theoretically analysed and assessed in simulation relying on the planar configuration of a Franka Emika Panda robot manipulator.**

Index Terms— **Integral sliding mode control, reinforcement learning, neural networks, robot manipulators.**

I. Introduction

The need to cope with unavoidable modelling mismatches and disturbances affecting plants made sliding mode control (SMC) very popular because of its capability of robustifying the controlled system in front of matched uncertainties whenever the system states lie on a suitable *sliding manifold* [1]. In particular, the key ingredient of a SMC law is its discontinuous nature, which allows the so-called *sliding variable* to converge towards the sliding manifold in a finite time. The convergence time is in turn determined by the control gain which has to be selected so as to dominate the worst realization of the uncertainty terms acting on the system.

However, the classical SMC presents two main drawbacks. On the one hand, the designed control gain strongly influences the presence of the so-called chattering phenomenon [2]. On the other hand, during the *reaching phase* towards the manifold the controlled system is still sensitive to disturbances. In the literature, different solutions to these issues have been proposed. Higher order sliding mode controllers [3], [4], full order sliding mode strategies [5], and adaptive approaches [6]–[8] contributed to the field as valid methodologies for chattering alleviation. Instead, the enhancement of robustness features of SMC during the reaching phase has been first solved in the seminal work [9], where the so-called integral sliding mode (ISM) control has been presented.

The basic idea of an ISM control is to remove the reaching phase, thus enabling the sliding mode since the initial time instant, with beneficial effects in terms of robustness despite matched uncertainties. An extension in the case of both matched and unmatched disturbances is instead proposed in [10] and [11]. Among many applications, robotic systems represent a challenging case study for which ISM control has been successfully used, see e.g., [12], [13]. Such systems are often characterized by a partial knowledge of their dynamics. Specifically, the knowledge of the so-called *control effectiveness matrix* is often a common assumption, while this is not valid for the *drift* component, which depends on frictions and Coriolis effects typically unknown to the controller designer. This aspect can motivate the use of a learning approach to estimate the missing dynamics information.

Indeed, by virtue of the growth of the computational power, the so-called learning paradigm has become very popular. One type of learning approach, used with very satisfactory results in a wide number of fields, is reinforcement learning (RL) [14], as for instance in robotics (see, e.g., [15], [16]) or healthcare (see, e.g., [17], [18]). Furthermore, in the case of very complex problems, the use of universal function approximators like deep neural networks (DNNs) [19] is required, giving rise to deep RL (DRL) methods. As mentioned, in this context, neural networks (NNs) are the core of the learning approach. NNs have been indeed successfully applied in many control design applications giving rise to a variety of learning-based data-driven control strategies, with stability and robustness guarantees, see e.g., [20], [21] for an overview. Among many works, for instance, [22] introduced weight adaptation laws for NNs which are derived from Lyapunov stability analysis. Such techniques have been used to directly approximate the optimal control law (see, e.g., [23]) or to estimate online part of the system model. More recently, in [24] a two-layers NN with the aforementioned weight adaptation laws to estimate the plant model has been proposed in the framework of ISM control, giving rise to a novel NN-ISM control approach.

Inspired by [24] and motivated by robotic applications, in this work we propose an extended version of the NN-ISM control which relies on the use of a DNN, in order to estimate the unknown drift term of the robot dynamics. More precisely, differently from [24], the DNN is initially treated as a DRL agent, trained off-line according to a reward function which depends on the sliding variable. The latter is

This is the final version of the accepted paper submitted for inclusion in the Proceedings of the European Control Conference 2023, Bucharest, Romania, June 2023, doi:10.23919/ECC57647.2023.10178264. The work has been partially supported by the Italian Ministry for Research in the framework of the 2017 Program for Research Projects of National Interest (PRIN), Grant no. 2017YKXYXJ.

Nikolas Sacchi and Antonella Ferrara are with the Dipartimento di Ingegneria Industriale e dell'Informazione, University of Pavia, 27100, Pavia, Italy (e-mail: *nikolas.sacchi01@universitadipavia.it*, *antonella.ferrara@unipv.it*). Gian Paolo Incremona is with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy (e-mail: *gianpaolo.incremona@polimi.it*).

indeed a valuable index to estimate the unknown drift term of the robot dynamics, which is essential to design the ISM control law. Then, the DNN weights achieved as outcome of the DRL training are adopted to initialize the DNN used online. Furthermore, an adaptation law based on Lyapunov stability analysis is introduced to perform the so-called *fine tuning* for the weight of the last layer of the DNN.

Making reference to [24], where the weights of the NN are randomly initialized, thus possibly causing temporary large estimation error, here the joint use of DRL and Lyapunov-based adaptation leads to different improvements and advantages. On the one hand, pre-training the DNN using DRL allows to initialize the online control phase with smaller estimation error, resulting in a faster recovering of the sliding mode generation whenever this is lost. On the other hand, the introduction of fine tuning based on Lyapunov stability analysis allows to compensate the possible lack of generalization of the DNN caused by an insufficient quantity of experience during the DRL training phase. The convergence properties enabled by the proposed DRL Lyapunov-based ISM control are proved in the paper and assessed in simulation on a realistic model of a Franka Emika Panda robot manipulator.

The paper is structured as follows. In Section II the considered robot model is introduced and some preliminaries on ISM control and the DRL method are recalled. The DRL based estimation of the unknown dynamics of the robot is presented in Section III, while the adopted ISM control is discussed in Section IV. The proposal is theoretically analysed in Section V, and simulation results relying on a realistic robot simulator are illustrated in Section VI. Finally, some conclusions are drawn in Section VII.

Notation: Let $x \in \mathbb{R}^m$ be a column vector, then $x^\top \in \mathbb{R}^{1\times m}$ represents its transpose. Given a real matrix $A \in \mathbb{R}^{m\times m}$, then $\mathrm{tr}(A)$ is its trace, while $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ denote the largest and the smallest eigenvalue of matrix A, respectively. Given two real matrices $A, B \in \mathbb{R}^{m\times m}$, then $\mathrm{tr}(A+B) = \mathrm{tr}(A) + \mathrm{tr}(B)$, while given $A \in \mathbb{R}^{n\times m}$, $B \in \mathbb{R}^{m\times n}$, then $\mathrm{tr}(AB) = \mathrm{tr}(BA)$. Given two real column vectors $a, b \in \mathbb{R}^m$, the trace of the outer product is equivalent to the inner product, i.e., $\mathrm{tr}(ba^\top) = a^\top b$. Given two functions $f(x)$ and $g(x)$, then the function composition $h(x) = g(x) \circ f(x)$ is such that $h(x) = g(f(x))$.

II. Preliminaries and Problem Statement

In this section the dynamical model of the considered robot manipulator is presented, and the design of the ISM control, originally introduced in [9], is recalled.

A. Robot modelling

Consider an open-chain robot manipulator with n revolute joints, whose angular positions are collected in the vector $q \in \mathbb{R}^n$. The dynamics of such a manipulator is given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) = \tau + \tau_{\mathrm{d}}, \qquad (1)$$

where $M(q) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the inertia matrix, $C(q,\dot{q}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the Coriolis/centripetal matrix, $F(\dot{q}) : \mathbb{R}^n \to \mathbb{R}^n$ is the vector of friction terms, $G(q) : \mathbb{R}^n \to \mathbb{R}^n$ is the gravity vector, $\tau_{\mathrm{d}} \in \mathbb{R}^n$ is the disturbances vector, while $\tau \in \mathbb{R}^n$ is the input torque [25, Ch. 2]. Note that the dependence on time t of all the variables is omitted when obvious, for the sake of simplicity.

Having in mind a motion control problem, let $\tilde{q} := q - q^\star$ be the position error, with $q^\star$ being the desired reference position. As result of a 2-fold derivative, one obtains the error model corresponding to (1), i.e.,

$$\ddot{\tilde{q}} = -M^{-1}(q)[C(q,\dot{q})\dot{q}+F(\dot{q})+G(q)]+M^{-1}(q)(\tau+\tau_{\mathrm{d}})-\ddot{q}^\star. \qquad (2)$$

Now, let us consider the state space formulation of model (2), posing $x := [\tilde{q}^\top\ \dot{\tilde{q}}^\top]^\top$, with $x \in \Omega \subset \mathbb{R}^{2n}$, i.e.,

$$\dot{x} = f(x) + B(x)u + h(x,t) + a, \qquad (3)$$

with $x_0 \in \Omega$ being the initial condition, and $f(x) : \Omega \to \mathbb{R}^{2n}$ representing the smooth drift dynamics, defined as

$$f(x) := \begin{bmatrix} \dot{\tilde{q}} \\ -M(q)^{-1}\left[C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q)\right] \end{bmatrix}.$$

Furthermore, $B(x) : \Omega \to \mathbb{R}^{2n\times n}$ is the control effectiveness matrix, defined as

$$B(x) := \begin{bmatrix} 0_{n\times n} \\ M(q)^{-1} \end{bmatrix},$$

for which the following assumption holds.

$\boldsymbol{\mathcal{A}_1}$: There exist known constants $\underline{b}, \bar{b} \in \mathbb{R}_{>0}$ such that

$$\inf_{x\in\Omega} \|B(x)\| \geq \underline{b}, \qquad (4a)$$

$$\sup_{x\in\Omega} \|B(x)\| \leq \bar{b}. \qquad (4b)$$

Finally, let $u := \tau \in \mathbb{R}^n$ be the input, and let $h(x,t) : \Omega \times \mathbb{R} \to \mathbb{R}^{2n}$ and $a \in \mathbb{R}^{2n}$ be the system perturbation and the reference acceleration vector respectively, given by

$$h(x,t) := B(x)\tau_{\mathrm{d}}, \quad a := \begin{bmatrix} 0_{n\times 1} \\ -\ddot{q}^\star \end{bmatrix}$$

such that the following assumption needs to be introduced.

$\boldsymbol{\mathcal{A}_2}$: There exist known constants $\bar{\tau} \in \mathbb{R}_{>0}$ and $\bar{h} = \bar{b}\bar{\tau} \in \mathbb{R}_{>0}$ such that

$$\sup_{t\in\mathbb{R}_{\geq 0}} \|\tau_{\mathrm{d}}(t)\| \leq \bar{\tau}, \qquad (5a)$$

$$\|h(x,t)\| \leq \bar{h},\ \forall x \in \Omega,\ \forall\, t \in \mathbb{R}_{\geq 0}. \qquad (5b)$$

Note that the previous assumption is instrumental to design the ISM control. Characterizing this bound could require a careful analysis of the considered system, relying on, e.g., data analysis or physical insights.

B. ISM control

Making reference to [9] and having in mind the robot application, the design of an ISM control for system (3) is hereafter recalled. The classical formulation of the ISM control consists of two components, i.e.,

$$u = u_0 + u_1, \tag{6}$$

with $u_0 \in \mathbb{R}^n$ being a stabilizing control law which makes the origin be an asymptotically stable equilibrium point for the nominal dynamics given by (3) when $h = 0$ (i.e., $\tau_{\mathrm{d}} = 0$), and $u_1 \in \mathbb{R}^n$ being the discontinuous control law designed so as to reject the uncertainty terms. Since (3) is a multi-input-multi-output coupled nonlinear system, the discontinuous law is selected according to the unit vector approach [1] as

$$u_1 = -\rho \frac{s(x)}{\|s(x)\|}, \tag{7}$$

where $\rho \in \mathbb{R}_{>0}$ is a constant gain selected to dominate the worst realization of the uncertainty terms, while $s(x) : \Omega \to \mathbb{R}^n$ is the so-called *integral sliding variable* given by

$$s(x) = s_0(x) + z(x), \quad s(x_0) = 0. \tag{8}$$

The term $z(x) : \Omega \to \mathbb{R}^n$ in (8) is the *transient function*, defined so that

$$\dot{z} = -\frac{\partial s_0}{\partial x}\left(f(x) + B(x)u_0 + a\right), \quad z(0) = -s_0(0). \tag{9}$$

From conditions (9), it is clear that the sliding mode is enabled from the initial time instant thus guaranteeing robustness in front of matched uncertainties for any $t \geq 0$ [9].

C. Some preliminaries on DRL

The main ingredients of a RL algorithm are the so-called *agent* and the *environment* represented by the *state* $\sigma_t \in \mathcal{S}$, with $\mathcal{S}$ being the state space. At each time instant t, the agent performs an action, namely $\alpha_t \in \mathcal{A}$, with $\mathcal{A}$ being the action space, according to a *policy* π. Then, the environment transits into a new state, namely $\sigma_{t+1} \in \mathcal{S}$, such that the action is evaluated with a *reward* function, namely r_{t+1}. The goal of the agent is to learn the policy which maximizes the total reward over an episode of length T_{h}, i.e., $R_t = \sum_{k=0}^{T_{\mathrm{h}}} \gamma^k r_{t+k+1}$, with $\gamma \in [0, 1]$ being a discount factor. The key element to learn such a policy is the so-called *state-action value function* $Q_\pi(\sigma_t, \alpha_t) = \mathbb{E}\{R_t \mid \sigma = \sigma_t, \alpha = \alpha_t\}$, which quantifies the expected long term reward starting from a certain state σ_t, taking action α_t and following policy π thereafter. Unfortunately, such a function is unknown and must be estimated. If both state space $\mathcal{S}$ and action space $\mathcal{A}$ are continuous, *Actor-Critic* methods can be employed, see e.g., [26], [27]. In particular, in such methods, two main components can be identified: an *Actor* and a *Critic*. The former is a DNN which represents the policy π according to which the agent selects the action, while the latter is a DNN which estimates $Q(\sigma_t, \alpha_t)$. The Actor parameters, i.e., the weights of the DNN, are then updated according to the direction suggested by the Critic network.

III. Drift Term Approximation

As mentioned in §II-B, in order to design an ISM control, the knowledge of the nominal model, i.e., of terms $f(x)$, $B(x)$ and a, is required. However, when considering robot manipulators, it is common to assume the inertia matrix $M(q)$, hence matrix $B(x)$ in (3), known, see e.g., [12], [13]. On the other hand, while vector a is known by definition, the drift term $f(x)$ is often partially or fully unknown, as in this paper. Therefore, the universal approximation property and a DRL approach are presented to estimate $f(x)$.

A. DNN function approximation

Given $x \in \Omega$, there exists an ideal DNN characterized by $k \in \mathbb{N}_{>0}$ hidden layers so that

$$f(x) = W^\top g_k(\Phi(x)) + \varepsilon(x), \tag{10}$$

where $W \in \mathbb{R}^{L_k \times 2n}$ are the ideal weights of the output layer, with L_k being the number of neurons in the kth hidden layer, $\varepsilon(x) : \Omega \to \mathbb{R}^{2n}$ is the so-called *function reconstruction error*, while $\Phi(x) \in \mathbb{R}^{L_k}$ is the output of the kth hidden layer. The latter is computed as

$$\Phi(x) = V_{k-1}^\top g_{k-1} \circ V_{k-2}^\top g_{k-2} \circ \cdots \circ V_1^\top g_1 \circ V_0^\top x, \tag{11}$$

with $V_0 \in \mathbb{R}^{2n \times L_1}$ being the ideal weights of the input layer, and $V_j \in \mathbb{R}^{L_j \times L_{j+1}}$, with $j = 1, 2, \dots, k-1$, being the ideal weights of the hidden layers. Moreover, the DNN is characterized by k bounded ideal activation functions $g_i : \mathbb{R}^{L_i} \to \mathbb{R}^{L_i}$, with $i = 1, 2, \dots, k$. The following assumption on the bounds of the ideal DNN needs to be introduced.

$\boldsymbol{\mathcal{A}_3}$: There exist known constants $\overline{W}$, $\bar{\varepsilon}_f$, $\bar{V}_j$, $\bar{g}_i \in \mathbb{R}_{>0}$, with $j = 0, 1, \dots, k-1$ and $i = 1, 2, \dots, k$, so that the ideal output layer weights W, the input layer weights V_0, the hidden layers weights $V_1, V_2, \dots, V_{k-1}$, the unknown ideal activation functions $g_1, g_2, \dots, g_k$, and the reconstruction error $\varepsilon(x)$ are bounded as

$$\sup_{x(t)\in\Omega}\|W\| \leq \overline{W}, \quad \sup_{x(t)\in\Omega}\|\varepsilon\| \leq \bar{\varepsilon}_f,$$
$$\sup_{x(t)\in\Omega}\|V_j\| \leq \bar{V}_j, \quad \sup_{x(t)\in\Omega}\|g_i\| \leq \bar{g}_i.$$

Since the ideal DNN weights are not known, an approximation of them can be used. Therefore, the unknown drift term is estimated as

$$\widehat{f}(x) = \widehat{W}^\top \widehat{g}_k(\widehat{\Phi}(x)), \tag{12a}$$

$$\widehat{\Phi}(x) = \widehat{V}_{k-1}^\top \widehat{g}_{k-1} \circ \widehat{V}_{k-2}^\top \widehat{g}_{k-2} \circ \cdots \circ \widehat{V}_1^\top \widehat{g}_1 \circ \widehat{V}_0^\top x, \tag{12b}$$

with $\widehat{\Phi}$ being the output of the last hidden layer of the network with approximated weights, while $\widehat{g}_i$, with $i = 1, 2, \dots, k$, are user-defined activation functions which may differ from the ideal ones. For the sake of readability, from now on the quantities $g_k(\Phi(x))$ and their estimates $\widehat{g}_k(\widehat{\Phi}(x))$ will be indicated as g_k and $\widehat{g}_k$, respectively. Moreover, the following assumption holds.

$\boldsymbol{\mathcal{A}_4}$: There exists a known constant $\bar{\bar{g}}_i \in \mathbb{R}_{>0}$, with $i = 1, 2, \dots, k$, so that the user-selected activation functions of the DNN are bounded as

$$\sup_{x(t)\in\Omega}\|\widehat{g}_i\| \leq \bar{\bar{g}}_i.$$

Finally, the weights estimation errors can be computed as

$$\widetilde{W} = W - \widehat{W}, \tag{13a}$$

$$\widetilde{V}_j = V_j - \widehat{V}_j, \quad i = 0, 1, \ldots, k-1. \tag{13b}$$

B. DRL agent for drift term estimation

Differently from [24], where a single hidden layer NN is adopted, in this paper a DNN with k hidden layers is introduced in (11). Although this could represent a further complication from computational viewpoint, the introduction of a DRL represents a valid assistant-tool for the online adaption law.

Due to the continuous nature of the considered problem, among different approaches, in this paper we rely on the so-called twin delayed deep deterministic policy gradient (TD3) Actor-Critic method introduced in [27]. In particular, making reference to §II-C, in our case the Actor network is exactly the one in (12). Then, since the objective of the agent is to estimate the drift term $f(x)$, the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are instead defined as $\mathcal{S} = \{x\}$, $\mathcal{A} = \left\{\widehat{f}(x)\right\}$, while, in order to train the DRL agent, the reward at time t is selected as

$$r_t = -\Delta s(t), \tag{14}$$

where $\Delta s(t) := \|s(t)\| - \|s(t^-)\| \in \mathbb{R}^n$, where $s \in \mathbb{R}^n$ is the sliding variable, discussed in the next section, while $s(t^-)$ indicates the immediately past value of the sliding variable itself. After the off-line training phase, that is when the learning procedure is completed, the DNN weights are extracted and used to initialize the DNN of the proposed online Lyapunov based control presented in the next section.

IV. The Proposed Lyapunov-Based ISM Control

We are in a position to introduce the proposed Lyapunov-based ISM control approach, as depicted in Fig. 1.

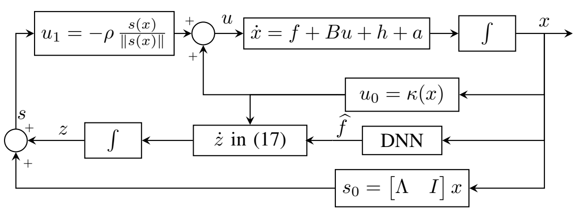

$u_1 = -\rho\,\frac{s(x)}{\|s(x)\|}$
$+$
u
$+$
$\dot{x} = f + Bu + h + a$
$\int$
x
$u_0 = \kappa(x)$
s
$+$
z
$\int$
$\dot{z}$ in (17)
$\widehat{f}$
DNN
$+$
$s_0 = [\Lambda \quad I]\,x$

Fig. 1. Block diagram of the proposed DRL-ISM control scheme.

First, referring to (8), the component s_0 is designed as

$$s_0(x) = \begin{bmatrix}\Lambda & I\end{bmatrix} x, \tag{15}$$

where $\Lambda \in \mathbb{R}^{n \times n}$ is a diagonal matrix with positive entries. Hence, it is possible to rewrite (9) as

$$\dot{z} = -\begin{bmatrix}\Lambda & I\end{bmatrix}\left(\hat{f}(x) + B(x)u_0 + a\right), \quad z(0) = -s_0(0), \tag{16}$$

which, exploiting (12), can be reformulated as

$$\dot{z} = -\begin{bmatrix}\Lambda & I\end{bmatrix}\left(\widehat{W}^\top \widehat{g}_k + B(x)u_0 + a\right), \tag{17}$$

where the nominal dynamics used to design the transient function is

$$\dot{x} = \widehat{W}^\top \widehat{g}_k + B(x)u_0 + a. \tag{18}$$

Given any suitable (in whatever appropriate sense in terms of robustness and performance) control law $u_0 = \kappa(x)$ (see, e.g., [25, Ch. 6]), the whole control law (6) is given by

$$u = \kappa(x) - \rho\,\frac{s(x)}{\|s(x)\|}. \tag{19}$$

As discussed in §III-B, all the weights of the DNN trained off-line are used to initialize the DNN in (12). Moreover, an adaptation law is introduced to perform the fine tuning of the weight of the outer layer $\widehat{W}$. Specifically, the adaption law is chosen as

$$\dot{\widehat{W}} = \Gamma \widehat{g}_k s^\top \begin{bmatrix}\Lambda & I\end{bmatrix}, \tag{20}$$

where $\Gamma \in \mathbb{R}^{L_k \times L_k}$ is a constant diagonal matrix with positive entries selected by the controller designer to determine the adaptation rate.

Finally, considering (15), deriving (8) with respect to time, substituting (3), (10), (17) and (19), and exploiting (13a), the dynamics of the sliding variable can be computed as

$$\begin{aligned}\dot{s} = \begin{bmatrix}\Lambda & I\end{bmatrix}\big(&\varepsilon + h - B(x)\rho\frac{s}{\|s\|} + \\ &+ W^\top(g_k - \widehat{g}_k) + \widetilde{W}^\top \widehat{g}_k\big).\end{aligned} \tag{21}$$

V. Stability Analysis

In this section, the main theoretical result is presented.

Theorem 1: Consider the robot dynamics expressed as in (3), with initial condition $x_0 \in \Omega$, control law (19), sliding variable as in (8), (15) and (17), and a pre-trained DNN (12) with adaptation laws for the outer layer weights (20). If $\boldsymbol{\mathcal{A}_1}$, $\boldsymbol{\mathcal{A}_2}$, $\boldsymbol{\mathcal{A}_3}$, and $\boldsymbol{\mathcal{A}_4}$ hold, and

$$\rho > \frac{n\lambda_{\max}(\Lambda + I)\left(\overline{\varepsilon} + \overline{h} + \overline{W}(\overline{g}_k + \overline{\widehat{g}}_k)\right)}{\underline{b}\lambda_{\min}(\Lambda + I)}, \tag{22}$$

then a sliding mode $s = 0$ is enforced.

Proof: Select the Lyapunov-like candidate function $v(x(t)) : \mathbb{R}^{2n} \to \mathbb{R}$ as

$$v(x) = \frac{1}{2}s^\top s + \frac{1}{2}\mathrm{tr}\left(\widetilde{W}^\top \Gamma^{-1} \widetilde{W}\right), \tag{23}$$

where s is the integral sliding variable defined in (8), and $\widetilde{W}$ as in (13a). Differentiating (23) with respect to time, one obtains

$$\dot{v}(x) = s^\top \dot{s} + \mathrm{tr}\left(\widetilde{W}^\top \Gamma^{-1} \dot{\widetilde{W}}\right). \tag{24}$$

Substituting (21) and given that $\dot{\widetilde{W}} = -\dot{\widehat{W}}$, the above equation becomes

$$\begin{aligned}\dot{v}(x) = s^\top \begin{bmatrix}\Lambda & I\end{bmatrix}\Big(&\varepsilon + h - B(x)\rho\frac{s}{\|s\|} + \\ &+ W^\top(g_k - \widehat{g}_k) + \widetilde{W}^\top \widehat{g}_k\Big) - \mathrm{tr}\Big(\widetilde{W}^\top \Gamma^{-1} \dot{\widehat{W}}\Big).\end{aligned} \tag{25}$$

Using the update law (20), one has

$$\dot{v}(x) = s^\top \begin{bmatrix}\Lambda & I\end{bmatrix} \Big(\varepsilon + h - B(x)\rho \frac{s}{\|s\|} + \\ + W^\top (g_k - \widehat{g}_k) + \widetilde{W}^\top \widehat{g}_k \Big) - \mathrm{tr}\Big(\widetilde{W}^\top \widehat{g}_k s^\top \begin{bmatrix}\Lambda & I\end{bmatrix} \Big). \tag{26}$$

Exploiting the property of the trace operator, the term depending on $\widetilde{W}$ is canceled, obtaining

$$\dot{v}(x) = s^\top \begin{bmatrix}\Lambda & I\end{bmatrix} \Big(\varepsilon + h - B(x)\rho \frac{s}{\|s\|} + W^\top (g_k - \widehat{g}_k) \Big). \tag{27}$$

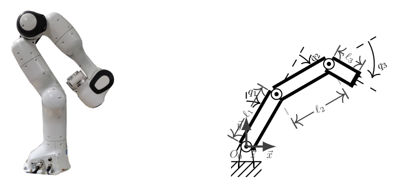
If $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{A}_4$ hold, then (27) can be upper bounded as

$$\begin{aligned} \dot{v}(x) \leq\ & s^\top \begin{bmatrix}\Lambda & I\end{bmatrix} \Big(\mathbb{1}_{2n\times 1}(\bar{\varepsilon} + \bar{h} + \overline{W}^\top (\bar{g}_k + \bar{\widehat{g}}_k)) \\ & -\rho \underline{b} \begin{bmatrix} I & I\end{bmatrix}^\top \frac{s}{\|s\|} \Big) \\ \leq\ & \mathbb{1}_{n\times 1}^\top [\Lambda \quad I] \mathbb{1}_{2n\times 1} \left(\bar{\varepsilon} + \bar{h} + \overline{W}^\top (\bar{g}_k + \bar{\widehat{g}}_k) \right)\|s\| \\ & - \rho \underline{b} s^\top \begin{bmatrix}\Lambda & I\end{bmatrix} \begin{bmatrix} I & I\end{bmatrix}^\top \frac{s}{\|s\|} \\ =\ & \mathbb{1}_{n\times 1}^\top [\Lambda \quad I] \mathbb{1}_{2n\times 1} \left(\bar{\varepsilon} + \bar{h} + \overline{W}^\top (\bar{g}_k + \bar{\widehat{g}}_k) \right)\|s\| \\ & - \rho \underline{b} s^\top (\Lambda + I) \frac{s}{\|s\|} \\ \leq\ & n\lambda_{\max}(\Lambda + I) \left(\bar{\varepsilon} + \bar{h} + \overline{W}^\top (\bar{g}_k + \bar{\widehat{g}}_k) \right)\|s\| \\ & - \rho \underline{b} \lambda_{\min}(\Lambda + I)\|s\| = -\eta \|s\|, \end{aligned} \tag{28}$$

with $\mathbb{1}_{*\times 1}$ being a column vector of all ones and $\eta := -(n\lambda_{\max}(\Lambda + I) \left(\bar{\varepsilon} + \bar{h} + \overline{W}^\top (\bar{g}_k + \bar{\widehat{g}}_k) \right) - \rho \underline{b} \lambda_{\min}(\Lambda + I))$. Hence, if condition (22) is satisfied, then $\dot{v}(x) \leq -\eta\|s\| < 0$, $\forall\, x \in \Omega$, implying that a sliding mode $s = 0$ is enforced, which concludes the proof. ■

VI. Simulations and Results

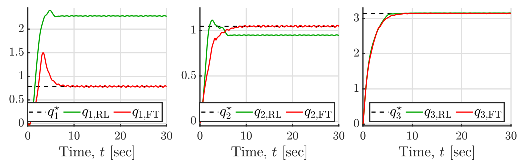
In this section, in order to assess the proposed NN-ISM control algorithm, simulations carried out relying on the model of a Franka Emika Panda robot (see Fig. 2) are illustrated and discussed. Only three joints and planar motions are considered, while the robot dynamics is derived from [28]. As for the parameters, they are obtained from direct measurements and from the analysis carried out in [29], so that the length of the links are $l_1 = 0.35\,\mathrm{m}$, $l_2 = 0.4\,\mathrm{m}$ and $l_3 = 0.15\,\mathrm{m}$, while the masses are $m_1 = 3.875\,\mathrm{kg}$, $m_2 = 4.814\,\mathrm{kg}$, $m_3 = 2.401\,\mathrm{kg}$.



(a) setup (b) schematic view

Fig. 2. Franka Emika Panda robot manipulator. Setup (a). Schematic view of a planar manipulator with three joints (b).

In order to estimate the drift term, the TD3 agent, implemented by using PyTorch, is trained for 500 episodes, each of them characterized by duration and time-step equal to 10 and 10^{-3} seconds, respectively. In each episode, the robot is controlled relying on the scheme in Fig. 1 so that it moves towards a random configuration in the joint space while being subject to a random matched disturbance h. Note that, in this phase, the adaptation law derived from stability analysis is not applied. The discontinuous control gain is instead designed large enough so that, in the ideal case of known drift term, the disturbance is widely rejected. Specifically, it is selected as $\rho = 25$ and kept constant during the whole training. Moreover, the gain matrix of the integral sliding variable is designed as $\Lambda = I_{n\times n}$, while the stabilizing part of the control law is chosen as $u_0 = G(q) - 15 \cdot \widetilde{q} - 20\dot{q}$. As for the TD3 agent, the Actor network, i.e., (12), consists of 6 inputs, 2 hidden layers with 400 and 300 neurons (see, [26]), respectively, and 6 outputs. The hidden layers and the output layer are characterized by $\tanh(\cdot)$ and linear activation functions, respectively. The Critic networks is finally defined as in [27]. Moreover, Actor



(a) Joint 1 (b) Joint 2 (c) Joint 3

Fig. 3. Time behaviour of the joint positions (expressed in rad) when fine tuning is not applied (green line) and when it is applied (red line).

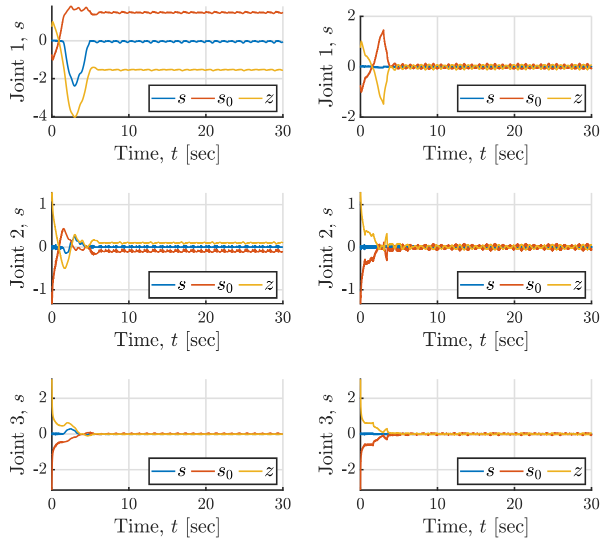


Fig. 4. Time behaviour of the sliding variable vector when no fine tuning is performed (left column) and when adaptation law (20) is applied (right column).

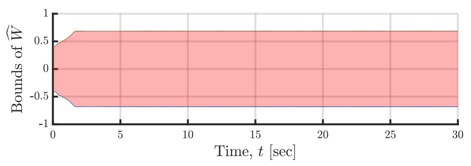


Fig. 5. Time behavior of the bounds of the outer layer weights.

and Critic networks are trained using Adam optimizer with learning rate equal to 10^{-5}.

After the off-line preliminary DRL-based tuning of the weights in (12), two different simulations are executed. In the first one, the DNN (12) is employed directly in the NN-ISM control without the online fine tuning (briefly, FT). In the second one, the adaptation law (20) is instead used to fine tune online the outer layer of the DNN (12). In both scenarios, the manipulator is controlled to reach the desired configuration $q^\star = \begin{bmatrix}\pi/4 & \pi/3 & \pi\end{bmatrix}^\top$ rad, while being subject to a disturbance τ_d chosen so that $h = \begin{bmatrix}0_{3\times 1}^\top & 0.7\cdot\sin(2\pi t) & -0.5\cdot\sin(4\pi t) & 0.5\cdot\cos(1.5\pi t)\end{bmatrix}^\top$. For both simulations, the controller parameters are selected equal to the ones used in the training phase. Moreover, in the simulation with fine tuning, the constant gain matrix is set as $\Gamma = 15\cdot I_{300\times 300}$. As it is possible to see from Fig. 3, when the online fine tuning is not adopted, the desired configuration is not achieved. This could happen for different reasons, like, e.g., sub-optimal design of the reward function or lack of exploration during the DRL training phase. If the online adaptation (20) is applied, the robot reaches the desired configuration and keeps it with very small oscillations. Moreover, Fig. 4 and Fig. 5 further validate the theoretical results introduced in Theorem 1. In particular, Fig. 4 shows that, differently from the case in which the fine tuning is not applied, when using the online adaptation law (20) a sliding mode is always enforced. To conclude, Fig. 5 shows that, for all the duration of the simulation, the time-varying weights $\widehat{W}$, determined by the adaptation law (20), remain bounded.

VII. Conclusions

In this paper, an enhanced version of the NN-ISM control algorithm in [24] is proposed for addressing the motion control problem of robot manipulators with partial known dynamics. In order to design the ISM control approach, the nominal model of the plant is indeed required. Thus, the integral sliding manifold is designed relying on a DNN, whose weights adaptation is initialized by a DRL-based offline training to estimate the unknown drift term. The theoretical analysis of the proposal is reported in the paper providing conditions for the integral sliding modes generation, and its assessment is finally illustrated relying on a realistic robot manipulator.

References

[1] A. Ferrara, G. P. Incremona, and M. Cucuzzella, *Advanced and optimization based sliding mode control: Theory and applications*. SIAM, 2019.

[2] V. Utkin and H. Lee, "Chattering problem in sliding mode control systems," in *International Workshop on Variable Structure Systems*, Alghero, Sardinia, Jun. 2006, pp. 346–350.

[3] A. Levant, "Higher-order sliding modes, differentiation and output-feedback control," *International Journal of Control*, vol. 76, no. 9-10, pp. 924–941, 2003.

[4] G. Bartolini, A. Ferrara, and E. Usai, "Chattering avoidance by second-order sliding mode control," *IEEE Transactions on Automatic Control*, vol. 43, no. 2, pp. 241–246, 1998.

[5] Y. Feng, F. Han, and X. Yu, "Chattering free full-order sliding-mode control," *Automatica*, vol. 50, no. 4, pp. 1310–1314, 2014.

[6] A. Pisano, M. Tanelli, and A. Ferrara, "Switched/time-based adaptation for second-order sliding mode control," *Automatica*, vol. 64, pp. 126–132, 2016.

[7] G. P. Incremona, M. Cucuzzella, and A. Ferrara, "Adaptive suboptimal second-order sliding mode control for microgrids," *International Journal of Control*, vol. 89, no. 9, pp. 1849–1867, 2016.

[8] G. P. Incremona, L. Mirkin, and P. Colaneri, "Integral sliding-mode control with internal model: A separation," *IEEE Control Systems Letters*, vol. 6, pp. 446–451, 2022.

[9] V. Utkin and J. Shi, "Integral sliding mode in systems operating under uncertainty conditions," in *35th IEEE Conference on Decision and Control*, vol. 4, Kobe, Japan, Dec. 1996, pp. 4591–4596.

[10] F. Castanos and L. Fridman, "Analysis and design of integral sliding manifolds for systems with unmatched perturbations," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 853–858, 2006.

[11] M. Rubagotti, A. Estrada, F. Castaños, A. Ferrara, and L. Fridman, "Integral sliding mode control for nonlinear systems with matched and unmatched perturbations," *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2699–2704, 2011.

[12] A. Ferrara and G. P. Incremona, "Design of an integral suboptimal second-order sliding mode controller for the robust motion control of robot manipulators," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2316–2325, 2015.

[13] A. Ferrara, G. P. Incremona, and B. Sangiovanni, "Tracking control via switched integral sliding mode with application to robot manipulators," *Control Engineering Practice*, vol. 90, pp. 257–266, 2019.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press Cambridge, 2015.

[15] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[16] B. Sangiovanni, G. P. Incremona, M. Piastra, and A. Ferrara, "Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 397–402, 2021.

[17] N. Sacchi, G. P. Incremona, and A. Ferrara, "Deep reinforcement learning of robotic prosthesis for gait symmetry in trans-femoral amputated patients," in *29th Mediterranean Conference on Control and Automation*, Bari, Italy, Jun. 2021, pp. 723–728.

[18] A. Coronato, M. Naeem, G. De Pietro, and G. Paragliola, "Reinforcement learning for intelligent healthcare applications: A survey," *Artificial Intelligence in Medicine*, vol. 109, p. 101964, 2020.

[19] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[20] F. Bonassi, M. Farina, J. Xie, and R. Scattolini, "On recurrent neural networks for learning-based control: Recent results and ideas for future developments," *Journal of Process Control*, vol. 114, pp. 92–104, 2022.

[21] K. Guo and Y. Pan, "Composite adaptation and learning for robot control: A survey," *Annual Reviews in Control*, vol. -, no. -, pp. –, 2022.

[22] F. Lewis, S. Jagannathan, and A. Yesildirak, *Neural network control of robot manipulators and non-linear systems*. Taylor & Francis, 1999.

[23] R. Safaric and K. Jezernik, "Trajectory tracking neural network controller for a robot mechanism and Lyapunov theory of stability," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, Munich, Germany, Sep. 1994, pp. 626–633.

[24] N. Sacchi, G. P. Incremona, and A. Ferrara, "Neural network-based practical/ideal integral sliding mode control," *IEEE Control Systems Letters*, vol. 6, pp. 3140–3145, 2022.

[25] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[27] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, Stockholm, Sweden, Jul. 2018, pp. 1587–1596.

[28] A. Calanca, L. M. Capisani, A. Ferrara, and L. Magnani, "MIMO closed loop identification of an industrial robot," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 5, pp. 1214–1224, 2010.

[29] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. De Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4147–4154, 2019.