Research Article

Switched time delay control based on artificial neural network for fault detection and compensation in robot manipulators

Dihya Maincer[1] · Moufid Mansour[1] · Amar Hamache[2] · Chemseddine Boudjedir[3] · Moussaab Bounabi[4]

Received: 3 September 2020 / Accepted: 13 February 2021 / Published online: 6 March 2021

Abstract
This work proposes a switched time delay control scheme based on neural networks for robots subjected to sensors faults. In this scheme, a multilayer perceptron (*MLP*) artificial neural network (*ANN*) is introduced to reproduce the same behavior of a robot in the case of no faults. The reproduction characteristic of the *MLP*s allows instant detection of any important sensor faults. In order to compensate the effects of these faults on the robot's behavior, a time delay control (*TDC*) procedure is presented. The proposed controller is composed of two control laws: The first one contains a small gain applied to the faultless robot, while the second scheme uses a high gain that is applied to the robot subjected to faults. The control method applied to the system is decided based on the *ANN* detection results which switches from the first control law to the second one in the case where an important fault is detected. Simulations are performed on a SCARA arm manipulator to illustrate the feasibility and effectiveness of the proposed controller. The results demonstrate that the free-model aspect of the proposed controller makes it highly suitable for industrial applications.

Keywords Fault detection isolation (*FDI*) · Time delay and control (*TDC*) · Artificial neural network (*ANN*) · Multilayer perceptron (*MLP*) · Robot manipulator (*SCARA*)

1 Introduction

In various industrial processes robot manipulators have invaded the mode of technology; they are used to carry out complex and repetitive tasks quickly and efficiently [1, 2]. They are connected to each other by joints so that the manipulators follow the reference trajectory, where articulations must be precisely controlled. To perform these tasks, the manipulators are usually quite complex which increases their factor for fault. Thus, in order to have a good fault diagnosis on a manipulator, it is necessary to have a precise knowledge of its mathematical model. However, it is very difficult to obtain a precise of model as the modeling of dynamic robot which is not always an obvious task. For this purpose, various problems can arise such as uncertainties, external disturbances, uncertain dynamics and measurement noises, which cause deterioration of the fault detection performance by causing false alarms [3].

In this respect, fault detection in a robot manipulator arm is necessary for monitoring an effective support in utilization of a manipulators as independent systems [4, 5]. Methods of fault detection and isolation (*FDI*) are generally founded on the concept of production and residual analysis of the residuals. Many techniques have been assessed in order to be successfully applied.

✉ Dihya Maincer, dmaincer@usthb.dz; Moufid Mansour, m.mansour@city.ac.uk; Amar Hamache, hamache81@yahoo.fr; Chemseddine Boudjedir, chemseddine.boudjedir@g.enp.edu.dz; Moussaab Bounabi, moussaab.bounabi@g.enp.edu.dz | [1]Instrumentation and Automatic Control Department, Laboratory of Robots Parallelism Electroénergetic, University Of Science And Technology Houari Boumediene, Algiers, Algeria. [2]Laboratory of Vision Artificial and Automatic Systems, University Mouloud Mammeri Tizi-Ouzou, Tizi Ouzou, Algeria. [3]Laboratory of Process and Control, Polytechnic National School, Algiers, Algeria. [4]Photovoltaic Communication and Conversion Devices Laboratory, Polytechnic National School, Algiers, Algeria.

SN Applied Sciences (2021) 3:424 | https://doi.org/10.1007/s42452-021-04376-z

Taking into account the reliability which must be the most important criterion of the operation, these techniques allow reliable decisions to be made without knowledge of the mathematical system model. In this respect, artificial neural networks (*ANNs*) are suitable for such problems. One of this remarkable cleanliness is their ability to learn from their environment and improve their behavior from learning, in addition to the learning results in an adaptation (adjustment) of the weights and bias of the neural network [6, 7]. Ideally, after each learning step (iteration), performance improves. There are different learning approaches which differ from each other by the way of adjusting the weights, and their structure depends on the architecture of the neural network and the task to be performed. Besides, neural networks have been searched and carried out in real systems [8, 9]; there are many *ANN* applications in data analysis, identification and model control [10]. Amid various types of *ANN*, a *MLP* is quite popular and used extensively in research. In order to achieve good fault compensation, controllers capable of effectively compensating for faults are necessary to enable them to perform their task independently and realistically. In this regard, numerous experiments have been developed to compensate for fault such as robust control algorithms, including synchronization control [11], artificial neural networks (*ANN*) [12, 13], sliding mode control (*SMC*) [14, 15] and time delay control (*TDC*) [16, 17]. *SMC* are well known for their robustness against unknown systems dynamics.

To eliminate external perturbations and nonlinear dynamics with delay signal, handy nonlinear control strategies for unmodeled disturbances have been developed, for example *TDC*. This last, its principal objective is to use past observation of the system response as a control input at the present time to immediately change the control actions instead of identifying the parameters or adjusting the controller gain of the control system, which leads to an independent model controller, i.e., compensation without any use of dynamic model [18]. On the other hand, the big disadvantage of *TDC* is undesirable tracking errors and *TDE* errors. To compensate errors for *TDE*, many procedures have been tested by combining commands with a *TDC*. An auxiliary control [16–19] has been selected to settle its gains adaptively in order to have a switching control scheme. In [20] an auxiliary control in fuzzy sliding mode has also been chattering using fuzzy rules. In general, we caused that several works were realized by combining *TDC* and neural networks [21].

In other words, in order to eliminate *TDE* errors, a *SMC* [15] has been consolidated to allow quick tailoring of switching gains, which improves tracking performance compared to a conventional *TDC*. In particular, the use of fixed control gains from the *TDC* allows to ameliorate a performance of the system and the rapid adaptation of the gain [11]. *TDC* control combined with sliding mode [22, 23], requires a gain adjustment. However, the adaptive time delay control (*ATDC*) adaptation law does not directly reflect current tracking errors or sliding variables, which leads to a slow convergence rate. Thus, it would be useful to develop a *TDC* control combined with sliding mode, which compensates for the fault with a fast convergence speed, while suppressing *TDE* errors and avoiding.

In this paper, a methodology has been developed in order to Fault Detection, Isolation and compensate for sensor (*FDI*) in a robot manipulator, which is based on the concept of residual generation with neural network and compensation with a *TDC* controller. The network learning is based on nonlinear modeling and it allows the approximation of the real model in the absence of faults. The principal target behind this approach is to employ neural networks to observe the robotic system to detect and isolate all modification in system dynamics owing to faults. By utilizing the approximation capabilities of neural network, the network can be used for residue generation, in which trained trajectories are considered to perform system operation training. Another trajectory is used to test the efficiency of the network; as a result, the *MLP* network has the same operation as the faultless system.

The difference between the output of the manipulator with sensor faults and that of the neural network gives us an error which will then be compensated. In this context, and from these compared results, a global control scheme that switches between two control laws has been developed. The first control approach is applied to the fault-free manipulator with a low *TDC* gain. The results show immediate continuation of the robot trajectory. Then, with the same control method (low gain), we introduce some defaults on the manipulator, the results this time show a divergence (failure of the tracking of the trajectory). After that, the same TDC control scheme, but with a high gain is used in the case of fault existence on the robot manipulator. The simulation results show that the robot manipulator successfully track the desired trajectory even though the measurements are faulty. The main contribution of this paper is summarized as follows: (1) Model-free control scheme is proposed to detect and compensate the effects of sensors defaults. Indeed, both the *TDC* and the detection-based *ANN* are model independent, where only the inputs and outputs were used to achieve the required performances. (2) The control scheme switches between low gain and high gain in order to achieve the best performance between tracking error and eliminating the chattering phenomena. (3) Simulations conducted on *SCARA* robot are conducted to reveal the feasibility and the effectiveness of the proposed approach.

This paper is organized as follows: Sect. 2 describes the formulation of the problem and presentation of the *ANN*

and *TDC* schemes, together with the control design and convergence analysis. Section 3 presents the simulations and conducted on a SCARA robot manipulator together with their results. In the end, Section 4 concludes the paper.

2 Research method

2.1 Problem formulation

Fault diagnosis is necessary for robot manipulators, notably those performed in dangerous locations. Robotic manipulator errors can cause significant damages. Robots, therefore, want the capacity to detect isolate and compensate for faults efficiently and independently, so that they can continue to perform their tasks autonomously. This rescues cost and time for mending the robot. This type of stand-alone fault tolerance is also nifty for industrial robots, identifies faulty integrant or subsystems to speed up the reparation process, reduces down-time by tolerating faults and stops the robot to detrition products during manufacture.

For this reason, we will propose an effective method for the detection, isolation and compensation of sensor faults on a robot manipulator. The details of this method is presented below.

The dynamic equation of robot manipulator [24] is as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{1}$$

where $M(q)$ the inertia matrix, $C(q,\dot{q})$ is the matrix of Coriolis, and centrifuge $G(q)$ is the vector of gravitational force, $qR^n, \dot{q}R^n, \ddot{q}R^n$ are the angle, the angular velocity, and the angular acceleration of the joints, respectively, and τR^n is the control input torque. In this paper, the dynamic model of *SCARA* robot has been used in the simulation [24].

We assume that there is an additive sensors fault expressed by:

$$q_t = q + \Delta q \tag{2}$$

By replacing (2) in (1), we obtain the following dynamic equation of the manipulator:

$$M(q_t)\ddot{q}_t + C(q_t,\dot{q}_t)\dot{q}_t + G(q_t) = \tau_t \tag{3}$$

where Δq is the sensor fault, q_t is the measured joint and q is the joint fault-free. $M(q_t)$,

$C(q_t,\dot{q}_t)$, $G(q_t)$ and τ_t is the inertia matrix, the matrix of Coriolis and centrifuge, the vector of gravitational force and the control input torque, we introducing the fault (additive fault).

2.2 Artificial Neural Network (*ANN*)

The any regularly practiced neural models are the feed-forward perceptron worn in multilayer networks, i.e., the multilayer perceptron (*MLP*). The network is able of approaching all nonlinear unique static function to erratically desire precisely. This shape of mapping is well adapted for model gratitude applications, where the input vector and the output one be elected by spatial design that are independent of time. The institution of specific dynamics into this *ANN* demands the spatial portrayal of time. The *MLP* qualified with the back propagation algorithm is a very celebrate model in neural network and may be worn as a convenient system for executing a nonlinear input/output mapping of overall nature [8, 9], which is represented in the following figure:

For a p-dimensional input vector and a q-dimensional output vector, the *MLP* input/output connection specific a mapping from a p-dimensional Euclidean space to a q-dimensional Euclidean output space. Utilizing just one hidden layer, introducing in the n^{th} sample where $(n = 1, 2, \ldots, p)$, the input vector $I(n) = \left[x_1(n), x_2(n), \ldots, x_p(n)\right]^T$, the activation of the output neuron k (where $k = 1, 2, \ldots, q$) is:

$$O_k(n) = \varphi_k\left[\sum_{j=0}^{n} w_{jk}(n)\varphi_j\left[\sum_{i=0}^{p} w_{ji}(n)x_i(v)\right]\right] \tag{4}$$

where n is the number of neurons in the hidden layer, w_{ji} is the weight between the i^{th} neuron of the input layer and the jth neuron of the hidden layer, w_{jk} is the weight between the jth neuron of the hidden layer and the kth neuron of the output layer, w_j is the nonlinear activation function of the hidden layer and w_k is the nonlinear activation function of the output layer. characteristically, the network comprise of a set of input parameters that comprise the input layer, one or most hidden layers of calculation knot and an output layer of calculation knot. The input signal disseminates across the network in a forward direction on a layer-by-layer base. *MLPs* have been practiced to resolve certain difficult and various problems by training them in a supervised way with a very folk algorithm famous as the error back propagation algorithm.

2.3 Residual generation

The state equation of a nonlinear dynamic system fault, in discrete time, [8, 9] is given by:

$$x(t,\Delta t) = f(x(t), u(t)) \tag{5}$$

where x(t) is the state vector at time *t*, Δt is the sample assess, *u(t)* is the applied control vector and $f(.)$ is the vector-valued nonlinear function of the system faultless. Seeing that now that a fault *j* occurs, the dynamics of the system are changed to:

$$x(t, \Delta t) = h_j(x(t), u(t)) \tag{6}$$

where $h_j(.)$ is the vector-esteemed nonlinear function of the system assumed by fault j. The faults can be additive inputs. The *j*th fault vector may be specified as the difference between the faulty system dynamics (5) and the faultless system dynamics (4):

$$\varepsilon_j = h_j(x(t), u(t)) - f(x(t), u(t)) \tag{7}$$

Evidently, for the faultless system $\varepsilon_J(t + \Delta t) = 0$, usually, for every possible fault *j*, the fault vector ε_j has a particular way of behaving, appealed the fault signature. For the identification of the fault type, the fault vector must be computed and analyzed (Fig. 1). Thus, the dynamic performance of the faultless system (4) must be estimated (for example, by the mathematical model or by an *ANN*). After that, when fault *j* happen, the residual vector may be calculated as:

$$\dot{\varepsilon}_j(t + \Delta t) = x(t + \Delta t) - \hat{x}(t + \Delta t) = h_j(x(t), u(t)) - \hat{f}(x(t), u(t)) = \varepsilon_j(t) + e_j(x(t), u(t)) \tag{8}$$

where $\hat{f}(.)$ is a vector that portray the input–output mapping of the estimated faultless.

dynamic conduct of the system and e_j is the error between the actual faultless way of behaving and the estimated one. In actual systems and faultless case, the error is owing to unmodeled system incertitude, measurement noise, external disturbances and mapping errors (or modeling errors in model-based systems).

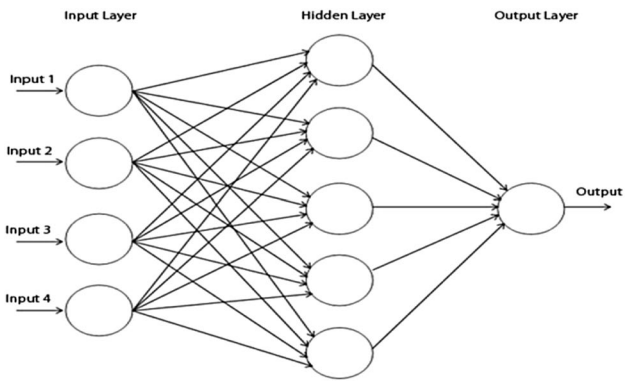

Fig. 1 *MLP* feed-forward neural network

2.4 Residual generation in mechanical manipulators

The dynamic of a faultless robotic manipulator with sensors in each joint is given by:

$$\dot{x} = \begin{pmatrix} \dot{q}_t \\ q_t \end{pmatrix} = \begin{bmatrix} \dot{q}_t \\ M^{-1}(q_t)\left[\tau_t - c\left(q_t, \dot{q}_t\right) - G\left(q_t\right)\right] \end{bmatrix} \tag{9}$$

where *q* is the vector of joint angular positions, $M\left(q_t\right)$, $C\left(q_t, \dot{q}_t\right)$, $G\left(q_t\right)$ and τ_t is the inertia matrix, the matrix of Coriolis and centrifuge, the vector of gravitational force and the control input torque. As the robot joints accelerations are not usually measured in robotic arm manipulators.

2.5 Residual analysis

In this paper, a residual analysis for fault detection is also executed with MLP. The global architecture is exposed in Fig. 2 [8, 9].

Outputs 1 across *q-1* correspond to the *q-1* possible fault modes, while output *q* corresponds to faultless operation. The *ANN* output *i (i = 1… q-1)* is trained to present a '0' in case fault *i* happen and '1' other. The output *q* is instructed to present a '0' in case fault less operation and '1' otherwise.

The *FDI* procedure can be summarized as follows:

1. Joint positions is measured at time step *t*;
2. The normalized positions and the torques experienced at *t* are introduced to the *MLP*, which estimates the positions at $t + \Delta t$;
3. The *MLP* outputs are compared with the normalized positions measured at $t + \Delta t$ to generate the residuals;
4. The MLP classified the residual and generated a vector that, when analyzed under the fault criterion, indicates

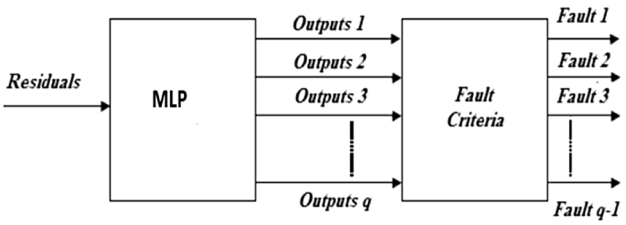

Fig. 2 Residual analysis employing architecture

the operation status of the system (faultless or faulty, along with the indication of the fault).

5. The time step t is incremented; return to step 1.

2.6 Fault indication criterion

Usually, the fault is indicated when a threshold is traversed. Besides, as the faults and the mapping errors are mostly correlated with the system dynamics, a small threshold may be a source of untrue alarms, while a large one may conceal the fault effects. The fault isolation scheme introduced up will advisable the fault information in the *ANN*s outputs. It is known, however, that untrue alarms can happen, and one cannot count on isolated information to decide on whether a failure literally happened. In this study we adopt the following fault criterion: a fault is said to have happened whenever one of the *ANN* outputs is large than the other ones for h successive time steps, where h is effectuated by process and error (researching for a good come to terms between untrue alarm assess and detection delays).

Remark In this paper, we consider a sensor fault on one or two of the manipulator joints. In other words, a loss of torque occurs on a joint. This fault can be caused, for example, by a mechanical fault in a drive system.

2.7 Time delay control (*TDC*)

In this paper, our objective is to synthesize a control law which is able to compensate the sensor fault in the manipulator based on the residuals between the real measurements and the neural network model and hence achieve good tracking performance [18].

Multiplying both parts of (3) by $\overline{M}^{-1}(q_t)$ and resolving for $\ddot{q}_t$, we obtain:

$$\tau_t = C(q_t, \dot{q}_t)\dot{q}_t + G(q_t) + \left(M(q_t) - \overline{M}(q_t)\right)\ddot{q}_t + \overline{M}(q_t)\ddot{q}_t \tag{10}$$

where t indicates the actual sample and $\overline{M} = diag\left(\overline{M}_1, \overline{M}_2, \ldots, \overline{M}_n\right) R^{n\times n}$ is a diagonal positive matrix.

We take up the common supposition that the robot dynamics in (3) comply with.

$\delta_m \le M(q_t) \le \delta_M$[25], for certain positive values δ_m and δ_M. This is since the inertia matrix $M(q_t)$ is voiced in terms of $\sin(q_t)$ and $\cos(q_t)$.

From (10) we write a compact and simple form of $\ddot{q}_t$ as follows:

$$\ddot{q}_t = N_t + \overline{M}^{-1}\tau_t \tag{11}$$

where $N_t = -\overline{M}^{-1}\left[C(q_t, \dot{q}_t)\dot{q}_t + G(q_t)\right] - \overline{M}^{-1}\left[\left(M(q_t) - \overline{M}\right)\ddot{q}_t\right] \in R^n$, the control approach purpose is to construct the joint angles q_t of a manipulator pursue the reference $q_{\mathrm{ref},t}$ accurately, so that the tracking error $e_t = q_{\mathrm{ref},t} - q_t R^n$ should be as close as possible to zero. Since $\mathrm{N_t}$ in (11) is not available, we use its estimate $\mathrm{N_{t-L}}$.

The *TDC* controller is expressed as [18]

$$\overline{\tau}_{\mathrm{t}} = -\overline{\mathrm{M}}\mathrm{N}_{t-L} + \overline{M}\left(\ddot{q}_{ref,t} + k_d\dot{e}_t + k_p e_t\right) \tag{12}$$

where $k_d = \mathrm{diag}\left(k_{d_1}, k_{d_2}, \ldots, k_{d_n}\right) \in R^{n\times n}$ and $k_p = \mathrm{diag}\left(k_{p_1}, k_{p_2}, \ldots, k_{p_n}\right) \in R^{n\times n}$ are positive concept matrices, $\ddot{q}_{\mathrm{ref},t} = \left[\ddot{q}_{\mathrm{ref}_1,t}, \ddot{q}_{\mathrm{ref2},t}, \ldots, \ddot{q}_{\mathrm{ref}_n,t}\right] \in R^n$ is the desired angular acceleration,$\dot{e}_t R^n$ is a by-product of the tracking error,$N_{t-L} = \left[N_{1,t-L}, \ldots, N_{n,t-L}\right] \in R^n$. The $N_{1,t-L}$ is the estimate of N_t in (11) obtained by a delayed measurement of the sample, called Time delay estimation (*TDE*) [18, 19].

The expression of $N_{1,t-L}$ is given by:

$$N_{1,t-L} = \ddot{q}_{\mathrm{t-L}} - \overline{\mathrm{M}}^{-1}\tau_{\mathrm{t-L}} \tag{13}$$

where L is a sampling time period, $t - L$ is a sample passed. Replacing (13) in (12), we obtain the following recursive control:

$$\overline{\tau}_t = -\overline{M}\ddot{q}_{t-L} + \tau_{t-L} + \overline{M}\left(\ddot{q}_{\mathrm{ref},t} + k_d\dot{e}_t + k_p\mathrm{e_t}\right) \tag{14}$$

which is often called *TDC* [18].

It should be noted that the TDE is bounded, if the control gain $\overline{M}$ is select to gratify the ensuing condition:

$$I - M^{-1}(q_t)\overline{M} < 1 \tag{15}$$

Formally, $t \ge 0$; also, the *TDE* error is limited by constant N_i^* for all $i = 1, 2, \ldots, n$, i.e., $\left|N_{i,t} - N_{1,t-L}\right| \le N_i^*$[26, 27]. It implies as the control gains ought to be selected to warrant the bounded ness of TDE errors. Thus, generally, little fixed control gains are utilized to gratify the inequality (15). Besides, if control gains are improperly little, the following performance degrades. Contrary, if those become improperly tall for quick response, those tend to produce a system unsteady.

2.8 Control design and convergence analysis

In this section, we will exploit the control design used in this paper together with the convergence analysis.

2.8.1 Control design

To achieve the control objectives outlined in the previous section, we must first determine the following sliding variable [28, 29]:

$$s_t = \dot{e}_t + K_d e_t \tag{16}$$

where $s_t = \left[s_{t,1}, s_{t,2}, s_{t,3}\right] \in R^3$, $K_d = \mathrm{diag}\left(K_{d,1}, K_{d,2}, K_{d,3}\right) \in R^3$ and $e_t = [[e_{t,1}, e_{t,2}, e_{t,3}] \in R^3]$. It is renowned that K_d in (16), is a conception parameter to be specified for ensuring the stability. In terms of the sliding variable s_t in (16), we build the following control scheme [19].

The proposed switching *TDC*-based neural network algorithm is given as follows:

The main objective of this paper is to suggest a global control method which switches between two controllers the first one (17) is applied to the manipulator fault-free, introducing a low gain. The second controller (18) is applied to the manipulator with a fault, but a large gain has been introduced.

If

$$e_{1,t} \le \alpha \text{ and } e_{2,t} \le \alpha \text{ and } e_{3,t} \le \alpha$$

Then,

$$\tau_t = \tau_{t-L} - \overline{M}\ddot{q}_{t-L} + \overline{M}\left(\ddot{q}_{\mathrm{ref}-t} + K_d\dot{e}_t + K_1 \mathrm{sign}\left(s_t\right)\right) \tag{17}$$

Else

$$\tau_t = \tau_{t-L} - \overline{M}\ddot{q}_{t-L} + \overline{M}\left(\ddot{q}_{\mathrm{ref},t} + K_d\dot{e}_t + K_2 \mathrm{sign}\left(s_t\right)\right) \tag{18}$$

End

where $\mathrm{sign}\left(s_t\right) = \left[\mathrm{sign}\left(s_{1,t}\right), \mathrm{sign}\left(s_{2,t}\right), \ldots, \mathrm{sign}\left(s_{n,t}\right)R^n\right.$ is defined as

$$\mathrm{sign}\left(s_{i,t}\right) = \begin{cases} 1 \text{ if} s_{i,t} \ge 0 \\ -1 \text{ if} s_{i,t} < 0 \end{cases}$$

$K_1 = \mathrm{diag}\left(K_{1,1}, K_{2,1}, K_{3,1}\right) \in R^3$, $K_2 = \mathrm{diag}\left(K_{1,2}, K_{2,2}, K_{3,2}\right) \in R^3$ are a positive constant switching gain matrix for guarantying stability, $\lambda_{\min}\left(K_2\right) > \lambda_{\max}\left(K_1\right)$, λ indicates the proper value of a matrix, $M_b = \mathrm{diag}\left(M_{1,b}, M_{2,b}, M_{3,b}\right) \in R^3$ is a control gain to be updated in-line according to the adaptive law, and α is the threshold not to be exceeded.

It should be noted that the results of the neural network outputs compared to those of the real system outputs generate residuals which are processed as to be compared. Our proposed controller switches between two controls schemes. The first one (Eq. (17)) is applied on a system fault-free, where the TDC uses a small gain, in order to achieve good tracking performances with a smooth control signal. The second control scheme (18) is developed on a faulty system. In this case, we introduce a large gain, in order to compensate the fault and external disturbances effects. The switching between the two schemes is based on the residuals values.

2.8.2 Convergence analysis

Consider the dynamic model of the robot manipulator described as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + d = \tau \tag{19}$$

where d is the external disturbance.

Let us consider that we have a sensors fault expressed in Eq. (2).

Hence, the dynamic model with sensors fault is given below:

$$M\left(q_t\right)\ddot{q}_t + C\left(q_t,\dot{q}_t\right)\dot{q}_t + G\left(q_t\right) + d_1 = \tau \tag{20}$$

where d_1 is the lumped disturbance which is expressed by:

$$d_1 = M(q)\ddot{q} + C(q,\dot{q}) + G(q) - M\left(q_t\right)\ddot{q}_t - \left(C\left(q_t,\dot{q}_t\right)\dot{q}_t + G\left(q_t\right)\right) + d \tag{21}$$

Therefore, N_t is given by:

$$N_t = -\overline{M}^{-1}\left(C\left(q_t,\dot{q}_t\right)\dot{q}_t + G\left(q_t\right) + d_1\right) - \overline{M}^{-1}\left[\left(M - \overline{M}\right)\ddot{q}\right] \tag{22}$$

Then, the *TDE* is used to estimate N_t, which allows us to evaluate the dynamics model without requiring any prior knowledge neither on the system nor on the sensors fault. Furthermore, *TDC* can use this estimation to compensate the effect of the lumped disturbances and achieve a good tracking performance.

Theorem *Consider the dynamic model of robot (1) subjected to external disturbances and sensors fault. The control law given by* (17) *and* (18), *guarantees the convergence asymptotically of the tracking error and the tracking error rate.*

Proof. In order to prove above theorem, we use the Lyapunov function applicant as specified below:

$$v = \frac{1}{2}s_t^T s_t = \frac{1}{2}\sum_{i=1}^{3} s_{t(i)}^2 \tag{23}$$

Its time derivative is given by:

$$\dot{v} = s_t^T \dot{s}_t \tag{24}$$

Using Eqs. (16) and (24) lead to:

$$\dot{v} = s_t^T\left(\dot{e}_{tt} + K_d\dot{e}_t\right) = s_t^T\left(\ddot{q}_{\mathrm{ref},t} - \ddot{q}_t + K_d\dot{e}_t\right) \tag{25}$$

Replacing in (25) $\ddot{q}_t$ by its expression deduced from (11), we obtain:

$$\dot{v} = s_t^T\left(\ddot{q}_{\mathrm{ref},t} + K_d\dot{e}_t - \overline{M}^{-1}\tau_t - N_t\right) \tag{26}$$

By applying the control law given by (18) we obtain:

$$\dot{v} = s_t^T\left(\ddot{q}_{ref,t} + K_d\dot{e}_t - N_t - \overline{M}^{-1}\left(\tau_{t-L} - \overline{M}\ddot{q}_{t-L} + \overline{M}\left(\ddot{q}_{ref,t} + K_d\dot{e}_t + K_2\mathrm{sign}\left(s_t\right)\right)\right)\right) \tag{27}$$

Hence,

$$\dot{v} = s_t^T\left(-N_t - \overline{M}^{-1}\left(\tau_{t-L} - \overline{M}\ddot{q}_{t-L}\right) - K_2\mathrm{sign}\left(s_t\right)\right) \tag{28}$$

From the definition of N_t Eq. (11), we can get the following:

$$\dot{v} = s_t^T\left(-N_t + N_{t-L} - K_2\mathrm{sign}\left(s_t\right)\right) \tag{29}$$

We obtain:

$$\dot{v} = s_t^T\left(-N_t + N_{t-L} - K_2\mathrm{sign}\left(s_t\right)\right) \tag{30}$$

Since $\left|\left|N_t - N_{t-L}\right|\right|_\infty \leq N^*$ as demonstrated in [26, 27]. Hence,

$$\dot{v} \leq \sum_1^3 |s_i|\left(N^* - K_{2,m}\right) \tag{31}$$

where $K_{2,m}$ is the minimum eigenvalue of the matrix K_2. Hence, if we choose $k_{2,m} > N^*$, then $\dot{V}$ becomes defined negative. Consequently, from the definition of v we can conclude that s_t converge asymptotically to zero:

$s_t = \dot{e}_t + K_d e_t \to 0$, as $t \to \infty$

As a result, the tracking error and the tracking error rate converge asymptotically to zero:

$e_t \to 0, \dot{e}_t \to 0$ as $t \to \infty$

3 Simulations and results

In order to test the control scheme presented in this paper, we consider a manipulator arm with 3 degrees of freedom, called *SCARA* as described in Sect. 1 of this paper. The control approach purpose here is to force the robot to follow a reference trajectory created in advance.

The manipulator's parameters are given as follows: $m_1 = 0.5, m_2 = 0.3, m_3 = 0.1$(Kg). The length of the links is set to be $L = 1m$, the moments of inertia are $I_1 = 0.02, I_2 = 0.03, I_3 = 0.05$(rad/s).

For the detailed expressions of the inertia matrix M, Coriolis and centripetal matrix C, and the gravitational vector G, please refer to [24]. The sampling time is equal to $t_s = 0.01s$.

The reference trajectory is defined as follows:

$$\begin{cases} X = R\sin(\theta)\cos(\vartheta) + x_0 \\ Y = R\sin(\theta)\sin(\vartheta) + y_0 \\ Z = R\cos(\theta) + z_0 \end{cases}$$

where R : circle radius, θ, ϑ: the angles are auxiliary parameters which are used to determine the position and the orientation of the terminal tool in the coordinate system linked to the sphere.

$[\,x_0\ y_0\ z_0\,]^T$ is the Cartesian coordinates of the tool in the coordinate system of the base.

In the following two sections, we will present the results of the simulations carried out using on the MATLAB software, the method proposed for the detection and compensation of faults in the robot arm. This method was presented in the previous sections of this paper.

3.1 Simulation Results of the fault detection and isolation

In order to detect the faults in our system (the SCARA robot arm), we have chosen to use the following neural network design: One input layer that contains 9 neurons (three articular positions, three articular velocities and three articular couples measured in t), one hidden layer, with 15 neurons, and two output layer with 6 neurons (three articular positions and three articular velocities at $(t+\Delta t)$). The *MLP* was designed with a back propagation algorithm. The learning set is made by simulating 4 flawless spherical trajectories, introducing the input / output of each trajectory. After the training, the model obtained is then tested and validated using a fifth spherical trajectory by introducing only the input. As a result, the trajectory was immediately followed, as the *MLP* produced a copy of the dynamic manipulator behavior, in which the neural network has the same functionality as the manipulator. Figures 3, 4, 5 and 6 present the normalized trajectory and joint positions of the *MLP* outputs in the simulation with a fault-free training trajectory.

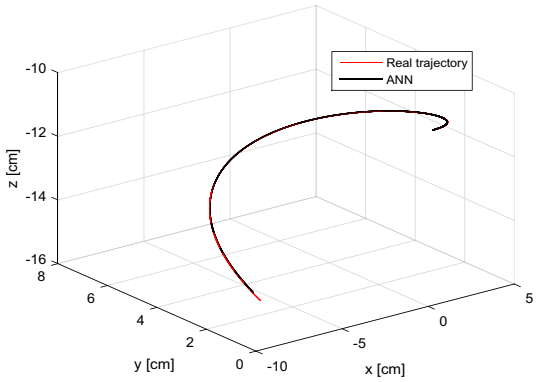

Fig. 3 Tracking trajectory in task space

Figure 3 represents the set trajectory to be followed together with the neural network output trajectory; it is noted that the neural network output trajectory follows exactly that of the manipulator's. Figure 4a–c represents the real and neural network trajectories following the x-axis, y-axis and z-axis, respectively. It is clear again that the neural network output trajectory is the same of that of the real robot manipulator.

The *MLP* is set to generate accurate models of the system under normal (fault-free) operating conditions. The comparison between the output of the network h_j (with fault) and the output network f (fault-free) gives the error vector $\varepsilon_j = h_j - f$.

The suggested procedure is executed online so that faults can be detect and isolate right away. In order to simulate a sensor fault, we add $\Delta q = 0.001°$ as a failure on a sensor which leads to a significant reduction or rise in the torque, resulting in anomalous variations of the residue. This residue is compared to some set values, and if the comparison is positive, it conducts to the detection and isolation of the fault. Once the fault detection and isolation has been carried out, one proceeds to the fault compensation using the TDC which is presented in the following section.

3.2 Simulation result of compensation

The simulation is conducted in three cases:

3.2.1 Case 1 (fault-free)

This case is considered as the nominal one, where the system is not subjected to any faults ($\Delta q = 0$). The *TDC* controller's gains are chosen as follows:

$M_b = \text{diag}[0.03, 0.03, 0.03]$, $K_d = [1.9, 1.9, 1.9]$, $K_1 = \text{diag}[1.5, 1.5, 1.5]$ and $K_2 = \text{diag}[6.1, 6.1, 6.1]$. Where as α is set to $\alpha = 0.2$

In this case, only the control law (17) is applied.

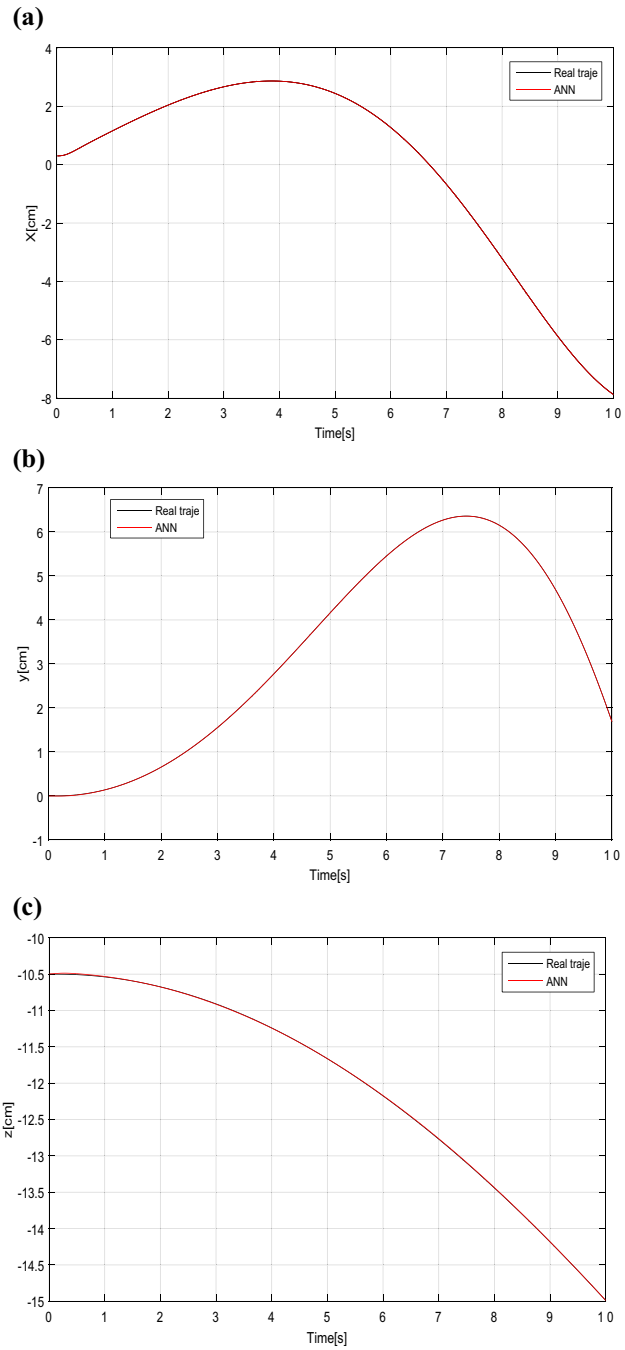

Fig. 4 **a**Tracking trajectory for the x-axis, **b** tracking trajectory for the y-axis, **c** tracking trajectory for the z-axis

The simulation results are shown in Figs. 5 and 6. Figure 5 presents the tracking trajectory in the operational space, and Fig. 6 presents the tracking trajectory in the axes x, y, and z, respectively. It is observed that the control law given by (17) can ensure good tracking performances, where a small tracking error is obtained in the three directions. Meanwhile, Fig. 7 indicates that the control torque signal stays in the admissible values.

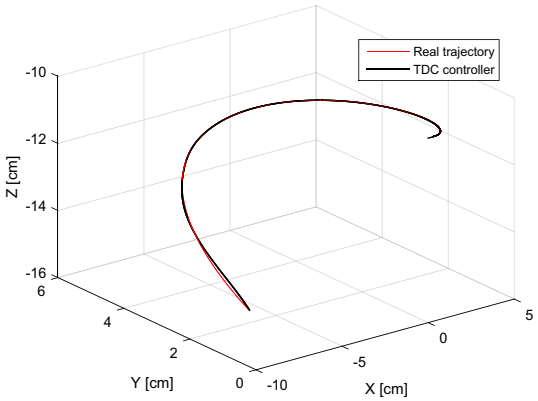

Fig. 5 Tracking trajectory in task space

3.2.2 Case 2 (Faulty case)

In order to evaluate the performances of the controller (17) in faulty environment, we have introduced sensor faults as follows: Δq=0.01° in joints 1 and 2 at t = 5 s. The results of the simulations are illustrated in Figs. 8, 9 and 10.

It is observed that the controller given by (17) lost its performance just at the moment of the introduction of the sensor faults. Indeed, we can see that there is an important error between the set trajectory and the desired trajectory. Moreover, Fig. 10 shows that a large magnitude peak has occurred in the control torque signal of joint 1.

3.2.3 Case 3 (faulty case under proposed scheme)

In order to handle the issues produced in the previous case and to overcome the drawbacks of the controller (17), this part investigates our switched time delay control based on neural networks. The proposed scheme uses the controller (17) in the scenario where there are small or no sensor fault and the controller (18) is used in the case of detecting important errors due to a fault.

The simulation results for the proposed controller are demonstrated in Figs. 11, 12 and 13. We can clearly see that at the time of the introduction of the fault (t = 5 s), the robot's trajectory converges (Fig. 11), to the desired trajectory after a certain time, which confirms the efficiency of the proposed controller. Indeed, after introducing a fault, the controller switches from control law (17) to controller law (18) which is designed with a large gain. This gain is able to compensate important sensor faults compared to small gain, which improves the tracking trajectory of the robot under a faulty environment. However, chatters occur due to the nature of the discontinuity of the sign function. To overcome this issue saturation function may

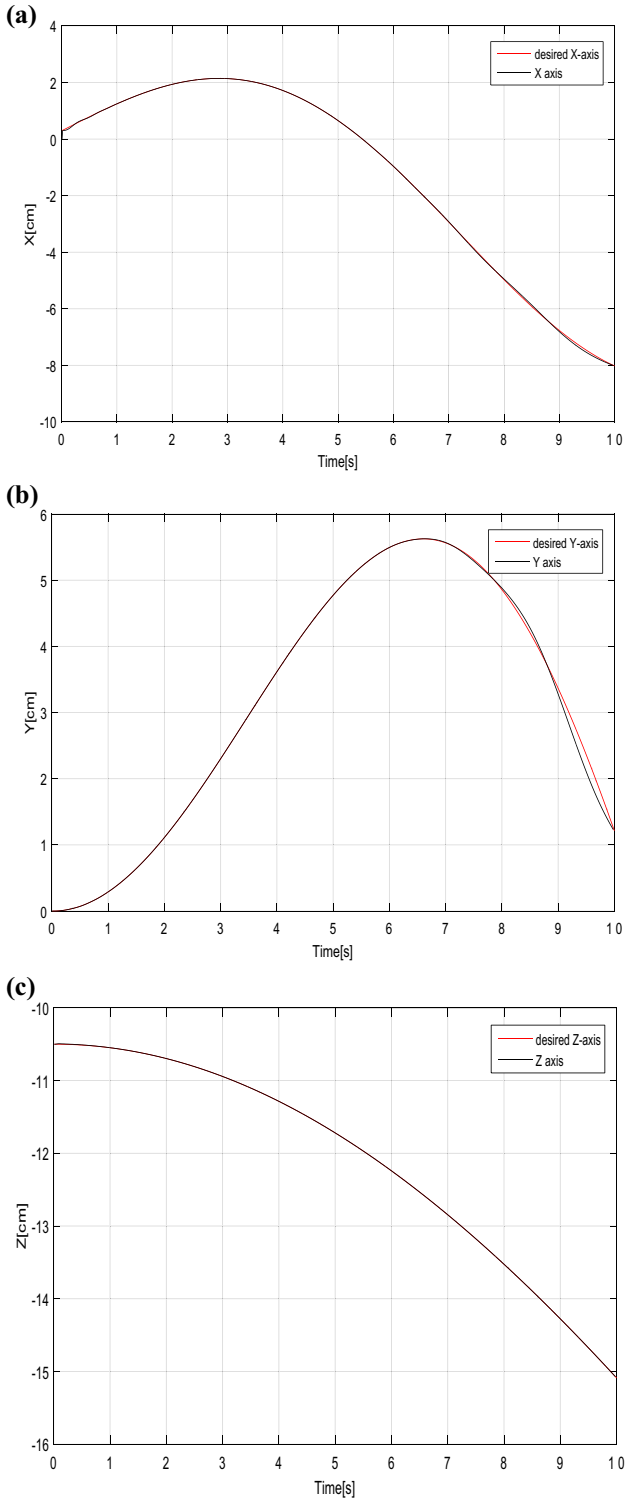

Fig. 6 **a** Tracking trajectory for x-axis, **b** tracking trajectory for y-axis, **c** tracking trajectory for z-axis

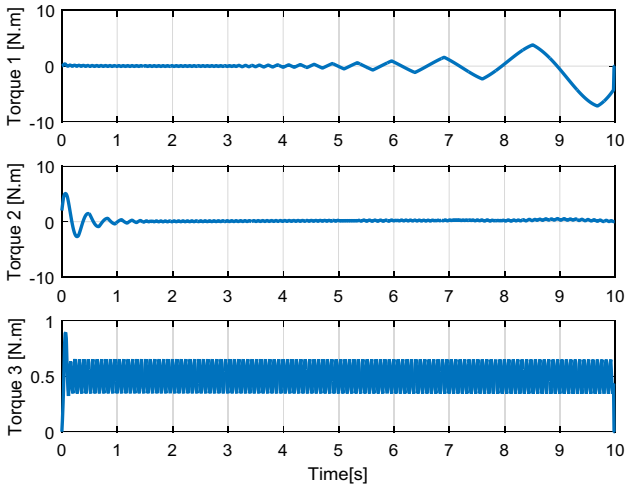

Fig. 7 Control torque

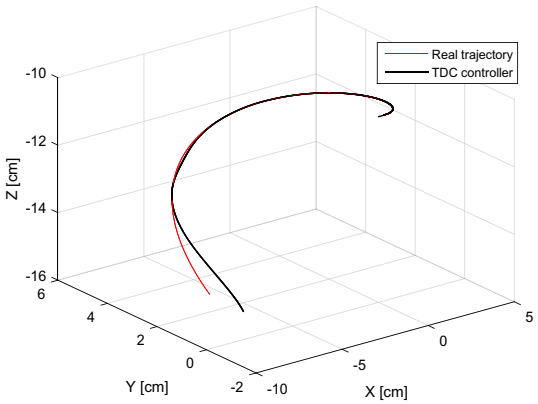

Fig. 8 Tracking trajectory in task space

be used instead of sign function. Figure 13 indicates that the proposed controller produces an acceptable control signal that stays in the admissible values.

3.3 Analysis of the results

The main goal of this paper is to develop a method that allows the detect, isolate and compensate of sensor fault of robot manipulators. So, we proposed an *MLP* neural network, which allows to learn the functioning of the system (manipulator) faultless, as it is shown in Figs. 2 and 3.

Then, we developed two controllers (17 and 18) which allow us to compensate this fault. Controller (17) is applied to the faultless system as it is shown in Figs. 5 and 6, where it is noted a divergence when the fault is introduced.

This result allowed us to think of developing another controller with a significant gain in order to compensate

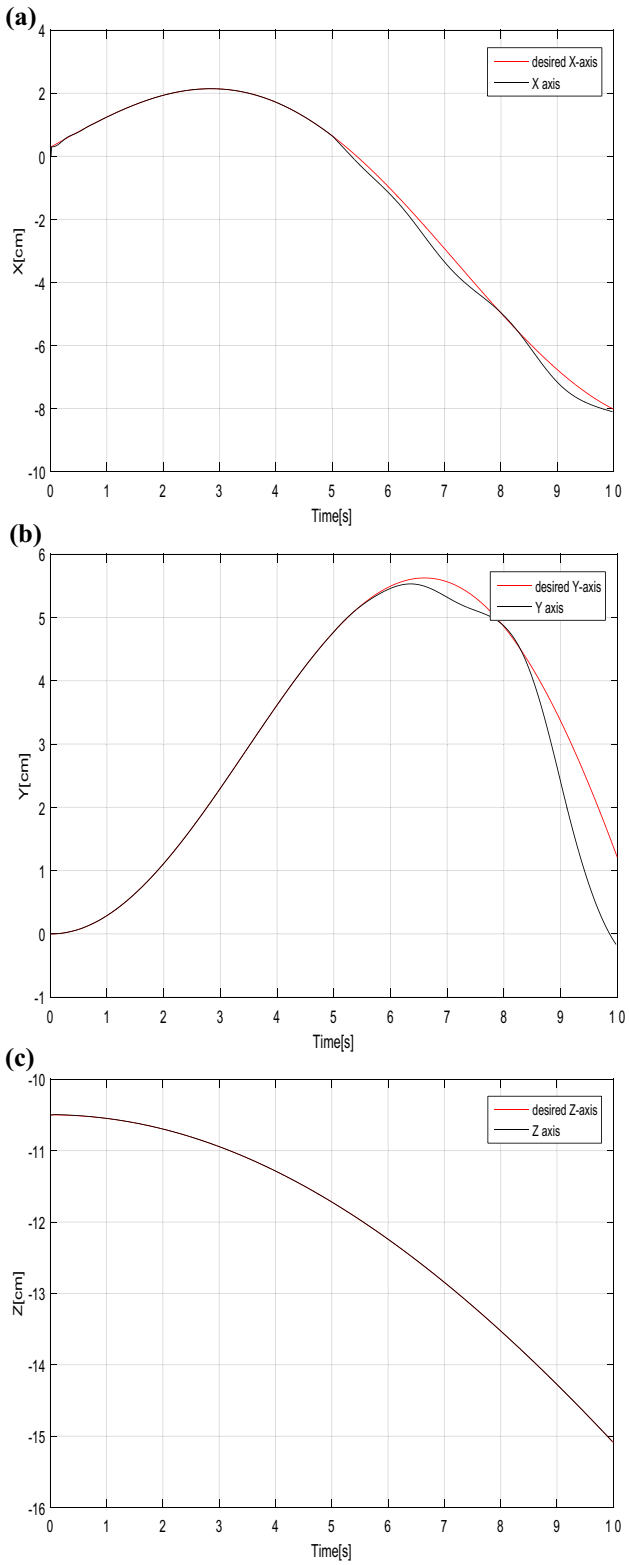

Fig. 9 **a** Tracking trajectory for x-axis, **b** tracking trajectory for y-axis, **c** tracking trajectory for z-axis

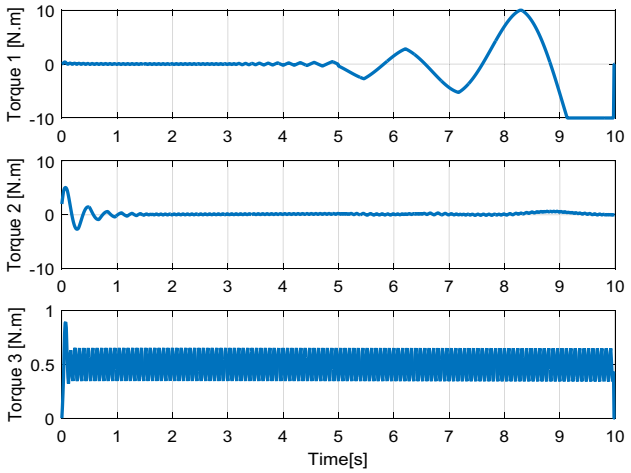

Fig. 10 Control torque

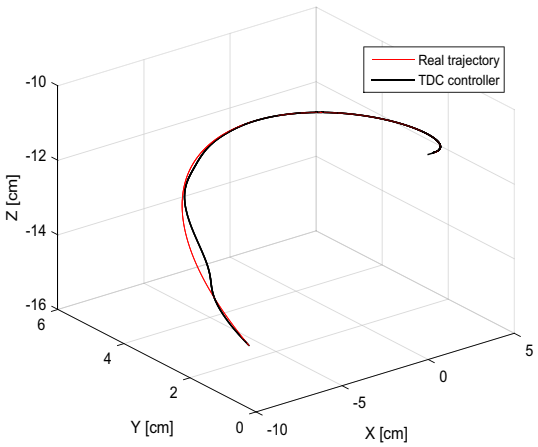

Fig. 11 Tracking trajectory in task space

for the fault. The controller 18 allows us to have a rapid compensation for the continuation of the trajectory even in the presence of a fault as it is shown in Fig. 11. The principle of this step is to switch between the controllers 17 and 18, in the presence or absence fault based on neural networks.

4 Conclusion

In this paper, a new concept of fault detection, isolation, and compensation based on neural networks and *TDC* has been developed and applied to a *SCARA* robot. The proposed controller composed of two independent schemes that switches between each other based on the *ANN* results. The first scheme is a *TDC* with a small gain, while the second one is a *TDC* with a high gain. This proposed control scheme has been applied to a *SCARA* robot where

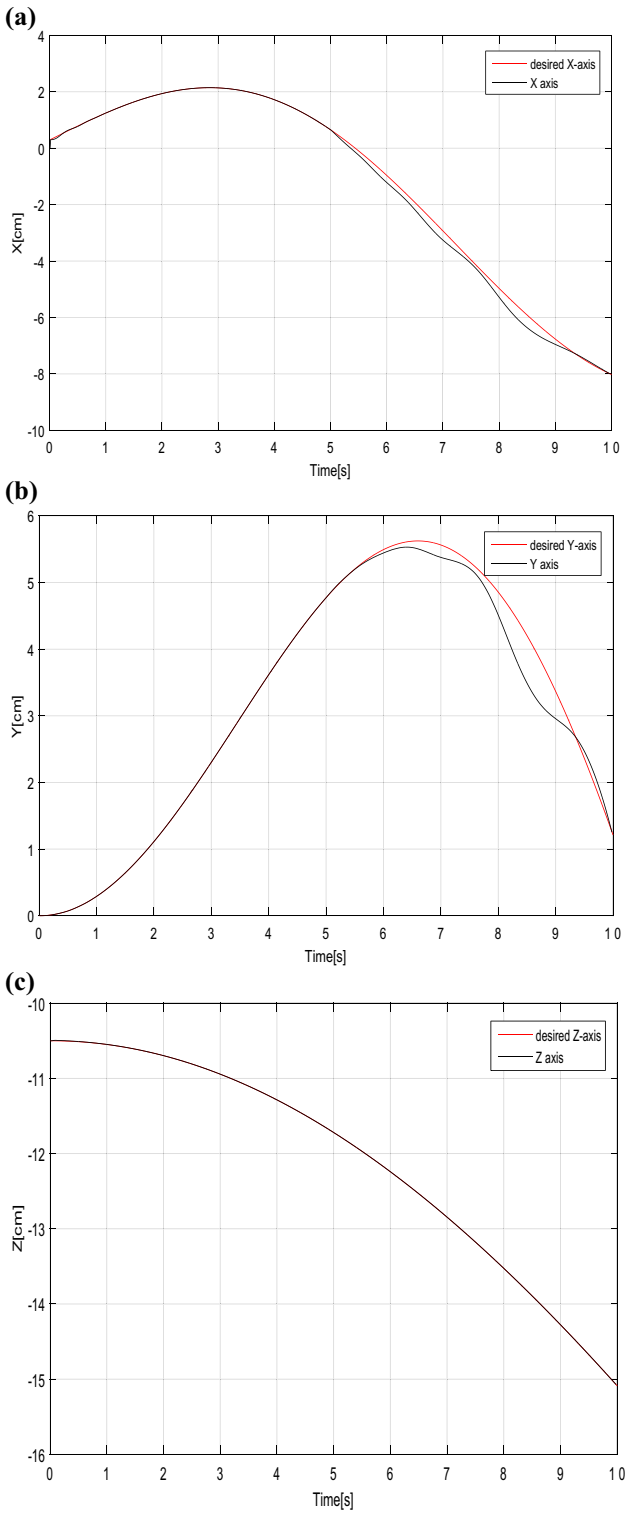

Fig. 12 **a** Tracking trajectory for x-axis, **b** tracking trajectory for y-axis, **c** tracking trajectory for z-axis

it switches from the *TDC* with a small gain to the *TDC* with a high gain when the *ANN* detects the occurrence of any sensor faults. The tracking error has been proven

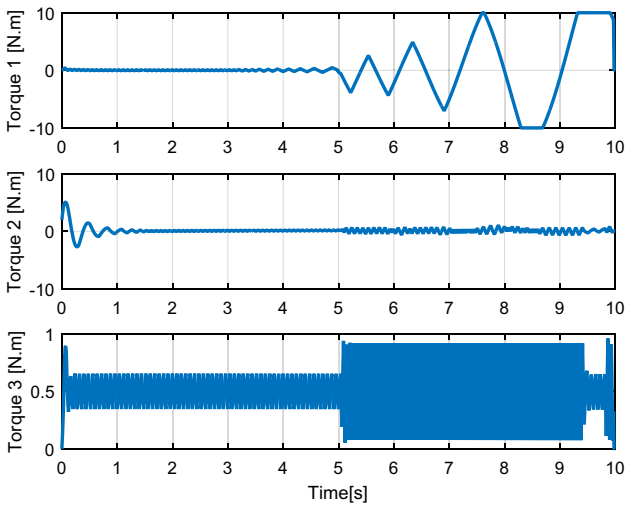

Fig. 13 Control torque

to converge to zero using Lyapunov analysis. Simulation results showed that the *ANN* was able to detect the sensors faults at every trial. Moreover, it was shown that the *TDC* with small gain exhibited poor tracking error in the face of faults, whereas despite the influence of the faults, the proposed switched *TDC* provided a good tracking error with an admissible control signal. The main advantage of the proposed controller resides in its free-model aspect where both the *ANN* and the *TDC* are independent from the model and use only the input and output of the function.

Our future works include fault detection and isolation based on adaptive time delay control where terminal nonsingular sliding surface might be used to obtain finite and fast convergence.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Boudjedir CE, Bouri M, and Boukhetala D (2020). Model-free iterative learning control with nonrepetitive trajectories for second-order mimo nonlinear systems- application to a delta robot. IEEE Trans Ind Electron, 1–1
2. Boudjedir CE, Boukhetala D (2021) Adaptive robust iterative learning control with application to a Delta robot. Proc Inst Mech Eng Part I J Syst Control Eng 235(2):207–221
3. Boudjedir CE, Boukhetala D, Bouri M (2019) Iterative learning control of multivariable uncertain nonlinear systems with non-repetitive trajectory. Nonlinear Dyn 95(3):2197–2208
4. Caccavale F, Cilibrizzi P, Pierri F, Villani L (2009) Actuators fault diagnosis for robot manipulators with uncertain model. Control Eng Pract 17(1):146–157
5. Van M, Franciosa P, Ceglarek D (2016) Fault diagnosis and fault-tolerant control of uncertain robot manipulators using high-order sliding mode. Math Probl Eng 2016:7926280
6. Qi W, Su H, Yang C, Ferrigno G, De Momi E, Aliverti A (2019) A fast and robust deep convolutional neural networks for complex human activity recognition using smartphone. Sensors 19:3731
7. Jadhav SD, Channe H (2016) Comparative study of k-NN, Naive Bayes and decision tree classification techniques. Int J Sci Res 5(1):1842–1845
8. Su H, Qi W, Yang C, Sandoval J, Ferrigno G, De Momi E (2020) Deep neural network approach in robot tool dynamics identification for bilateral teleoperation. IEEE Robot Autom Lett 5(2):2943–2949
9. Su H, Hu Y, Karimi HR, Knoll A, Ferrigno G, De Momi E (2020) Improved recurrent neural network-based manipulator control with remote center of motion constraints: Experimental results. Neural Netw 131:291–299
10. Ronao CA, Cho S-B (2016) Human activity recognition with smartphone sensors using deep learning neural networks. Expert Syst Appl 59:235–244
11. Cho SJ, Jin M, Kuc TY, Lee JS (2014) Stability guaranteed auto-tuning algorithm of a time-delay controller using a modified Nussbaum function. Int J Control 87(9):1926–1935
12. Zare M, Pourghasemi HR, Vafakhah M, Pradhan B (2013) Landslide susceptibility mapping at Vaz Watershed (Iran) using an artificial neural network model: a comparison between multilayer perceptron (MLP) and radial basic function (RBF) algorithms. Arab J Geosci 6(8):2873–2888
13. Nikdel N, Nikdel P, Badamchizadeh MA, Hassanzadeh I (2013) Using neural network model predictive control for controlling shape memory alloy-based manipulator. IEEE Trans Industr Electron 61(3):1394–1401
14. Boudjedir CE, Boukhetala D, Bouri M (2018) Nonlinear PD plus sliding mode control with application to a parallel delta robot. J Electr Eng 69(5):329–336
15. Baek J, Jin M, Han S (2016) A new adaptive sliding-mode control scheme for application to robot manipulators. IEEE Trans Industr Electron 63(6):3628–3637
16. Su H, Yang C, Ferrigno G, De Momi E (2019) Improved human–robot collaborative control of redundant robot for teleoperated minimally invasive surgery. IEEE Robot Autom Lett 4(2):1447–1453
17. Galicki M (2015) Finite-time control of robotic manipulators. Automatica 51:49–54
18. Youcef-Toumi, K., and Ito, O. (1990). A time delay controller for systems with unknown dynamics.
19. Cho SJ, Jin M, Kuc TY, Lee JS (2014) Control and synchronization of chaos systems using time-delay estimation and supervising switching control. Nonlinear Dyn 75(3):549–560

20. Tong S, Li YF (2013) Adaptive fuzzy output feedback control of MIMO nonlinear systems with unknown dead-zone inputs. IEEE Trans Fuzzy Syst 21(1):134–146
21. Zhang X, Wang H, Tian Y, Peyrodie L, Wang X (2018) Model-free based neural network control with time-delay estimation for lower extremity exoskeleton. Neurocomputing 272:178–188
22. Sun J, Han G, Zeng Z, Wang Y (2019) Memristor-based neural network circuit of full-function pavlov associative memory with time delay and variable learning rate. IEEE Trans Cybern 50:2935–2945
23. Jin M, Lee J, Tsagarakis NG (2016) Model-free robust adaptive control of humanoid robots with flexible joints. IEEE Trans Ind Electron 64(2):1706–1715
24. Sciavicco L, Siciliano B (2012) Modelling and Control of Robot Manipulators. Springer, New York, NY, USA
25. Spong MW, Hutchinson S, Vidyasagar M (2005) Robot Modeling and Control. Wiley, Hoboken, NJ, USA
26. Hsia TC, Gao LS (1990) Robot manipulator control using decentralized linear time-invariant time-delayed joint controllers. In: Proceedings, IEEE international conference on robotics and automation, 2070–2075.
27. Baek J, Cho S, Han S (2017) Practical time-delay control with adaptive gains for trajectory tracking of robot manipulators. IEEE Trans Industr Electron 65(7):5682–5692
28. Utkin V, Guldner J, Shi J (2017) Sliding mode control in electromechanical systems. CRC Press, Boca Raton
29. Jin M, Lee J, Chang PH, Choi C (2009) Practical nonsingular terminal sliding-mode control of robot manipulators for high-accuracy tracking control. IEEE Trans Ind Electron 56(9):3593–3601