Deep Reinforcement Learning Based Self-Configuring Integral Sliding Mode Control Scheme for Robot Manipulators

Bianca Sangiovanni[1], Gian Paolo Incremona[2], Antonella Ferrara[1] and Marco Piastra[1]

Abstract— **This paper deals with the design of an intelligent self-configuring control scheme for robot manipulators. The scheme features two control structures: one of centralized type, implementing the inverse dynamics approach, the other of decentralized type. In both control structures, the controller is based on Integral Sliding Mode (ISM), so that matched disturbances and uncertain terms, due to unmodeled dynamics or couplings effects, are suitably compensated. The use of the ISM control also enables the exploitation of its capability of acting as a "perturbation estimator" which, in the considered case, allows us to design a Deep Reinforcement Learning (DRL) based decision making mechanism. It implements a switching rule, based on an appropriate reward function, in order to choose one of the two control structures present in the scheme, depending on the requested robot performances. The proposed scheme can accommodate a variety of velocity and acceleration requirements, in contrast with the genuine decentralized or centralized control structures taken individually. The assessment of our proposal has been carried out relying on a model of the industrial robot manipulator COMAU SMART3-S2, identified on the basis of real data and with realistic sensor noise.**

I. Introduction

Motion control of robot manipulators is considered a classical topic in robotics. Depending on the type of joints, their natural mechanical decoupling properties, and the performances required by the control problem, there are generally two possible approaches: the decentralized or the centralized approach [1], [2]. The decentralized control scheme is typically used when manipulator joints present high transmission ratios, and high performances in terms of velocity and acceleration are not required. With this approach, the manipulator is seen as a composition of linear and decoupled Single-Input-Single-Output (SISO) systems, one for each joint of the robot, where nonlinearities and coupling effect are regarded as disturbances acting on the single joint. In contrast, the centralized control approach is used when manipulators, considered as Multi-Input-Multi-Output (MIMO) systems, do not present gear boxes at the joints and/or higher performances in terms of velocity and acceleration are required, which implies that nonlinearities and coupling effects among the joints are not negligible and need to be taken into account explicitly during control design. The centralized control approach is typically based on the so-called inverse dynamics control method, which performs a global feedback linearization and a decoupling of the controlled system [2].

In classical scenarios, having joints with a non negligible transmission ratio, the decision whether one should use the decentralized or the centralized approach must be taken a priori. However, it is possible that the performances in terms of velocity and acceleration required during the working cycle of the robot can vary, passing from low to very high demand, thus resulting in a loss of efficiency of the chosen control approach. Then, it may be advantageous to allow the control system to autonomously reconfigure itself at run-time, depending on the performances required during the different phases of the robot operation. This means moving the decision-making online, by conferring to the control system the capability of evaluating which control structure is more suitable at a given time instant. A similar approach has been presented in [3], exploiting a switching mechanism based on the estimate of the perturbation compared to a manually preset threshold.

The main contribution of the present paper is the design of the decision-making mechanism which makes the online self-configuration possible. Our proposal is to adopt a mechanism that relies on machine learning: specifically, the logic behind the switching between the decentralized control structure and the centralized one that we propose is based on the training of a neural network via a DRL algorithm. The idea is to seek for an optimal solution to guarantee the required performances while avoiding abrupt changes between the control structures present in the scheme, which may stress the mechanical system. The elements considered for the training phase are the torques, the tracking errors and the estimate of the term that accounts for coupling and matched disturbances. This estimate is suitably provided by exploiting the so-called "perturbation estimator" property of the Integral Sliding Mode (ISM) controllers [4] used to realize both the decentralized and centralized control structures present in the overall motion control scheme. Note that ISM, apart from providing the online estimation capability which is the key feature upon which the proposed self-configuring scheme is based, possesses several other appreciable properties, which will be suitably revised in this paper. A notable example of the advantages of ISM can be observed, for instance, when it is used in combination with Model Predictive Control (MPC), as in [5], [6], or in a Second Order Sliding Mode framework [7]. As for Sliding Mode Control (SMC) [8]–[10], it was already applied to solve control problems in robotics (see,

This is the final version of the accepted paper submitted for inclusion in the Proceedings of the 57th IEEE Conference on Decision and Control, Miami Beach, FL, USA, Dec., 2018. [1] B. Sangiovanni, A. Ferrara and M. Piastra are with Dipartimento di Ingegneria Industriale e dell'Informazione, University of Pavia, Via Ferrata 3–5, 27100, Pavia, Italy (`bianca.sangiovanni01@universitadipavia.it`, `{antonella.ferrara, marco.piastra}@unipv.it`)

[2] G. P. Incremona is with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133, Milano, Italy (`gianpaolo.incremona@polimi.it`)

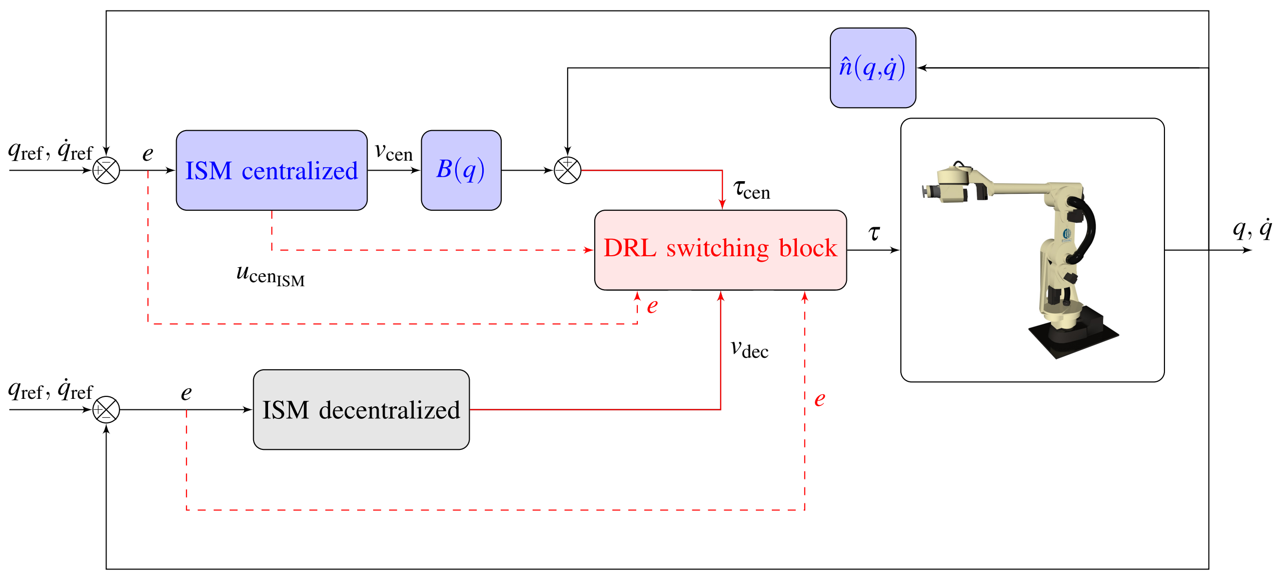


Fig. 1. The proposed multi-loop switching ISM control scheme with the signals needed to the DRL switching block (dashed red line) and the generated control torques signals (solid red line) generated by the centralized (blue blocks) and decentralized (gray block) architectures, respectively

e.g., [11]–[13]) in a classical setting. Coming to Artificial Intelligence (AI) methodologies, they have been used to solve a motion control problem for robot manipulators, for instance in [14], [15].

The present paper is organized as follows. In Section II some preliminary elements on Sliding Mode Control and DRL are reported. In Section III the proposed control scheme is described alongside the switching strategy adopted for the decision making, while in Section IV the ISM control is discussed and theoretically analyzed. Training results obtained relying on the model of a real robot manipulator COMAU-SMART3-S2 are illustrated in Section V. Some conclusions are reported in Section VI.

II. Preliminaries and Problem Formulation

In this section, preliminaries about the elements in the control structure and the machine learning framework adopted are illustrated, leading to the problem formulation.

A. Preliminary on Sliding Mode Control

In order to formulate the control problem, it is convenient to make reference to a canonical form frequently used in the design of SMC laws. Consider a SISO perturbed double integrator given by

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = v(t) + h(t) \\ y(t) = \sigma(x(t)) \end{cases} \tag{1}$$

where $x \in \Omega \subset \mathbb{R}^2$ is the state vector with $x(t_0) = x_0$, $v(t) \in \mathbb{R}$ is the input and $h(t) \in \mathbb{R}$ is a bounded matched uncertainty such that $|h(t)| \in \mathcal{H}$, with $\mathcal{H}$ being a compact set containing the origin, and $\mathcal{H}^{\text{sup}} \equiv \sup_{h\in\mathcal{H}}\{|h|\}$. The output function $\sigma(x) : \Omega \to \mathbb{R}$ is of class $\mathrm{C}(\Omega)$ and is called "sliding variable" in the following, that is the variable to steer to zero in a finite time in order to solve the control problem, according to classical SMC theory [8], [10]. The sliding variable $\sigma(x)$ has to be selected such that if $v(t)$ is designed so that, in a finite time t_{r} (ideal reaching time), $\sigma(x(t_{\mathrm{r}})) = 0\ \forall x_0 \in \Omega$ and $\sigma(x(t)) = 0\ \forall t > t_{\mathrm{r}}$, then $\forall t \geq t_{\mathrm{r}}$ the origin is an asymptotically stable equilibrium point of (1) constrained to $\sigma(x(t)) = 0$.

In the following sections the robot control problem under consideration will be formulated taking into account the structure (1), in order to be solved via a SMC law.

B. Elements of Reinforcement Learning

The main idea behind Reinforcement Learning (RL) is that an *agent*, relying on the knowledge acquired through past experience, understands which *actions* will lead to maximize a *reward* in a given time horizon, for any given situation (*state*) [16]. At each time step t, the agent and environment can be modeled as a *state* $s_t \in \mathcal{S}$ with $\mathcal{S}$ being the state space, containing all the relevant information needed. Starting from a given state, the agent performs an action $a_t \in \mathcal{A}$ with $\mathcal{A}$ being the action space, that affects the environment by changing its state. Before advancing to the next time step $t+1$, the agent receives a *reward* $r \in \mathbb{R}$, and moves to the new state s_{t+1}, according to the *dynamics* indicated as a probability distribution $P(s_{t+1}\,|\,s_t, a_t)$. A *reward* r_t is a scalar feedback signal that indicates "how well" an agent has done at step t. The agent's goal is to maximize the (expected) cumulative reward it receives in the long run. In case of episodic tasks with finite horizon T, the expected cumulative reward R_t is defined as

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{2}$$

where the term $0 \leq \gamma \leq 1$ is the *discount rate*, used to prioritize earlier rewards over later ones. Given the current state s, action a and the next state s', the expected value of the next reward is defined as:

$$r(s,a,s') = \mathbb{E}\left[r_{i+1}|s_t,a_t,s_{t+1}\right] \ . \tag{3}$$

A *policy* $\pi(a|s)$ is the probability that the agent will perform action a while in state s (i.e., the mapping from states to action). The expected, discounted cumulative reward the agent can accumulate in the long run, starting from a certain state and following the policy π, is called *value function*. For a given control task, the learning process is divided into *episodes*, where the agent interacts with the environment for a fixed number of time-steps before being reset. One of the main advantages of the RL method is that it does not rely on an annotated dataset like other supervised methods like SVMs, thus freeing the experiments from the burden of annotating time series of data.

C. Deep Reinforcement Learning

If the model of the environment (i.e., the transition probability) is not completely observable, which is usually the case of complex systems, a model-free approach is typically used. To this end, Q-learning is an off-policy algorithm that directly approximates the optimal action-value function (or Q-function) $Q^\star$ independently of the policy being followed. In case of continuous action problems with a large number of states, a parametric approximator of the Q-function is generally used; one way to build such approximator is a Deep Neural Network (DNN), i.e., a parametric function that can model complex non-linear relationships. The term *deep* refers to the level of composition of the parameters and the use of multiple hidden layers between the input and output ones.

D. The Robot Model

Consider now the dynamical model of a n-joints robot, given by

$$B(q)\ddot{q} + n(q,\dot{q}) = \tau \tag{4}$$
$$n(q,\dot{q}) = C(q,\dot{q})\dot{q} + F_{\mathrm{v}}\dot{q} + F_{\mathrm{s}}\,\mathrm{sgn}(\dot{q}) + g(q) \tag{5}$$

where $B(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ the centripetal and Coriolis torques, $F_{\mathrm{v}} \in \mathbb{R}^{n\times n}$ the viscous friction matrix, $F_{\mathrm{s}} \in \mathbb{R}^{n\times n}$ the static friction matrix, $g(q) \in \mathbb{R}^n$ the vector of gravitational torques, and $\tau \in \mathbb{R}^n$ the motor torques. The system (4)-(5) represents a MIMO nonlinear coupled model. In the following, the time dependence of the control variables has been omitted for the sake of simplicity ($q = q(t)$ and $\dot{q} = \dot{q}(t)$).

E. Problem Formulation

Given the robot manipulator model in (4)-(5), assume that q_{ref} and $\dot{q}_{\mathrm{ref}} \in \mathbb{R}^n$ are the reference signals for the joint variables and their first time derivative, specified a priori. It is assumed that the components of q_{ref} are bounded and $\dot{q}_{\mathrm{ref}}$ is Lipschitz continuous. The tracking errors are defined as

$$e_1 = q_{\mathrm{ref}} - q \tag{6}$$
$$e_2 = \dot{q}_{\mathrm{ref}} - \dot{q} \tag{7}$$

so that $e = \begin{bmatrix} e_1 & \dot{e}_1 \end{bmatrix}^\top = \begin{bmatrix} e_1 & e_2 \end{bmatrix}^\top$. In the following, let $e_j = \begin{bmatrix} e_{1_j} & e_{2_j} \end{bmatrix}^\top$ be the position error and the velocity error of joint j, $j = 1,\ldots,n$, with n number of joints in the manipulator. The control problem solved in this paper is a classical robot motion control problem, where the joints have to follow a given trajectory [2].

III. The proposed ISM based Self-Configuring Control Scheme

In this work the control problem formulated in the previous section is solved with the control architecture in Figure 1. Note that, without loss of generality, a control scheme designed in the joint space is considered. The first loop is characterized by a decentralized control structure (Mode 1) as described in Subsection III-A; the second loop, is based on the inverse dynamics based control structure (Mode 2) illustrated in Subsection III-B. The ISM controller in both structures is fed with errors e_1, e_2, which then provides the control signal $u_{\mathrm{dec_{ISM}}_j}(t)$ and $u_{\mathrm{cen_{ISM}}_j}(t)$ to the so-called switching block or decision maker based on the NAF learning algorithm, described hereafter.

A. Decentralized Control Structure

Let $K_{\mathrm{r}} \in \mathbb{R}^{n\times n}$ be the matrix of the gear ratios on the motor shafts such that the motor positions are $K_{\mathrm{r}}q = q_{\mathrm{m}}$, and let $K_\tau \in \mathbb{R}^{n\times n}$ be the matrix containing the torque constants, so that the shaft torques are $K_\tau\tau = \tau_{\mathrm{m}}$. Moreover, let $F_{\mathrm{m}} = K_{\mathrm{r}}^{-1}F_{\mathrm{v}}K_{\mathrm{r}}^{-1}$ be the matrix of the viscous friction coefficients referred to the motor shafts, and consider the inertia matrix $B(q)$ composed of a nominal component $\bar{B}$ and an unknown term $\Delta B(q)$, such that $B(q) = \bar{B} + \Delta B(q)$. The dynamic model (4)-(5) is now expressed as

$$\ddot{q} = (K_{\mathrm{r}}\bar{B}^{-1}K_{\mathrm{r}})\tau_{\mathrm{m}} - (K_{\mathrm{r}}\bar{B}^{-1}K_{\mathrm{r}})F_{\mathrm{m}}\dot{q} - (K_{\mathrm{r}}\bar{B}^{-1}K_{\mathrm{r}})d(q,\dot{q},\ddot{q}) \tag{8}$$

where the term that accounts for nonlinearities and coupling is

$$d = K_{\mathrm{r}}^{-1}\Delta B(q)K_{\mathrm{r}}^{-1}\ddot{q} + K_{\mathrm{r}}^{-1}C(q,\dot{q})K_{\mathrm{r}}^{-1}\dot{q} + K_{\mathrm{r}}^{-1}g(q) \ . \tag{9}$$

We can introduce and write the error model as

$$\dot{e}_2 = \ddot{q}_{\mathrm{ref}} - \ddot{q} \ . \tag{10}$$

By posing $h_{\mathrm{dec}} = \ddot{q}_{\mathrm{ref}} + (K_{\mathrm{r}}\bar{B}^{-1}K_{\mathrm{r}})F_{\mathrm{m}}\dot{q} + (K_{\mathrm{r}}\bar{B}^{-1}K_{\mathrm{r}})d(q,\dot{q},\ddot{q})$ and $v_{\mathrm{dec}} = -(K_{\mathrm{r}}\bar{B}^{-1}K_{\mathrm{r}})\tau_{\mathrm{m}}$, one has

$$\begin{cases} \dot{e}_{1_j}(t) = e_{2_j}(t) \\ \dot{e}_{2_j}(t) = v_{\mathrm{dec}_j}(t) + h_{\mathrm{dec}_j}(t) \\ y_j(t) = \sigma_j(e_j(t)) \end{cases} \tag{11}$$

with v_{dec_j} and h_{dec_j} being the j-th component of the vectors v_{dec} and h_{dec}, respectively. In (11), the generated torque will be also indicated as $v_{\mathrm{dec}} = \tau_{\mathrm{dec}}$. Note that one can assume $h_{\mathrm{dec}} \in \mathcal{H}_{\mathrm{dec}}$, with $\mathcal{H}_{\mathrm{dec}}$ being a compact set containing the origin and $\mathcal{H}_{\mathrm{dec}}^{\mathrm{sup}} \equiv \sup_{h_{\mathrm{dec}}\in\mathcal{H}_{\mathrm{dec}}}\{|h_{\mathrm{dec}}|\}$ being known.

B. Centralized Control Structure

The second architecture considered in this work is the so-called inverse dynamics based centralized control scheme.

Assume to exactly estimate the inertia matrix $B(q)$ and to have a quite accurate replica of the vector $n(q,\dot{q})$, such that $\hat{n}(q,\dot{q}) \neq n(q,\dot{q})$. Moreover, let v_{cen} be an auxiliary control vector such that the control torque is selected as

$$\tau_{\text{cen}} = -B(q)v_{\text{cen}} + \hat{n}(q,\dot{q}) \, . \tag{12}$$

Substituting (12) into model (4)-(5), one obtains

$$B(q)\ddot{q} + n(q,\dot{q}) = -B(q)v_{\text{cen}} + \hat{n}(q,\dot{q}) \, , \tag{13}$$

If one writes again $\dot{e}_2$ as in (10), one has

$$\begin{cases} \dot{e}_{1_j}(t) = e_{j_2}(t) \\ \dot{e}_{2_j}(t) = v_{\text{cen}_j}(t) + h_{\text{cen}_j}(t) \\ y_j(t) = \sigma_j(e_j(t)) \end{cases} \tag{14}$$

with v_{cen_j} and h_{cen_j} being the j-th component of the vectors v_{cen} and $h_{\text{cen}} = \ddot{q}_{\text{ref}} - B(q)^{-1}\left(\hat{n}(q,\dot{q}) - n(q,\dot{q})\right)$, respectively. Moreover, analogously to the decentralized case, one can assume $h_{\text{cen}} \in \mathcal{H}_{\text{cen}}$, with $\mathcal{H}_{\text{cen}}$ being a compact set containing the origin and $\mathcal{H}_{\text{cen}}^{\text{sup}} \equiv \sup_{h_{\text{cen}} \in \mathcal{H}_{\text{cen}}} \{|h_{\text{cen}}|\}$ being known.

C. Online Decision-Maker

The decision to switch to the decentralized structure (Mode 1) or to the centralized one (Mode 2) is made by a DNN trained using the Normalized Advantage Function algorithm.

The idea behind Normalized Advantage Function (NAF), introduced and illustrated in [15], is to design the Q-function in a way that computing $\operatorname{argmax}_a Q(s_t, a_t)$ does not imply a heavy computational load, which can be a critical requirement in time-sensitive real-world application scenarios The main advantage of the NAF algorithm is that it is suitable for continuous variables, both observed and controlled, and does not require discretization.

The observations, i.e., the *states* s_t, used by the algorithm are dynamical quantities retrieved from the control schemes, namely the torques exerted by the joints generated alternatively by both architectures (τ_{dec} and τ_{cen}), the root mean square of the position error signals e_{1_j}, and the coupling terms h_{dec_j} and h_{cen_j}. The action, in this case, is only a flag that determines on which control structure the scheme must switch. Since the decision procedure must be transparent with respect to the motion control problem that must be solved, in the sense that it should not have a negative impact on the robot performances, the reward function is defined in a way that encourages the least possible tracking error and a smooth transition between one scheme to the other, depending on the status of the robot. The elements used for the training process will be described with greater detail in section V.

IV. Control Design and Perturbation Estimation

Assuming that the robot manipulator is equipped with sensors and solvers, and that the torque exerted by each joint can be measured, in order to implement the proposed scheme, crucial to obtain an effective estimate of the term which accounts for the couplings and the unmodeled dynamics. To this end, the design of the ISM controller and its capability to estimate the uncertainties acting on a system, plays a fundamental role, as it will be discussed in the following.

A. Design of the Control Law

Consider the j-th joint of the robot. The goal is to design the ISM control laws to be used in the decentralized and centralized case. ISM is typically characterized by a control variable $v_j(t)$ split into two parts, i.e.,

$$v_j(t) = u_j(t) + u_{\text{ISM}_j}(t) \tag{15}$$

where $u_j(t)$ is generated by a suitable high level controller designed relying on the nominal model (i.e., the model of the plant - assuming that no uncertainty is present), and $u_{\text{ISM}_j}(t)$ is the sliding mode (typically discontinuous) control action, designed in order to reject the uncertainties affecting the system. Specifically, the latter is designed based on the errors e_{1_j}, e_{2_j} previously defined. The error model describing the dynamics of such errors can thus be written in compact form as

$$\dot{e}_j(t) = A_j e_j(t) + B_j\left(v_j(t) + h_j(t)\right) \tag{16}$$

where matrices A_j, B_j can be easily deduced in the decentralized and centralized cases making reference to (11) and (14), respectively. In system (16), $v_j(t)$ and $h_j(t)$ contain the components v_{dec_j} and h_{dec_j} or v_{cen_j} and h_{cen_j}, for $j = 1, \ldots, n$, depending on the case.

The so-called integral sliding manifold is selected as follows

$$\Sigma_j(t) = \sigma_j(t) + \varphi_j(t) = 0 \tag{17}$$

where Σ_j is the auxiliary sliding variable, $\sigma_j = m e_{1_j} + e_{2_j}$ is the actual sliding variable equal to the joint position error, with m being a positive constant, and the integral term φ_j is given by

$$\varphi_j(t) = -\sigma_j(t_0) - \int_{t_0}^{t} m e_{2_j}(\zeta) + u_j(\zeta) d\zeta \tag{18}$$

with the initial condition $\varphi_j(t_0) = -\sigma_j(e_j(t_0))$. Then, the discontinuous control law is defined as

$$u_{\text{ISM}_j}(t) = -K_j \operatorname{sgn}(\Sigma_j(t)) \, . \tag{19}$$

B. Perturbation Estimation

As anticipated in III-C, it is important to present the "perturbation estimator" property of ISM control. Note that the ISM controller is able to estimate the uncertain and coupling terms if the equivalent control is available [4]. As claimed in [4], it is shown that an approximation of the equivalent control can be obtained via a first order linear filter with the real discontinuous control (19) as input signal, i.e.,

$$\tilde{u}_{\text{ISM}_{\text{eq}_j}}(t) = \frac{1}{\mu}\int_{t_0}^{t} e^{-\frac{1}{\mu}(t-\zeta)} u_{\text{ISM}_j}(\zeta) d\zeta \tag{20}$$

where $\tilde{u}_{\text{ISM}_{\text{eq}_j}}(t_0) = 0$ and μ is the time constant of the filter, that should be set such that the linear filter does not distort the slow component of the switching action, which is $u_{\text{ISM}_{\text{eq}}}$.

Furthermore, the integral term has to be redesigned as

$$\varphi_j(t) = -\sigma_j(t_0) + \\ -\int_{t_0}^{t} me_{2_j}(\zeta) + u_j(\zeta) + \tilde{u}_{\mathrm{ISM}_{\mathrm{eq}_j}}(\zeta) - u_{\mathrm{ISM}_j}(\zeta)d\zeta \quad (21)$$

with initial condition $\varphi_j(t_0) = \sigma_j(e(t_0))$. It can be proved that $\tilde{u}_{\mathrm{ISM}_{\mathrm{eq}_j}} \simeq -h_j$, i.e., $\tilde{u}_{\mathrm{ISM}_{\mathrm{eq}_j}} = \hat{h}_j$. This quantity is used as a "state" (i.e observation) for the reinforcement learning algorithm described in subsection III-C, and will thus be a part of the decision making unit trained by the neural network.

V. Results

The experiments were carried out on a COMAU SMART3-S2 identified on the basis of real data [17].

The aim of the learning process is to find an optimal strategy for deciding which control structure present in the scheme is most suitable for meeting the demands in terms of tracking performances, while keeping into consideration that frequent variations of the control structure may cause excessive stress on the mechanical system. Nevertheless, it is also worth considering that the decentralized structure is in general less computationally expensive than the centralized one.

The *state space* used for the training with the DRL consists of the value of the coupling effects estimated by the ISM controller, as illustrated in Section IV-B, the torque exerted by each joint and the position, velocity and acceleration errors for each joint. The *action space* consists of a single value determining which control structure should be used at a given time instant; more specifically, 1 for the decentralized structure and -1 for the centralized one. The *reward function* is defined as:

$$r = c_1 r_{\mathrm{torque}} + c_2 r_e + c_3 r_{\dot{e}} + c_4 r_{\ddot{e}} + cost \quad (22)$$

where r_{torque}, r_e, $r_{\dot{e}}$ and $r_{\ddot{e}}$ are computed as the Euclidean norm of the normalized values of the torques, the position errors, the velocity errors and the acceleration errors of each joint, respectively. The terms $c_1 = 50$, $c_2 = 500$, $c_3 = 1000$ and $c_4 = 100$ are the weights associated to each component. The set of weights described has been defined via repeated experiments. As a matter of fact, the experiments showed that velocity errors are relatively more relevant than others for the overall performance. The *cost parameter* is representative of the extra computational workload that the centralized scheme may entail, due to the presence of the inverse dynamics described in Section III-B. Clearly, such cost may well depend on the specific application scenario considered, hence, the values described here are just exemplificative. In other words, in the DRL method proposed, the computational cost is a *parameter* that can be adapted to the specific case study.

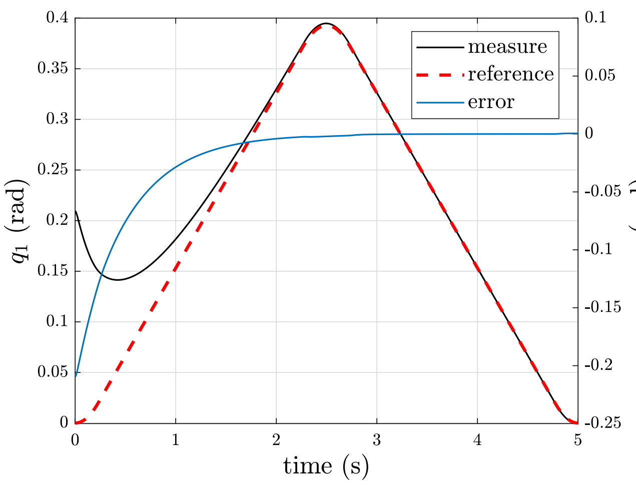
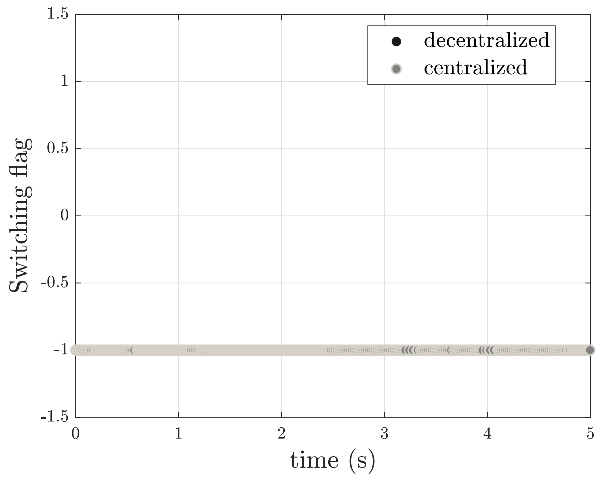
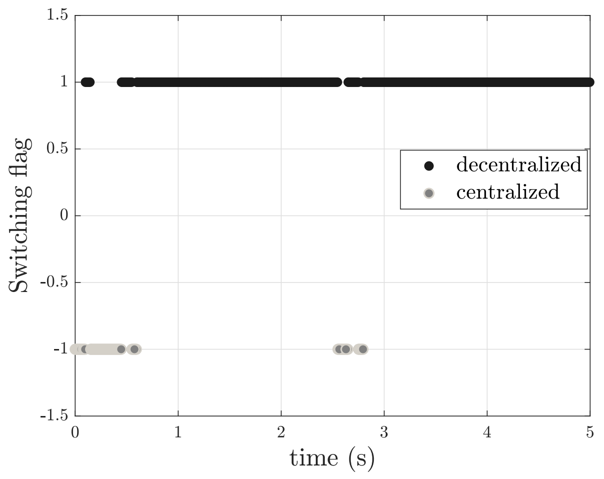


Fig. 2. Example of trajectory followed by a joint during a training episode



(a)



(b)

Fig. 3. Policy on a validation trajectory, with a null cost in the first case and a cost of -200 in the second

Each episode in the machine learning process consists of 100 steps of 0.05 s each. During each episode, the robot has to track a pick-and-place trajectory in the operative space defined by a random Trapezoidal Velocity Profile (TVP), such as the one in Fig. 2. The control structure selected at the initial time instant is chosen randomly as well. The policy derived from a learning session is then validated using a set of predefined trajectories, characterized by different TVP profiles.

As expected, when the *cost parameter* of the centralized scheme is set to zero, the policy obtained through DRL at the end of a training session tends to that of a fixed centralized approach, since such control scheme guarantees best performances in terms of velocity and acceleration. With increasing values of the *cost parameter* for the centralized approach, the agent becomes more leaning towards a decentralized one, thus entailing more switching actions, as reported in Fig. 3. With all non-zero values of the *cost parameter*, the actual amount of switching actions performed by the DRL agent can be controlled by acting on the c_1 in Eq. (22), since each switch between the two structures inevitably causes a peak in the torques exerted by the joints.

VI. Conclusions

This paper presents an advanced motion control scheme for robot manipulators, based on the switched use of a decentralized control structure and a centralized inverse dynamics based control. The scheme has a self-configuring capability thanks to the presence of a smart decision making mechanism which uses a switching rule to alternatively activate one of the two control structures. The rule relies on a DNN trained with the NAF algorithm for DRL. The proposal, which is then a combination of classical control concepts and AI elements, allows one to extend the operative velocity and acceleration range in which the robot manipulator can work. In this paper, the proposed approach has been validated relying on a model of an industrial COMAU SMART3-S2 anthropomorphic robot manipulator identified on the basis of real data.

References

[1] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. Wiley New York, 2006, vol. 3.

[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[3] A. Ferrara, G. P. Incremona, and B. Sangiovanni, "Integral sliding mode based switched structure control scheme for robot manipulators," in *Proc. 15th International Workshop on Variable Structure Systems*, Graz, Austria, July 2018.

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

[5] A. Ferrara, G. P. Incremona, and L. Magni, "A robust MPC/ISM hierarchical multi-loop control scheme for robot manipulators," in *Proc. IEEE 52nd Annual Conference on Decision and Control*, Florence, Italy, Dec. 2013, pp. 3560–3565.

[6] G. P. Incremona, A. Ferrara, and L. Magni, "MPC for robot manipulators with integral sliding modes generation," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1299–1307, 2017.

[7] 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.

[8] V. I. Utkin, *Sliding modes in control and optimization*. Springer Science & Business Media, 1992.

[9] V. Utkin, J. Guldner, and J. Shi, *Sliding mode control in electromechanical systems*. CRC press, 1999, vol. 34.

[10] C. Edwards and S. Spurgeon, *Sliding mode control: theory and applications*. CRC Press, 1998.

[11] A. Ferrara and L. Magnani, "Motion control of rigid robot manipulators via first and second order sliding modes," *Journal of Intelligent & Robotic Systems*, vol. 48, no. 1, pp. 23–36, 2007.

[12] L. M. Capisani, A. Ferrara, and L. Magnani, "Design and experimental validation of a second-order sliding-mode motion controller for robot manipulators," *International Journal of Control*, vol. 82, no. 2, pp. 365–377, 2009.

[13] G. P. Incremona, G. De Felici, A. Ferrara, and E. Bassi, "A supervisory sliding mode control approach for cooperative robotic system of systems," *IEEE Systems Journal*, vol. 9, no. 1, pp. 263–272, 2015.

[14] B. Sangiovanni, A. Rendiniello, G. P. Incremona, A. Ferrara, and M. Piastra, "Deep reinforcement learning for collision avoidance of robotic manipulators," in *Proc. 17th Annual European Control Conference*, Limassol, Cyprus, June 2018.

[15] S. Gu, T. P. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proceedings of the 33rd International Conference on Machine Learning*, New York, NY, USA, Jun. 2016.

[16] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[17] 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, 2011.