INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. 2014; **28**:386–411
Published online 18 December 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/acs.2459

Adaptive neuro-predictive control for redundant robot manipulators in presence of static and dynamic obstacles: A Lyapunov-based approach

Ashkan M. Jasour and Mohammad Farrokhi*,†

Department of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran

SUMMARY

This paper presents a novel approach for online path tracking and obstacle avoidance of redundant robot manipulators. To this end, a nonlinear model predictive control (NMPC) method is designed that can track a desired path or reaches a moving target in the Cartesian space while avoiding static or moving obstacles as well as singular configurations in the workspace of the robot. The finite cost function of the NMPC is optimized at every sampling time, yielding an online optimal approach. In order to avoid collisions with moving obstacles and, at the same time, capturing a moving target, the future positions of the obstacles and the moving target are predicted using artificial neural networks (ANNs). Moreover, ANNs are employed to find a proper nonlinear model for the NMPC. The adaptation laws for the ANNs are obtained using the Lyapunov's direct method. The advantages of the proposed method are fourfold: (i) an adaptive and optimal approach is obtained, which can cope with changes in the system parameters; (ii) no inverse kinematics of the redundant manipulators is required; (iii) no prior knowledge about the obstacles and motion of the moving object is required; and (iv) stability of the closed-loop system is guaranteed. Numerical simulations, performed on a four degree-of-freedom redundant spatial manipulator actuated by DC servomotors, show effectiveness of the proposed method. Copyright © 2013 John Wiley & Sons, Ltd.

Received 11 November 2011; Revised 14 November 2013; Accepted 21 November 2013

KEY WORDS: redundant robot manipulator; nonlinear model predictive control; obstacle avoidance; adaptive neuro-control; Lyapunov's direct method

1. INTRODUCTION

Robot manipulators are increasingly used in many tasks such as industry, medicine, and space. One of the main reasons for using manipulators is to replace human in doing lengthy and repetitive operations or even dangerous tasks such as handling explosives or working in unhealthy environments. In the near future, robots will be able help human in doing daily works [1]. Therefore, efficient control strategies for robot manipulators play key roles in having them to behave like human.

Control strategies for robot manipulators can be classified into two categories: (i) joint coordinates and (ii) Cartesian coordinates [1, 2]. While in the joint-coordinate methods, the desired joint positions are specified, in the Cartesian-coordinate schemes, the desired position of the end effector is specified in the Cartesian space. The Cartesian-coordinate methods are considered as one of the challenging problems encountered in the robotics field. In this approach, manipulators are employed to track a predefined path in the Cartesian space or to reach a desired position in such a way that no collision with the obstacles in the workspace occurs. Therefore, feasible paths for joints of the robot

*Correspondence to: Mohammad Farrokhi, Department of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran.

†E-mail: farrokhi@iust.ac.ir

must be determined in such a way that the required criteria are met. Because of the physical limitations of manipulators, not all the criteria can be satisfied simultaneously. Hence, providing more degrees of freedom (DoF) to the robot is necessary. However, high DoFs in redundant manipulators lead to infinite number of possible joint positions for the same pose of the end effector. In other words, there will be infinite number of solutions to the problem of inverse kinematics of redundant manipulators, among which the collision-free and singular-free situations must be selected. The focus of this paper is to control redundant manipulators in the Cartesian space while obstacles are present in the workspace.

The control of a manipulator is performed in the joint space, while the goal is usually defined in the Cartesian space. In most control methods presented in literature, there is need for solving the inverse kinematic problem, which is finding the joint angles of the robot in terms of the position and orientation of the goal in the Cartesian space. Solving the inverse kinematic problem of redundant manipulators is a challenging task, and in general, there is no closed-form solution for it and, hence, must be solved by iterative methods. Different methods are proposed in literature for redundancy resolution; among them are task-priority redundancy resolution [3, 4], gradient projection, generalized inverse Jacobin, and extended Jacobin approaches [5, 6]. These methods are based on pseudoinverse of the Jacobian matrix. They are very time consuming and usually are performed offline. Hence, they may not be suitable for real-time applications [7]. On the other hand, the proposed method in this paper does not require solving the redundancy resolution of the inverse kinematic problem because the method works in the Cartesian space. This is one of the main advantages of the proposed method for using the model predictive control (MPC) strategy.

In some applications, such as interactions with human, robot manipulators must perform in dynamic environments, where moving objects exist in the workspace. Moreover, sometimes robots must grasp moving targets. Some papers have addressed the control of robot manipulators in dynamic environments [8, 9]. Dynamic collision checking for single and multiple articulated robots in complex environments is proposed in [10], where the path segments are tested in configuration space by adjusting the sampling resolution and comparing the lower bounds on distance between the objects in relative motion with the upper bounds on the length of curves traced by the points on the moving objects. This manuscript presents a solution for online optimal control of redundant robot manipulators in both stationary and nonstationary environments. Hence, the proposed scheme yields a more practical method as compared with the previously reported methods in well-established literature. To predict the motion of the moving obstacles and/or targets in the workspace of robot, NNs are employed.

In optimization-based approaches, which are implemented in the workspace as well as the joint space, a cost function is defined on the basis of the distance between the robot and the desired path as well as the obstacle [11–15]. This cost function is minimized (from the start time until the final time) using suitable optimizers such as numerical algorithms [12, 14], Pontryagin's principle-based algorithms, and intelligent algorithms (e.g., genetic algorithm and particle swarm optimization) [16–19]. However, most of these methods are not suitable for real-time applications. Constraint least-square problem subject to time-dependent states and control constraints has been proposed in [20], where the states constraints are used to describe the obstacle constraints as well the workspace constraints of the robot. The control constraints are used to describe the velocity constraints of the robot. In this manuscript, the nonlinear constraint optimization problem is considered in the Cartesian space of the robot and is solved in the optimizer part of the MPC in real time. The optimization problem is solved using the sequential quadratic programming (SQP) method, which will be explained in Section 3.

Conventional control methods (such as computed torque, inverse dynamic, and passivity based) have been proposed in literature to track the desired paths [1, 21–23]. One of the main concerns at this stage is the uncertainties in the robot parameters, which demand robust [1, 21, 24–26] and adaptive [1, 3, 27–29] approaches.

The MPC method is an optimal control strategy, which uses a finite cost function that is optimized at every sampling time under imposed constraints. In [30], authors have used self-organizing radial basis function NN to provide a predictive model for the nonlinear MPC (NMPC). The optimization problem is solved using a fast gradient method. Stability of the closed-loop system is shown using

Lyapunov method. Yan and Wang have used extreme learning machine with supervised learning to model uncertainties of constrained discrete-time nonlinear systems [31]. The minmax optimization problem is iteratively solved by a two-layer recurrent NN. However, no stability analysis of the closed-loop system or convergence of the parameters is reported. Ławryńczuk has used multilayer perceptron (MLP) to form a nonlinear predictive model for the NMPC [32]. A simple gradient of the performance index gives online adaptation method to the model. The method is applied to practical examples, and no stability analysis is provided. Salahshoor *et al.* [33] have proposed an adaptive growing and pruning radial basis function NN as a predictor model for NMPC to recursively capture the essential dynamics of casing-heading instability, which is caused by dynamic interaction between injection gas and multiphase fluid in gas-lift oil wells. Training of the NN is performed using extended and unscented Kalman filters. In [34], authors have proposed a linear predictive model plus a diagonal recurrent NN to compensate the nonlinear modelling error. In this way, under certain conditions, the linear MPC is extended to nonlinear process without any need to solve nonlinear optimization problem.

Although MPC is not a new control method, works related to robot manipulators is limited. Related works are about joint-space control methods, where the MPC is used as a controller for tracking a predefined trajectory for every joint without considering the redundancy resolution and the obstacle and singularity avoidance. The linear MPC is used in [35–37], whereas the NMPC has been employed in [37–39] for the joint-space control of manipulators. In this manuscript, using the NMPC, two stages (i.e., trajectory planning and controlling) are combined into a single stage. The input voltage of joint servomotors are determined in such a way that the end effector of the redundant manipulator tracks a predefined trajectory in the Cartesian space and, at the same time, reaches a desired static or moving target while avoiding collisions with the static and/or moving obstacles and singular configurations in the workspace of the robot. For the model prediction in the NMPC, artificial neural networks (ANNs) are used mainly because of their attractive advantage that no prior knowledge about the system parameters is needed. Moreover, using the online training algorithm, robustness against changes in the robot parameters is obtained. To guarantee stability of the closed-loop system, the weights of the NNs are adapted using the Lyapunov's direct method. In contrast to the previous approaches, which only guarantee local stability of the NMPC, the proposed method provides global stability.

The advantages of the proposed method can be summarized as follows. The MPC is an optimal approach because a cost function with some desired constraints is defined in order to fulfill the control goals. Moreover, the proposed method can be applied online because the defined cost function is finite. In addition, it works well for dynamic environments because the cost function is optimized at every sampling time; hence, the method can cope with changes in the environment of the robot. Furthermore, the proposed method can handle nonlinear dynamics of the robot as well as nonlinear constraints in the optimization process. The modeling is performed online, yielding an adaptive control strategy that can deal with changes in the system parameters. Moreover, there is no need for solving the inverse kinematics and redundancy resolution of redundant manipulators because the method works in the Cartesian space of the robot. In addition, dynamics of DC servomotors are also considered in the nonlinear dynamics of the robot.

This paper is organized as follows. Section 2 describes the problem considered in this paper. Section 3 briefly explains the NMPC method. Section 4 presents the proposed control method. In Section 5, stability of the proposed control method using the Lyapunov's direct method is developed. Simulation results are presented in Section 6. Conclusions are drawn in Section 7.

2. PROBLEM SETTING

Consider a 4-DoF spatial redundant manipulator shown in Figure 1. The position of the end effector in Cartesian space can be calculated in terms of the joint positions as [1,2]

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c_1(l_4c_{234} + l_3c_{23} + l_2c_2) \\ s_1(l_4c_{234} + l_3c_{23} + l_2c_2) \\ l_4s_{234} + l_3s_{23} + l_2s_2 \end{bmatrix} \tag{1}$$

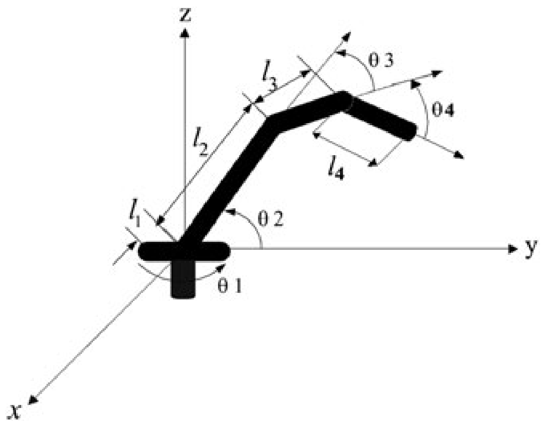

Figure 1. Schematic diagram of a 4-DoF spatial manipulator.

where l_i is the length of the ith link of the robot and $c_{ijk} := \cos(\theta_i + \theta_j + \theta_k)$ and $s_{ijk} := \sin(\theta_i + \theta_j + \theta_k)$, in which θ_i is the angle of the ith joint.

The desired coordinates for the position of the end effector at each sampling time (k) are predefined in Cartesian space as $(x_d(k), y_d(k), z_d(k))$. Moreover, obstacles are considered in the workspace of the robot. It is well known that extra DoFs in redundant manipulators give infinite number of possible joint positions for the desired position and/or orientation of the end effector.

The control objective is to obtain the control law for every joint of a redundant manipulator such that the end effector tracks a predefined path and, at the same time, reaches a desired static or moving target in Cartesian space while avoiding collision with static and/or moving obstacles and singular configurations in the workspace of the robot. Moreover, the applied voltages to DC servomotors of each joint must satisfy the constraints of these servomotors as well as physical constraints of the robot. In addition, the robustness of the controller against changes in robot parameters and stability of the entire closed-loop system must be satisfied.

3. MODEL PREDICTIVE CONTROL

Unlike classical control methods, where the control actions are calculated on the basis of the past output(s) of the system or state variables, the MPC is a model-based optimal controller, which uses predictions of the system outputs to calculate the control law [40,41]. On the basis of measurements obtained at the sampling time k, the controller predicts the output of the system over the prediction horizon N_P in the future instances using the system model and determines the input over the control horizon $N_C \geqslant N_P$ such that a predefined cost function is minimized.

From the theoretical point of view, the MPC algorithm can be expressed as

$$u = \arg\min_{u}(J(k)) \tag{2}$$

such that the following constraints are satisfied:

$$\begin{aligned}
\mathbf{x}(k|k) &= \mathbf{x}_0 \\
\mathbf{u}(k+j+1|k) &= \mathbf{u}\,(k+N_C|k) \quad j+1 \geqslant N_C \\
\mathbf{x}(k+j+1|k) &= \mathbf{f}_d\,(\mathbf{x}(k+j|k), \mathbf{u}(k+j|k)) \\
\mathbf{y}(k+j+1|k) &= \mathbf{h}_d\,(\mathbf{x}(k+j+1|k)) \\
\mathbf{x}_{\min} \leqslant \mathbf{x}&(k+j+1|k) \leqslant \mathbf{x}_{\max} \\
\mathbf{u}_{\min} \leqslant \mathbf{u}&(k+j+1|k) \leqslant \mathbf{u}_{\max}
\end{aligned} \tag{3}$$

where $j \in [0,\ N_P - 1]$, $\mathbf{x} \in R^n$ and $\mathbf{u} \in R^m$ are the states and inputs to the system, respectively, $\mathbf{x_0}$ is the initial condition, and $\mathbf{f}_d$ and $\mathbf{h}_d$ are functions of the system used for prediction. The notation $a(m|n)$ indicates the value of a at instant m predicted at instant n. Moreover, intervals $[\mathbf{x}_{\min}\ \mathbf{x}_{\max}]$

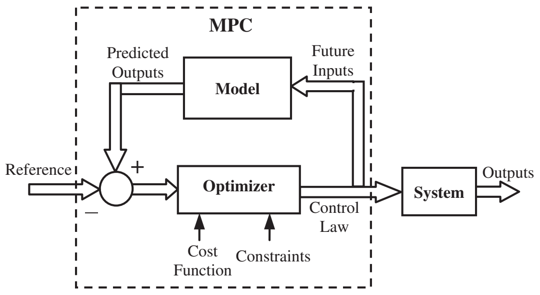

Figure 2. Block diagram of MPC.

and [$\mathbf{u}_{\min}$ $\mathbf{u}_{\max}$] stand for the lower and upper bounds of the states and inputs, respectively. The cost function J is defined in terms of the predicted and desired outputs of the system over the prediction horizon. The MPC schemes that are based on the nonlinear model of the system and/or consider a nonquadratic cost function and nonlinear constraints on the inputs and states are called NMPC [22].

The optimization problems (2) and (3) must be solved at every sampling time (k), which yields a sequence of optimal control law as $\{\mathbf{u}^*(k+1|k), \ldots\ \mathbf{u}^*(k+N_c|k)\}$. The first element of this sequence is applied to the system, and the rest is discarded. Using the new measurements at the next sampling time ($k+1$), the entire procedure of prediction and optimization is repeated (Figure 2).

To solve the constraint nonlinear optimization problem in (2) and (3), the SQP method has been employed in this paper. Appendix A gives a brief explanation of this method. Most books on optimization programming explain this method in details (e.g., [42]).

The schematic diagram of the NMPC is shown in Figure 2. According to this figure, the optimizer and the model used for prediction of the system's future behavior construct the main parts of the NMPC. Therefore, the design process of the NMPC includes obtaining a proper prediction model and a suitable cost function and constraints, satisfying the control objectives. These will be explained in the next section.

4. CONTROL OF REDUNDANT MANIPULATORS USING NMPC

The purpose of the control method is to obtain a control law such that the end effector tracks a predefined path or reaches a desired position in the Cartesian space and, at the same time, avoids collision with the obstacles and singular configurations in the workspace. To achieve these goals, the NMPC method is implemented in this paper. The block diagram of the closed-loop control system is shown in Figure 3. In order to obtain the control law, an appropriate cost function, a set of constraints, and a prediction model for the robot and the environment should be defined for the NMPC algorithm.

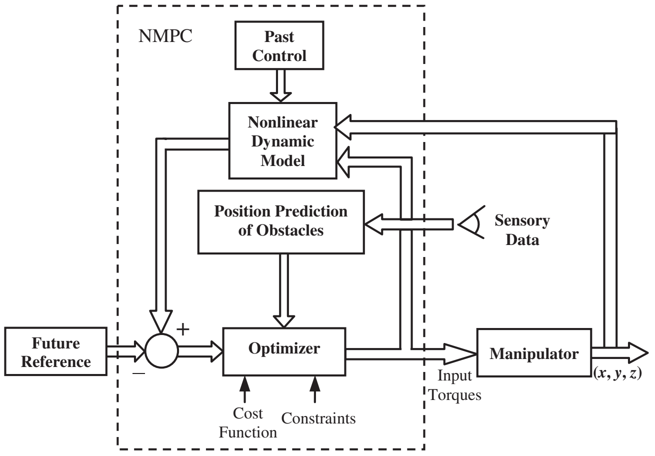

Figure 3. Block diagram of closed-loop system.

4.1. Cost function and constraints

In order to enable the end effector to track a predefined path or to reach a desired position in the Cartesian space, the cost function must have direct relation with the distance between the end effector and the desired coordinates. On the other hand, for the obstacle avoidance, the cost function must have inverse relation with the distance between the obstacle and the manipulator links. With this, the cost function is defined in this paper as follows:

$$J = \sum_{j=1}^{N_P} Q\left(\frac{e^{D_P(k+j|k)} - e^{D_{P\min}}}{e^{D_{P\max}} - e^{D_{P\min}}}\right) + R\left(\frac{e^{-D_O(k+j|k)} - e^{-D_{O\max}}}{e^{-D_{O\min}} - e^{-D_{O\max}}}\right) \tag{4}$$

where N_P is the prediction horizon, D_P is the Euclidean distance between the end effector and the desired coordinate, D_O is the minimum Euclidean distance between the manipulator and the obstacle, and $Q \geqslant 0$ and $R \geqslant 0$ are the weighting parameters. Moreover, intervals $[D_{P\min}\ D_{P\max}]$ and $[D_{O\min}\ D_{O\max}]$ are the range of variations for D_P and D_O, respectively. It is assumed that D_P and D_O are available here (i.e., they are measured by some means and measuring techniques). The two terms in (4), one for the path tracking and the other for the obstacle avoidance, are normalized to [0 1]. By minimizing the cost function in (4), the distance between the desired and the actual position of the end effector is minimized while the distance between the robot and the obstacle is maximized.

Next, constraints in the optimization problem is defined. In view of the fact that the amplitude of the input voltages applied to the DC servomotors is limited, one of the constraints is

$$V_{t\min} \leqslant V_t \leqslant V_{t\max} \tag{5}$$

where $V_{t\min}$ and $V_{t\max}$ stand for the lower and upper bounds of input voltages, respectively.

Next, in a singular configuration, theoretically the joint velocities become infinite. Therefore, the following constraint must be taken into account to avoid singularities:

$$\dot{\theta}_{\min} \leqslant \dot{\theta} \leqslant \dot{\theta}_{\max} \tag{6}$$

where $\dot{\theta}_{\min}$ and $\dot{\theta}_{\max}$ are the lower and upper bounds of joints velocities, respectively.

4.2. Robot manipulator prediction model

The dynamic equations of the robot manipulators with series links can be obtained using the Lagrangian dynamic formulation as [1,2]

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \mathbf{D}(\dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) = \boldsymbol{\tau} \tag{7}$$

where $\boldsymbol{\theta},\ \dot{\boldsymbol{\theta}} \in R^n$ are the angular position and velocity of joints, respectively, $\mathbf{M}(\boldsymbol{\theta}) \in R^{n\times n}$ is the symmetric and positive-definite inertia matrix, $\mathbf{C}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) \in R^n$ is the vector of centrifugal and Coriolis terms, $\mathbf{G}(\boldsymbol{\theta}) \in R^n$ is the vector of gravity terms, $\mathbf{D}(\dot{\boldsymbol{\theta}}) \in R^n$ is the vector of joints friction terms, $\tau \in R^n$ is the torque vector, and n is the DoF, which is equivalent to four for the robot considered in this paper. Details of the aforementioned matrix and vectors for a 4-DoF spatial manipulator with revolute joints (Figure 1) are given in Appendix B.

Friction for joint i is defined as [10]

$$D(i) = D_v\dot{\theta}_i + D_d\,\mathrm{sgn}(\dot{\theta}_i),\ \ i = 1,\ \ldots,\ n \tag{8}$$

where D_v and D_d are the coefficients of the viscous and dynamic frictions, respectively. The dynamics of the armature-controlled DC servomotors that drive the links can be expressed as [1,2]

$$\begin{aligned}\boldsymbol{\tau}_e &= \mathbf{K}_T\,\mathbf{i}_a\\ \boldsymbol{\tau}_e &= \mathbf{J}_m\ddot{\boldsymbol{\theta}}_m + \mathbf{B}_m\dot{\boldsymbol{\theta}}_m + \boldsymbol{\tau}_m\\ \mathbf{V}_t &= \mathbf{R}_a\mathbf{i}_a + \mathbf{L}_a\dot{\mathbf{i}}_a + \mathbf{K}_E\dot{\boldsymbol{\theta}}_m\end{aligned} \tag{9}$$

where $\boldsymbol{\tau}_e \in R^n$ is the vector of electromagnetic torques, $\mathbf{K}_T \in R^{n\times n}$ is the diagonal matrix of motor torque constants, $\mathbf{i}_a \in R^n$ is the vector of armature currents, $\mathbf{J}_m \in R^{n\times n}$ is the diagonal matrix of moment of inertia, $\mathbf{B}_m \in R^{n\times n}$ is the diagonal matrix of torsional damping coefficients, $\boldsymbol{\theta}_m, \dot{\boldsymbol{\theta}}_m, \ddot{\boldsymbol{\theta}}_m \in R^n$ denote vectors of the motor shaft positions, velocities, and accelerations, respectively, $\boldsymbol{\tau}_m \in R^n$ is the vector of load torques, $\mathbf{V}_t \in R^n$ is the vector of armature input voltages, $\mathbf{R}_a \in R^{n\times n}$ is the diagonal matrix of armature resistances, $\mathbf{L}_a \in R^{n\times n}$ is the diagonal matrix of armature inductances, and $\mathbf{K}_E \in R^{n\times n}$ is the diagonal matrix of the back electromotive force coefficients. Because the dynamic equations of DC servomotors are considered here, the relationship between the robot joint and the motor shaft can be represented as

$$\mathbf{R} = \operatorname{diag}\left(\frac{\theta_i}{\theta_{m_i}}\right) = \operatorname{diag}\left(\frac{\tau_{m_i}}{\tau_i}\right), \tag{10}$$

where $\mathbf{R} \in R^{n\times n}$ is a diagonal positive-definite matrix representing the gear ratios for n joints. Because armature inductances are small and negligible, (9) can be expressed as [1, 2]

$$\mathbf{J}_m\ddot{\boldsymbol{\theta}}_m + \left(\mathbf{B}_m + \frac{\mathbf{K}_E\mathbf{K}_T}{\mathbf{R}_a}\right)\dot{\boldsymbol{\theta}}_m + \boldsymbol{\tau}_m = \frac{\mathbf{K}_T}{\mathbf{R}_a}\mathbf{V}_t \tag{11}$$

Using (10) to eliminate $\boldsymbol{\theta}_m$ and $\boldsymbol{\tau}_m$ in (11) and then substituting for $\boldsymbol{\tau}$ using (7), the governing equation of an n-link robot manipulator including actuator dynamics can be obtained as

$$(\mathbf{J}_m + \mathbf{R}^2\mathbf{M})\ddot{\boldsymbol{\theta}} + \left(\mathbf{B}_m + \frac{\mathbf{K}_E\mathbf{K}_T}{\mathbf{R}_a}\right)\dot{\boldsymbol{\theta}} + \mathbf{R}^2(\mathbf{C}+\mathbf{G}+\mathbf{D}) = \frac{\mathbf{R}\mathbf{K}_T}{\mathbf{R}_a}\mathbf{V}_t \tag{12}$$

Equation (12) shows the governing nonlinear dynamic equation of the robot manipulator, where the armature input voltages are considered as the control inputs.

To obtain a prediction model for the NMPC, the nonlinear dynamic equation in (12) is used in this paper. Substituting $(\boldsymbol{\theta}(k+1) - \boldsymbol{\theta}(k))/T$ with $\dot{\boldsymbol{\theta}}(i = 1, \ldots, n)$ in (12), a one-step-ahead prediction for joint angles can be expressed as

$$\boldsymbol{\theta}(k+1) = \mathbf{f}_d\left(\boldsymbol{\theta}(k),\ \Delta\boldsymbol{\theta}(k),\ \mathbf{V}_t(k)\right) \tag{13}$$

where $\mathbf{f_d} \in R^{n\times 1}$ is a nonlinear function of the robot and servomotors parameters, k is the sampling time, and T is the sampling rate. Using forward kinematics in (1), a one-step-ahead prediction of the end-effector position can be obtained. However, in the predictive control methods, multistep predictions over the prediction horizon are needed. To this end, the one-step prediction is applied recursively to the model over the prediction horizon.

Because the prediction model of the robot in (13) is a function of the robot parameters, the exact values of these parameters are needed. Moreover, this model is valid only when no changes occur in the robot parameters. However, the robot parameters (e.g., frictions) change as the robot ages. To cope with these changes, an adaptive prediction model based on the online system identification is needed.

The ability of ANNs to approximate highly nonlinear functions and their adaptive nature make them suitable for adaptive identification of nonlinear systems. Hence, in this paper, ANNs are employed for modeling the nonlinear dynamics of the robot.

It can be shown that using an MLP, the prediction model in (13) can be represented as follows [43]:

$$\boldsymbol{\theta}(k+1) = \mathbf{W}^*(k)\boldsymbol{\varphi}\left(\boldsymbol{\theta}(k), \Delta\boldsymbol{\theta}(k), \mathbf{V_t}(k)\right) + \boldsymbol{\varepsilon} \tag{14}$$

where $\mathbf{W}^* \in R^{n\times s}$ is the ideal but unknown weight matrix that connects the neurons in the hidden layer to the output layer, $\boldsymbol{\varphi} \in R^{s\times 1}$ is the vector containing nonlinear (sigmoidal) functions in the neurons of the hidden layer, $\boldsymbol{\varepsilon}$ is the bounded approximation error, n is the number of outputs in the MLP, and s is the number of hidden neurons. Therefore, model (13) can be approximated as

$$\hat{\boldsymbol{\theta}}(k+1) = \mathbf{W}(k)\boldsymbol{\varphi}\left(\boldsymbol{\theta}(k), \Delta\boldsymbol{\theta}(k), \mathbf{V_t}(k)\right) \tag{15}$$

By updating the weights of the neural model in (15) using algorithms such as the well-known error back-propagation method, the weights error $(\mathbf{W}^*(k) - \mathbf{W}(k))$, and hence the identification

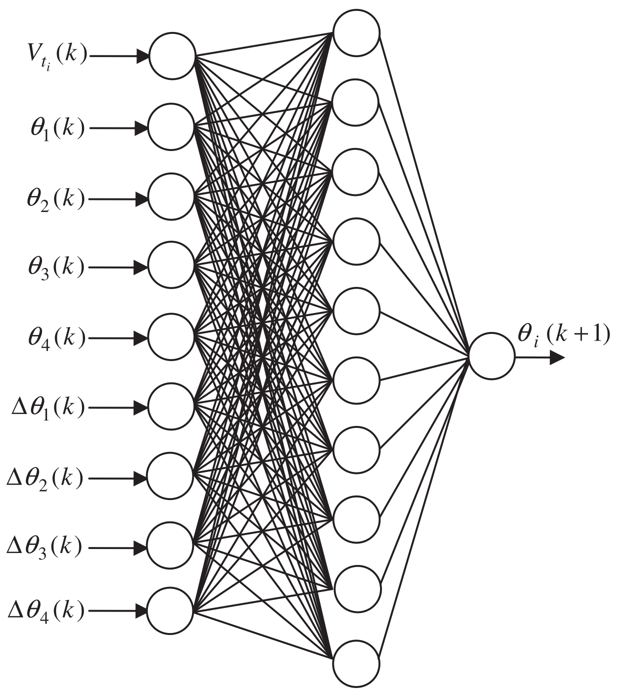

Figure 4. NN structure for model prediction.

error $(\boldsymbol{\theta}(k) - \hat{\boldsymbol{\theta}}(k))$, remains bounded [44]. According to (15), the neural prediction model is a function of the input voltages as well as the angular positions and velocities of joints at the current sampling time (k). The output of the predictor is the angular position of the joints at the next sampling time $(k + 1)$. Using this structure, a one-step-ahead prediction of the joints position can be obtained. In this case, the history of the outputs of the ANNs is considered as inputs for the next step. Then, by applying this prediction recursively, a multistep prediction can be achieved over the prediction horizon (N_P in (4)). In order to obtain better accuracies and faster convergence of the weights, one ANN is used for every joint angle of the robot. Therefore, in this case, the inputs to every ANN are the input voltage of the corresponding joint and the angular position and velocity of all joints and at the current sampling time. The output is the angular position of the corresponding joint at the next sampling time. In other words, the input vector applied to every ANN is $(\boldsymbol{\theta}(k)$, $\Delta\boldsymbol{\theta}(k)$, $V_{ti}(k))$, where for the case of 4-DoF, $\boldsymbol{\theta}(k) = [\theta_1(k) \quad \theta_2(k) \quad \theta_3(k) \quad \theta_4(k)]^T$ and the output is $\theta_i(k + 1)$.

The structure of the implemented NN is shown in Figure 4. The NNs are trained offline first using the data gathered from the model (13). The main reason for this is to have better initial values for the weights of NNs instead of random numbers. In this way, large initial tracking error and, hence, collision with the obstacles at the primary movements of the robot can be avoided. It should be mentioned that the offline training does not need to be very accurate; just a rough familiarity of NNs with the robot behavior suffices. Then, the online training of NNs can cope with all the changes in robot parameters, including the links masses, the joints frictions, and the parameters of servomotors.

4.3. Prediction model of robot environment

The environment of the robot contains stationary and/or moving obstacles, which should be avoided. In addition, there is a target, which should be touched or grabbed. In order to implement the NMPC to control a manipulator in this environment, the future positions of these objects must be predicted. For static objects, the data obtained using sensors at each sampling time can be used. For moving objects, however, a model is needed to predict their movements. The outputs of these prediction models are used in the cost function (4) over the prediction horizon (N_P).

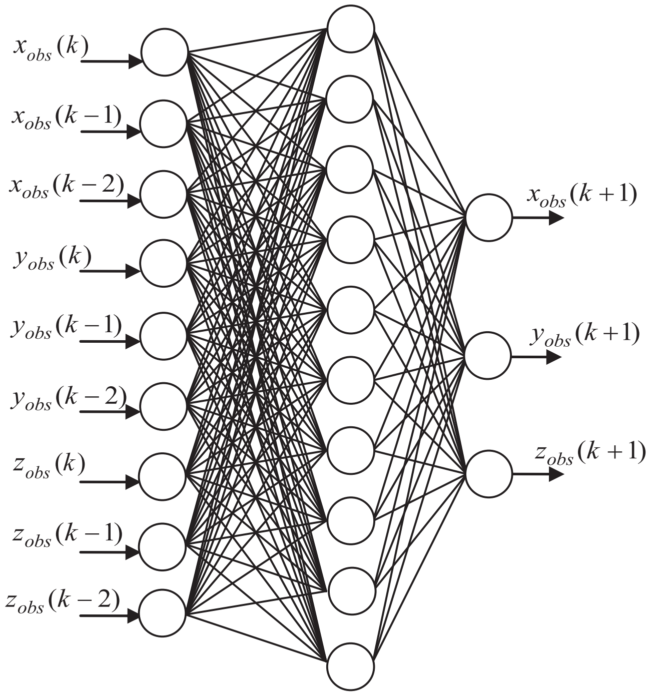

Figure 5. NN structure for prediction of obstacle position.

The states of a moving object (obstacle or target) can be represented as a function of the previous states as

$$\mathbf{s}(k+1) = \mathbf{f}\left(\mathbf{s}(k),\ \mathbf{s}(k-1),\ \mathbf{s}(k-2),\ \ldots\right) \tag{16}$$

In this paper, the nonlinear function in (16) is approximated using MLP NN. In particular, the prediction is based on the coordinate of the moving object at the current and two past sampling times (i.e., $[x(k)\ x(k-1)\ x(k-2)]$, $[y(k)\ y(k-1)\ y(k-2)]$, and $[z(k)\ z(k-1)\ z(k-2)]$). The output of the predictor is the coordinate of the moving object at the next sampling time (i.e., $[x(k+1)\ y(k+1)\ z(k+1)]$).

Figure 5 shows the structure of the proposed neuro-predictor in this paper. Training of the NN is performed online using the recursive least-square algorithm [45]. Using this algorithm at each sampling time k, summation of the matching errors for all the input–output pairs up to the kth sample is minimized. However, in this paper, summation of errors for the last N input–output pairs is considered for minimization. In this way, the training time can remain constant, and N can be selected in such a way that training time over every sampling time does not exceed the sampling rate. One of the main advantages of using NNs for prediction of moving objects is that by employing the online training scheme, no prior knowledge about the motion of the obstacle and/or target is needed.

5. STABILITY ANALYSIS

This section presents stability analysis of the proposed NMPC in the closed-loop system. Using an extension of the Lyapunov theory, first, a Lyapunov candidate is defined. Then, it is shown that the derivative of the defined Lyapunov function is negative.

Let the nonlinear system under control be defined by the following difference equation:

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{y}(k) &= \mathbf{h}(\mathbf{x}(k))\end{aligned} \tag{17}$$

where $\mathbf{x} \in \aleph \subset R^n$ is the state vector in which $\aleph$ is the compact set containing the origin; $\mathbf{u} \in U \subset R^m$ and $\mathbf{y} \in Y \subset R^p$ are the inputs and outputs of the system, respectively, in which

U is the compact set containing the origin; and mapping $\mathbf{f}: R^{m\times n} \to R^n$ is a continuous Lipschitz function with initial conditions $\mathbf{f}(0,0) = 0$.

On the basis of the state vector $\mathbf{x}$ and the input vector $\mathbf{u}$ at the sampling time k, the cost function for the NMPC is defined as [46]

$$V(\mathbf{x}, k, \mathbf{u}) = \sum_{i=k+1}^{k+N_P} \ell\left(\mathbf{x}^u(i; (\mathbf{x}, j)), \mathbf{u}(i)\right) + F(\mathbf{x}) \tag{18}$$

where ℓ is defined such that it satisfies the control objectives, $\ell(\mathbf{x}, \mathbf{u}) \geqslant 0$, and $\ell(0,0) = 0$; $\mathbf{x}^u(i; (\mathbf{x}, j))$ is the sate vector at the sampling time i resulting from an initial state $\mathbf{x}$ at the sampling time j; $\mathbf{u}(i)$ is the control sequence $F(\mathbf{x})$is a scalar function of states; and $N_P$is the prediction horizon.

Moreover, the neural prediction model for every state of the system in (17) is

$$\hat{x}_i(k) = \mathbf{w}_i(k)\ \boldsymbol{\varphi}\left(\mathbf{x}(k), \mathbf{u}(k)\right) \tag{19}$$

where $\hat{x}_i$ $(1 = 1, \ldots, n)$ is the ith state of the system, $\mathbf{w}_i \in R^{1\times s}$ is the ith row of the weight matrix $\mathbf{W}$ in (15), $\boldsymbol{\varphi} \in R^{s\times 1}$ is the transfer functions vector, and s is the number of hidden neurons.

The cost function (18) must be minimized at each sampling time (k) with the initial state $\mathbf{x}(0)$, yielding a sequence of optimal control law as

$$\mathbf{u}^*(\mathbf{x}, k) = \left\{\mathbf{u}^*\left(k+1; (\mathbf{x}, k)\right), \ldots, \mathbf{u}^*\left(k + N_C; (\mathbf{x}, k)\right)\right\} \tag{20}$$

where $N_C \leqslant N_P$ is the control horizon. The minimized cost function can be represented as

$$V^*(\mathbf{x}, k) = V\left(\mathbf{x}, k, \mathbf{u}^*(\mathbf{x}, k)\right) \tag{21}$$

And the state trajectory resulted form $\mathbf{u}^*(\mathbf{x}, k)$ is

$$\mathbf{x}^*(\mathbf{x}, k) = \{\mathbf{x}^*(k+1; \mathbf{x}),\ \mathbf{x}^*(k+2; \mathbf{x}),\ \ldots,\ \mathbf{x}^*(k+N_P; \mathbf{x})\} \tag{22}$$

Hence, the minimized cost function in (21) can be written as

$$V^*(\mathbf{x}, k) = \sum_{i=k+1}^{k+N_P} \ell\left(\mathbf{x}^*(i; \mathbf{x}), u^*(i; \mathbf{x})\right) + F(\mathbf{x}^*) \tag{23}$$

The first member of the obtained control law in (20) is applied to the system (17) until the next sampling time. Using the new measurements at the next sampling time, the entire procedure of the prediction and optimization procedure will be repeated.

Theorem 1

Consider the nonlinear system in (17), the nonlinear model predictive controller with the cost function in (18), and the neural prediction model in (19). If the adaptation law for the weights of neural models in (19) is defined as

$$\mathbf{w}_i(k) := \mathbf{w}_i(k-1) + \Psi_i(k)\Gamma_i(k) \tag{24}$$

where

$$\begin{aligned} \Gamma_i(k) &= x_i(k) - \mathbf{w}_i(k-1)\ \boldsymbol{\varphi}(k) \\ \Psi_i(k) &= \frac{\boldsymbol{\varphi}(k)}{\|\boldsymbol{\varphi}(k)\|^2}\left(1 - \lambda_i \frac{|e_i(k)|}{|\Gamma_i(k)|}\right) \\ e_i(k) &= x_i(k) - \hat{x}_i(k) \end{aligned} \tag{25}$$

in which e is the identification error and λ_i is a small positive constant, then the entire closed-loop system including the system under control and the model predictive controller is stable.

Proof
Consider the system in (17) and the cost function in (18). Function F in the cost function is defined as

$$F(\mathbf{x}) := \mathbf{E}(k)^T \mathbf{E}(k) \tag{26}$$

where $\mathbf{E}(k) = [e_1(k), \ \ldots, \ e_n(k)]^T$ is the vector of identification errors. On the basis of the cost function in (21), the Lyapunov function is defined as

$$L := V^*(\mathbf{x}, k) \tag{27}$$

The first difference of the Lyapunov function in (27) is

$$\Delta L = \Delta V^*(\mathbf{x}) = V^*(\mathbf{x}, k+1) - V^*(\mathbf{x}, k) \tag{28}$$

There is an upper bound for $V^*(\mathbf{x}, k+1)$ as

$$V^*(\mathbf{x}, k+1) \leqslant V^*(\mathbf{x}, k+1, \tilde{\mathbf{u}}) \tag{29}$$

where $V^*(\mathbf{x}, k+1)$ is the minimized cost function with the optimal control law $\mathbf{u}^*(\mathbf{x}, k+1)$ started from the initial sate $\mathbf{x}$ at the sampling time $k+1$ as

$$\begin{aligned} \mathbf{u}^*(\mathbf{x}, k+1) = \big\{&\mathbf{u}^*\left(k+1; (\mathbf{x}, k+1)\right), \ \mathbf{u}^*\left(k+2; (\mathbf{x}, k+1)\right), \ldots, \\ &\mathbf{u}^*\left(k+N_C; (\mathbf{x}, k+1)\right), \ \mathbf{u}^*(k+N_C+1; (\mathbf{x}, k+1))\big\} \end{aligned} \tag{30}$$

Moreover, $V(\mathbf{x}, k+1, \tilde{\mathbf{u}})$ in (29) is the cost function started from the initial sate $\mathbf{x}$ at the sampling time $k+1$ with the nonoptimal control law $\tilde{\mathbf{u}}(\mathbf{x}, k+1)$ as

$$\begin{aligned} \tilde{\mathbf{u}}(\mathbf{x}, k+1) = \big\{&\mathbf{u}^*(k+1; (\mathbf{x}, k+1)), \ \mathbf{u}^*\left(k+2; (\mathbf{x}, k+1)\right), \\ &\ldots, \ \mathbf{u}^*(k+N_c; (\mathbf{x}, k+1)), \ \hat{\mathbf{u}}\big\} \end{aligned} \tag{31}$$

The state trajectory resulted form $\tilde{\mathbf{u}}(\mathbf{x}, k+1)$ is

$$\tilde{\mathbf{x}}(\mathbf{x}, k+1) = \big\{\mathbf{x}^*(k+2; \mathbf{x}), \ \mathbf{x}^*(k+3; \mathbf{x}), \ \ldots, \mathbf{x}^*(k+N_p; \mathbf{x}), \mathbf{f}(\mathbf{x}^*(k+N_p; \mathbf{x}), \hat{\mathbf{u}})\big\} \tag{32}$$

$V(\mathbf{x}, k+1, \tilde{\mathbf{u}})$ can be written as

$$\begin{aligned} V(\mathbf{x}, k+1, \tilde{\mathbf{u}}) = &\sum_{i=k+2}^{k+N_P} \ell\left(\mathbf{x}^*(i; \mathbf{x}), \mathbf{u}^*(i; \mathbf{x})\right) \\ &+ \ell\left(\mathbf{f}\left(\mathbf{x}^*(k+N_p; \mathbf{x}), \hat{\mathbf{u}}\right), \hat{\mathbf{u}}\right) + \mathbf{E}(k+1)^T \mathbf{E}(k+1) \end{aligned} \tag{33}$$

Equation (33) can be rewritten as

$$\begin{aligned} V(\mathbf{x}, k+1, \tilde{\mathbf{u}}) = &\sum_{i=k+1}^{k+N_P} \ell\left(\mathbf{x}^*(i; \mathbf{x}), \mathbf{u}^*(i; \mathbf{x})\right) - \ell\left(\mathbf{x}^*(k+1; \mathbf{x}), \mathbf{u}^*(k+1; \mathbf{x})\right) \\ &+ \ell\left(\mathbf{f}\left(\mathbf{x}^*(k+N_p; \mathbf{x}), \hat{\mathbf{u}}\right), \hat{\mathbf{u}}\right) + \mathbf{E}(k+1)^T \mathbf{E}(k+1) \end{aligned} \tag{34}$$

Using $V^*(\mathbf{x}, k)$, (34) becomes

$$\begin{aligned} V(\mathbf{x}, k+1, \tilde{\mathbf{u}}) = &V^*(\mathbf{x}, k) - \mathbf{E}(k)^T \mathbf{E}(k) - \ell(\mathbf{x}^*(k+1; \mathbf{x}), \mathbf{u}^*(k+1; \mathbf{x})) \\ &+ \ell\left(\mathbf{f}(\mathbf{x}^*(k+N_p; \mathbf{x}), \hat{\mathbf{u}}), \hat{\mathbf{u}}\right) + \mathbf{E}(k+1)^T \mathbf{E}(k+1). \end{aligned} \tag{35}$$

Rearranging (35) yields

$$\begin{aligned} V(\mathbf{x}, k+1, \tilde{\mathbf{u}}) = &V^*(\mathbf{x}, k) + \ell\left(\mathbf{f}\left(\mathbf{x}^*(k+N_p; \mathbf{x}), \hat{\mathbf{u}}\right), \hat{\mathbf{u}}\right) - \ell\left(\mathbf{x}^*(k+1; \mathbf{x}), \mathbf{u}^*(k+1; \mathbf{x})\right) \\ &+ \mathbf{E}(k+1)^T \mathbf{E}(k+1) - \mathbf{E}(k)^T \mathbf{E}(k). \end{aligned} \tag{36}$$

The last two terms in (36) can be defined as a first difference

$$\Delta\mathbf{E}(k)^T\mathbf{E}(k) = \mathbf{E}(k+1)^T\mathbf{E}(k+1) - \mathbf{E}(k)^T\mathbf{E}(k), \tag{37}$$

or

$$\begin{aligned}\Delta\mathbf{E}(k)^T\mathbf{E}(k) &= \mathbf{E}(k)^T\mathbf{E}(k) - \mathbf{E}(k-1)^T\mathbf{E}(k-1)\\ &= \sum_{i=1}^{n}\left[e_i^2(k) - e_i^2(k-1)\right].\end{aligned} \tag{38}$$

Expanding (38) gives

$$\begin{aligned}\Delta\mathbf{E}(k)^T\mathbf{E}(k) &= \sum_{i=1}^{n}\left\{[x_i(k)-\hat{x}_i(k)]^2 - [e_i(k-1)]^2\right\}\\ &= \sum_{i=1}^{n}\left\{[x_i(k)-\mathbf{w}_i(k)\phi(k)]^2 - [e_i(k-1)]^2\right\}.\end{aligned} \tag{39}$$

Using the adaptation law (24), (39) can be rewritten as

$$\begin{aligned}\Delta\mathbf{E}(k)^T\mathbf{E}(k) &= \sum_{i=1}^{n}\left\{\left[x_i(k) - (\mathbf{w}_i(k-1)+\Gamma_i(k)\Psi_i(k))^T\ \boldsymbol{\varphi}\right]^2 - [e_i(k-1)]^2\right\}\\ &= \sum_{i=1}^{n}\left\{\left[x_i(k) - \mathbf{w}_i(k-1)\boldsymbol{\varphi} - \Gamma_i(k)\Psi_i^T(k)\boldsymbol{\varphi}\right]^2 - [e_i(k-1)]^2\right\}\\ &= \sum_{i=1}^{n}\left\{\left[\Gamma_i(k) - \Gamma_i(k)\Psi_i^T(k)\boldsymbol{\varphi}\right]^2 - [e_i(k-1)]^2\right\}.\end{aligned} \tag{40}$$

Using Ψ_i in (25), (40) becomes

$$\Delta\mathbf{E}(k)^T\mathbf{E}(k) = \sum_{i=1}^{n}\left\{\left(\Gamma_i(k) - \Gamma_i(k)\left[\frac{\boldsymbol{\varphi}(k)}{\|\boldsymbol{\varphi}(k)\|^2}\left(1-\lambda_i\frac{|e_i(k-1)|}{|\Gamma_i(k)|}\right)\right]^T\boldsymbol{\varphi}(k)\right)^2 - [e_i(k-1)]^2\right\}.$$

On the basis of the fact that $\boldsymbol{\varphi}^T(k)\boldsymbol{\varphi}(k)/\|\boldsymbol{\varphi}(k)\|^2 = 1$, it gives

$$\Delta\mathbf{E}(k)^T\mathbf{E}(k) = \sum_{i=1}^{n}\left\{\left(\Gamma_i(k) - \Gamma_i(k)\left(1-\lambda_i\frac{|e_i(k-1)|}{|\Gamma_i(k)|}\right)^T\right)^2 - [e_i(k-1)]^2\right\}.$$

After some simplifications, it yields

$$\begin{aligned}\Delta\mathbf{E}(k)^T\mathbf{E}(k) &= \sum_{i=1}^{n}\left\{\lambda_i^2 e_i(k-1)^2 - e_i(k-1)^2\right\}\\ &= -\sum_{i=1}^{n}\left\{(1-\lambda_i^2)e_i(k-1)^2\right\}.\end{aligned} \tag{41}$$

Therefore,

$$\mathbf{E}(k)^T\mathbf{E}(k) = -\mathbf{E}(k-1)^T(\mathbf{I}-\boldsymbol{\lambda})\mathbf{E}(k-1) \tag{42}$$

where $\mathbf{I} \in R^{n\times n}$ is the identity matrix and $\boldsymbol{\lambda} = \operatorname{diag}\left\{\lambda_i^2\right\} \in R^{n\times n}$. Then, using (42), (36) becomes

$$\begin{aligned}V(\mathbf{x},k+1,\tilde{\mathbf{u}}) = V^*(\mathbf{x},k) &+ \ell\left(\mathbf{f}\left(\mathbf{x}^*(k+N_p;\mathbf{x}),\hat{\mathbf{u}}\right),\hat{\mathbf{u}}\right)\\ &- \ell\left(\mathbf{x}^*(k+1;\mathbf{x}),\mathbf{u}^*(k+1;\mathbf{x})\right) - \mathbf{E}^T(k-1)\ (\mathbf{I}-\lambda)\ \mathbf{E}(k-1).\end{aligned} \tag{43}$$

Define

$$\Delta_{N_P}\ell := \ell\left(\mathbf{f}\left(\mathbf{x}^*(k+N_P;\mathbf{x}),\hat{\mathbf{u}}\right),\hat{\mathbf{u}}\right) - \ell(\mathbf{x}^*(k+1;\mathbf{x}),\mathbf{u}^*(k+1;\mathbf{x})) \tag{44}$$

as the difference of ℓ after N_P samples. Then, (43) can be rewritten as

$$V(\mathbf{x},k+1,\tilde{\mathbf{u}}) = V^*(\mathbf{x},k) + \Delta_{N_P}\ell - \mathbf{E}(k-1)^T(\mathbf{I}-\boldsymbol{\lambda})\mathbf{E}(k-1). \tag{45}$$

If $\lambda_i < 1 \ \ \forall i$, then $-\mathbf{E}(k-1)^T(\mathbf{I}-\lambda)\,\mathbf{E}(k-1) \leqslant 0$. Moreover, one important condition in solving the optimization problem (18) is that ℓ must not increase after N_P samples. In other words, the cost function must be a decreasing function after every sampling time. Hence, $\Delta_{N_P}\ell \leqslant 0$. Therefore, (45) becomes

$$V(\mathbf{x},k+1,\tilde{\mathbf{u}}) \leqslant V^*(\mathbf{x},k). \tag{46}$$

According to (29), (46) can be represented as

$$V^*(\mathbf{x},k+1) \leqslant V^*(\mathbf{x},k). \tag{47}$$

Therefore, ΔL becomes

$$\Delta L = \Delta V^*(\mathbf{x}) = V^*(\mathbf{x},k+1) - V^*(\mathbf{x},k) \leqslant 0. \tag{48}$$

This concludes the proof. □

6. SIMULATION RESULTS AND DISCUSSIONS

The simulated results of the proposed control method are presented in this section. Parameters of the robot manipulator and DC servomotors are given in Tables I and II, respectively. The sampling rate T is equal to 0.5 s. On the basis of the length of the links, $D_{P\,\min}$ and $D_{P\,\max}$ are 0 and 2.4 m and $D_{O\,\min}$ and $D_{O\,\max}$ are 0 and 1.2 m, respectively. Furthermore, on the basis of the robot parameters, it is assumed that $\dot{\theta}_{\min}$ and $\dot{\theta}_{\max}$ are equal to $-400°$/s and $400°$/s, respectively. In all simulations, $N_P = 5$, $N_C = 2$, $Q = 10$, $R = 0.9$, and $N = 100$. Moreover, it is assumed that the velocity and acceleration of the moving objects are within the acceptable range as compared to the manipulator joints limitations.

There are four NNs for modeling the robot (i.e., prediction of the next position of the corresponding joint) and one NN for prediction of moving targets. The NNs for modeling the robot consists of nine nodes in the input layer, 10 neurons in one hidden layer, and one neuron in the output layer (Figure 4). The activation functions of the neurons in the hidden layer ($\boldsymbol{\varphi}(\cdot)$ in (14)) are of tangent hyperbolic type and in the output layer of linear type. The NN for predicting the position of the

Table I. Parameters of robot manipulator.

Link	1	2	3	4
l (m)	1	0.5	0.4	0.3
m (kg)	1	0.5	0.4	0.3

Table II. Parameters of DC servomotors.

Motor	1	2	3	4
R_a	6.51	6.51	6.51	6.51
K_E	0.7	0.7	0.7	0.7
K_T	0.5	0.5	0.5	0.5
B_m	6.4×10^{-3}	6.4×10^{-3}	6.4×10^{-3}	6.4×10^{-3}
J_m	0.2	0.2	0.2	0.2
R	1:100	1:100	1:10	1:10
V_t	24	24	24	24

moving object consists of nine nodes in the input layer, 10 neurons in one hidden layer, and three neurons in the output layer (Figure 5).

In order to implement the adaptation law in (25), the equation of $\Psi_i(k)$ is slightly changed as follows:

$$\Psi_i(k) = \frac{\varphi_i(k)}{\|\varphi_i(k)\|^2}\left(1 - \lambda_i \frac{|e_i(k)|}{\eta_i + |\Gamma_i(k)|}\right) \tag{49}$$

where η_i is a small positive constant and has been added only to prevent the denominator to become zero; it does not affect the validity of the analytical work presented in Section 5, and hence, the proof of Theorem 1 remains intact. The constants of the adaptation law in (49) are selected as $\eta_i = 0.01\lambda_i = 0.9$.

In the first part of simulations, a rectangular path in the Cartesian space with static obstacles inside the workspace is considered (Figure 6). Simulation results are shown in Figures 6–14. As Figures 7–10 show, the end effector tracks the predefined path in the workspace with good accuracies. Figures 11–13 show input voltages, positions, and velocities of manipulator joints, respectively. The attached animation file (Anim1-Fig6, Supporting information) shows that the end effector tracks the predefined path without any collision with obstacles located inside the workspace. Figure 14 shows the simulation time for each sampling time. As this figure shows, the simulation time is above the sampling time for several instances. However, it should be noted that (i) simulations are carried out in MATLAB software environment, which requires more time than a lower level computer programming such as C++; (ii) in simulations, solving the robot equations incurs some computational times, which is not required in practice; and (iii) some aspects of computations, such as training of NNs, can be processed in parallel to spare computational times. Hence, in practice, the simulation

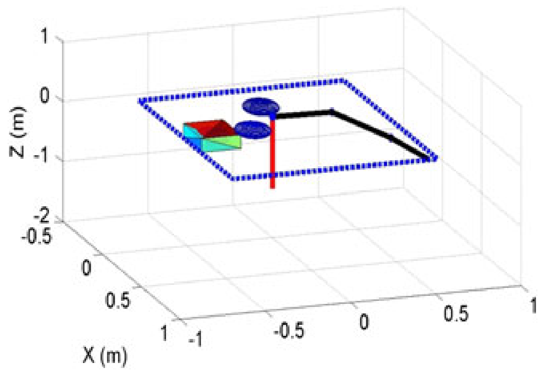

Figure 6. Path tracking with static obstacles in the workspace (see film animation Anim1-Fig6 in the Supporting information).

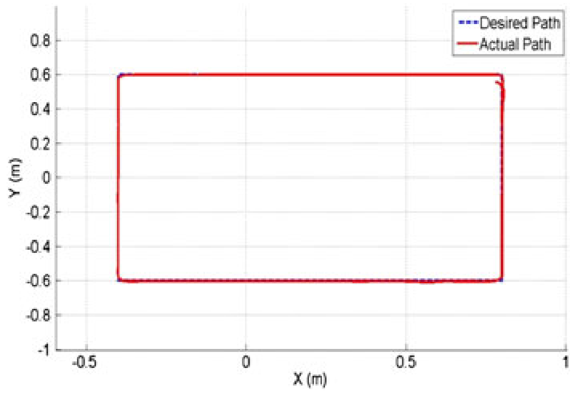

Figure 7. Desired and actual end-effector path.

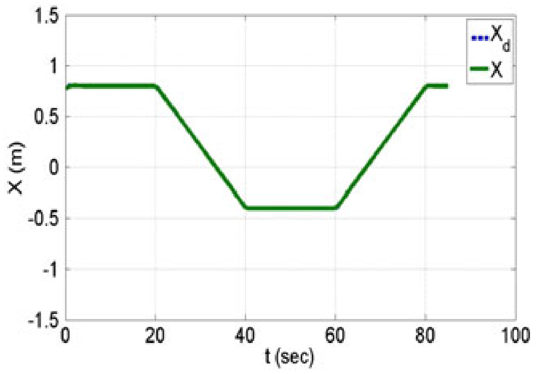

Figure 8. Desired and actual end-effector path along the x-axis.

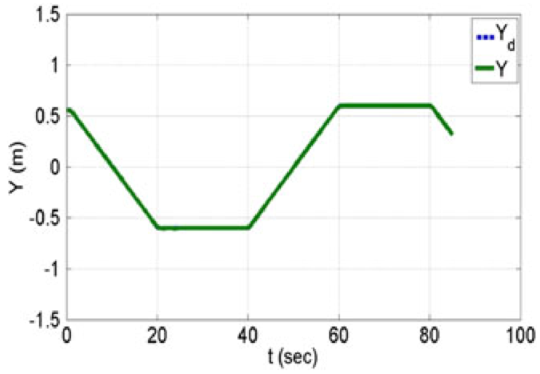

Figure 9. Desired and actual end-effector path along the y-axis.

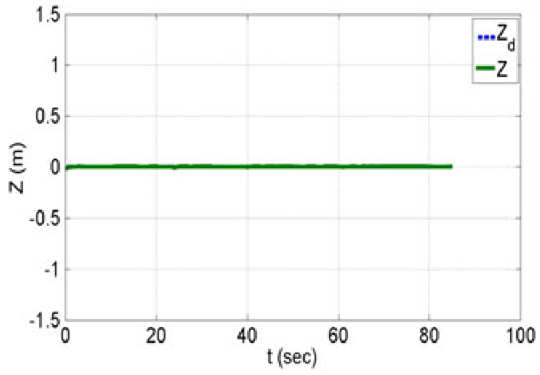

Figure 10. Desired and actual end-effector path along the z-axis.

run time can be substantially less than what is shown in Figure 14. Therefore, writing the computer program in C++ can yield a real-time NMPC control method.

In the second part of simulations, a rectangular path in the Cartesian space with static and moving obstacles inside the workspace is considered (Figure 15). To show the prediction accuracy of the NN, a highly nonlinear motion for the target and obstacles is considered. The motion equation of the moving obstacle is defined as

$$\begin{aligned} x_{\text{obs}}(t) &= R_1(t)\cos(0.01\pi t) \\ y_{\text{obs}}(t) &= R_1(t)\sin(0.01\pi t) \\ z_{\text{obs}}(t) &= -0.1 + R_2(t)\sin(0.05\pi t) \end{aligned} \tag{50}$$

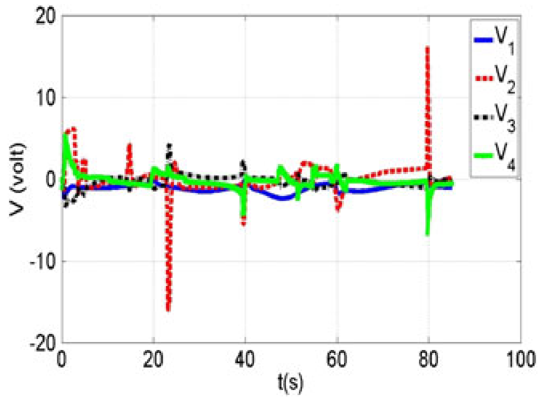

Figure 11. Input voltages of DC servomotors.

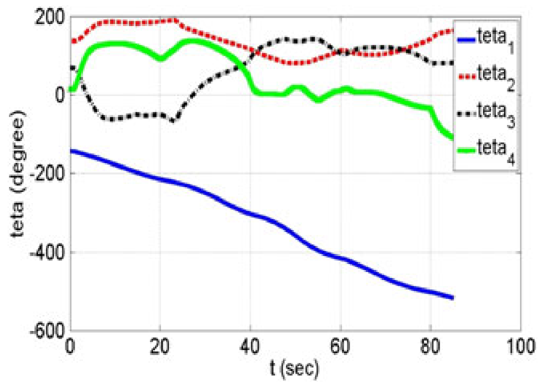

Figure 12. Position of joints.

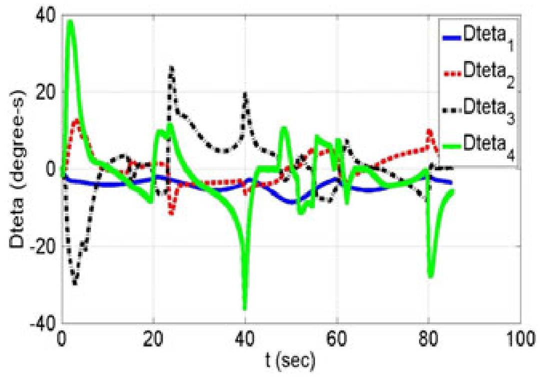

Figure 13. Velocity of joints.

where

$$\begin{aligned} R_1(t) &= 0.1\cos(0.06\pi t) + 0.5 \\ R_2(t) &= 0.2\cos(0.01\pi t) \end{aligned} \tag{51}$$

in which t is the time and $[x_{\text{obs}}(t)\ \ y_{\text{obs}}(t)\ \ z_{\text{obs}}(t)]$ is the obstacle coordinates in the 3D space (Figure 16). Simulation results are shown in Figures 15–21. Figure 17 shows that the end effector tracks the predefined path with good accuracies. The attached animation file (Amin2-Fig15, Supporting information) shows that the end effector tracks the predefined path without any collision with the static and dynamic obstacles. Figures 18 and 19 show the ability of NN to predict coordi-

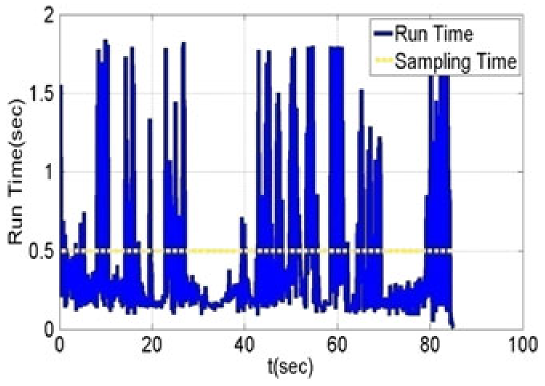

Figure 14. Simulation run time for every sampling time.

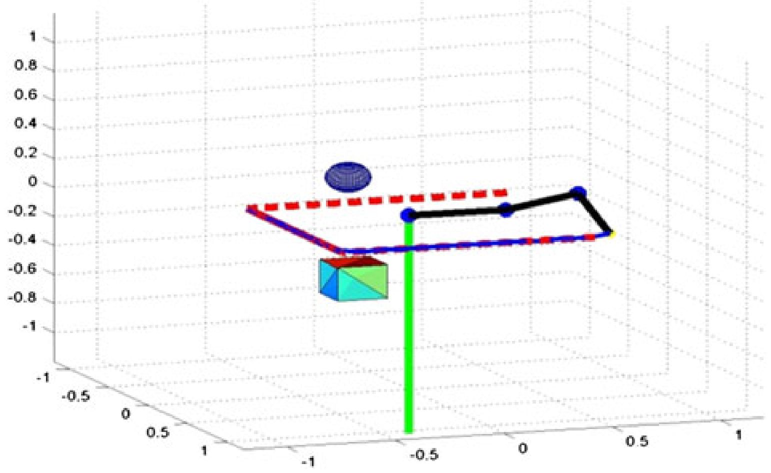

Figure 15. Desired rectangular path with static (cube) and dynamic (sphere) obstacles in workspace (see film animation Anim2-Fig15 in the Supporting information).

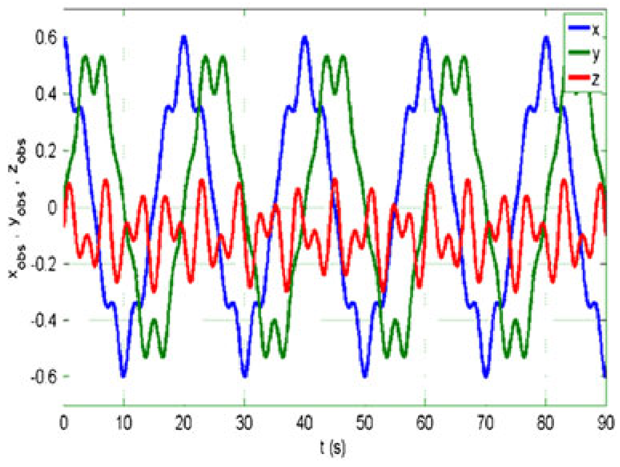

Figure 16. Coordinates of moving obstacle.

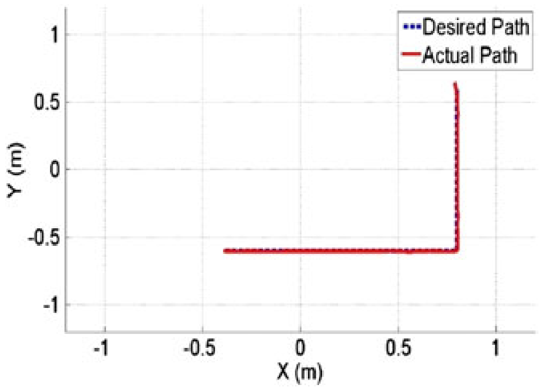

Figure 17. Desired and actual end-effector path.

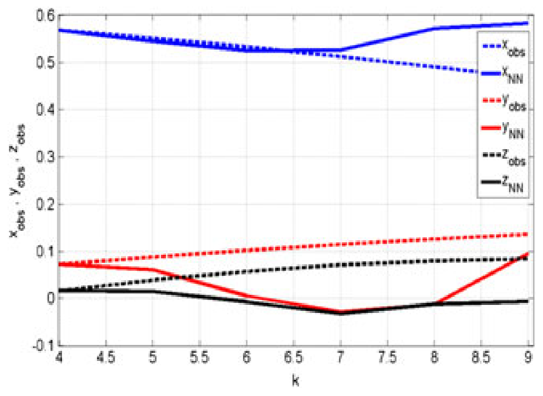

Figure 18. NN prediction over prediction horizon at sampling time $k = 4$.

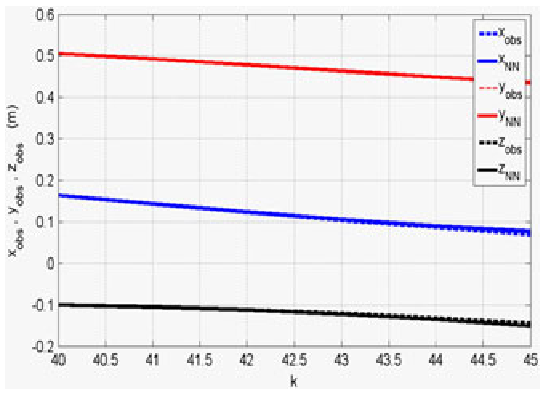

Figure 19. NN prediction over prediction horizon at sampling time sampling time $k = 40$.

nates of the moving obstacle at training instances $k = 4$ and $k = 40$, respectively. Figure 20 shows the maximum prediction error over the prediction horizon at each sampling time. As these figures show, NNs are able to learn the behavior of the moving obstacle as the time passes. Figure 21 shows that the training time for NNs at every sampling time is less than the sampling rate $T = 0.5$ s.

In the third part of simulations, in addition to the static and moving obstacles inside the workspace, the end effector has to catch a moving target (Figure 22). The motion equation of the moving obstacle in 3D space is defined as (Figure 23).

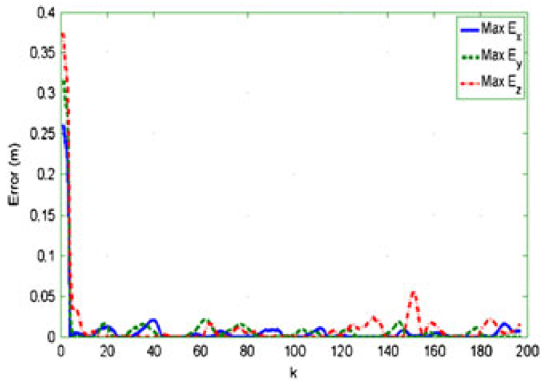

Figure 20. Maximum prediction error of NNs at each sampling time.

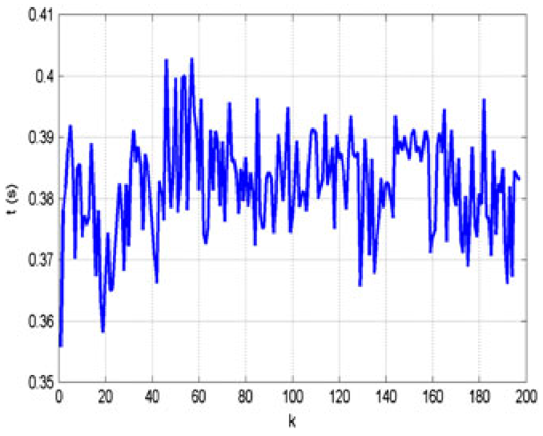

Figure 21. Training time for NNs at every sampling time.

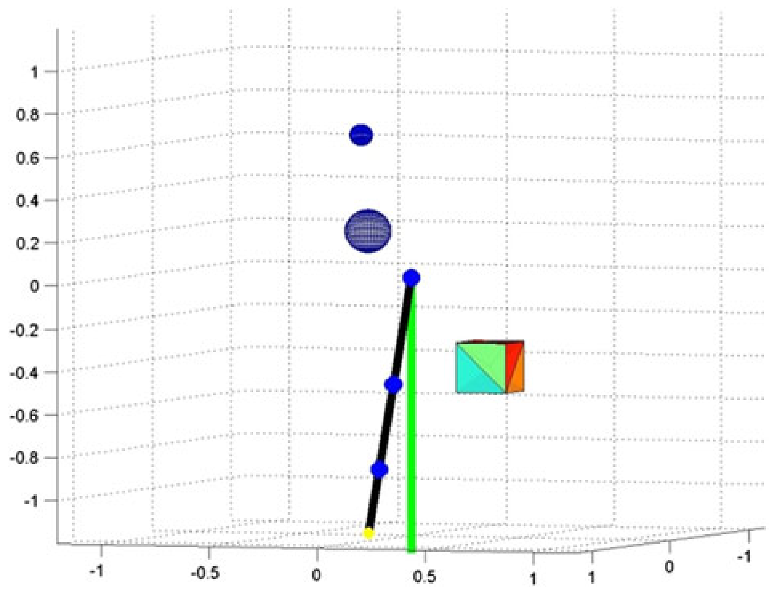

Figure 22. Moving target (small sphere) static (cube) & dynamic (sphere) obstacle inside workspace (see film animation Anim3-Fig22 in the Supporting information).

$$
\begin{aligned}
x_t(t) &= R_3(t)\cos(0.01\pi t) \\
y_t(t) &= R_3(t)\sin(0.01\pi t) \\
z_t(t) &= 0.5 + 0.2\cos(0.01t)
\end{aligned}
\tag{52}
$$

where

$$
R_3(t) = 0.2\cos(0.06\pi t) + 0.5 \tag{53}
$$

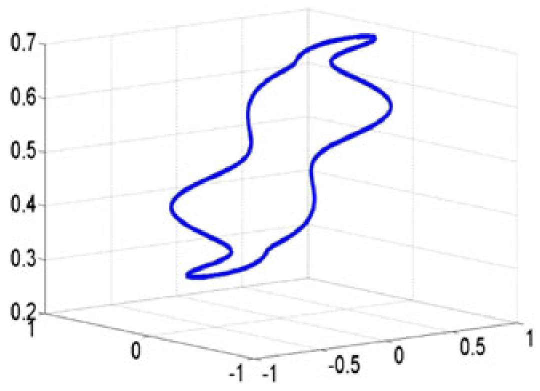

Figure 23. Path of moving target in 3D.

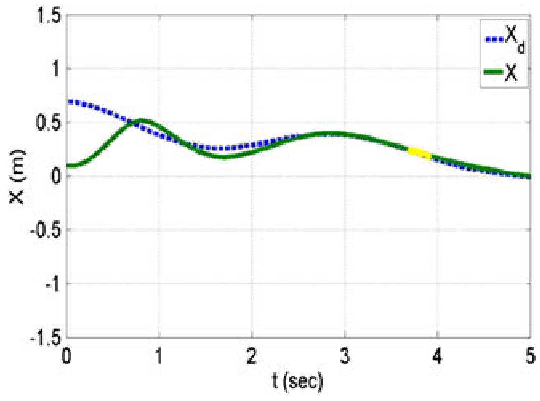

Figure 24. Path of moving target and end effector along the x-axis.

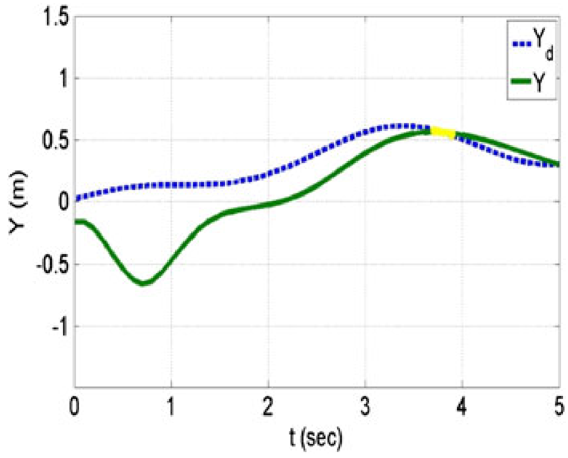

Figure 25. Path of moving target and end effector along the y-axis.

Simulation results are shown in Figures 24–27. Figures 24–26 show that the end effector reaches the target in a reasonable time. Figure 27 shows input voltages, which indicates that the voltages applied to the servomotors has been saturated initially in order to produce enough velocity to reach quickly to the goal and, at the same time, avoid the obstacles. The attached animation file (Anim3-Fig22, Supporting information) shows that the end effector reaches the moving target swiftly without any collision with the static and moving obstacles inside the workspace.

It should be emphasized that in all the aforementioned results, it is assumed that a solution exists to the trajectory following and obstacles avoidance problem. For instance, if an obstacle moves so

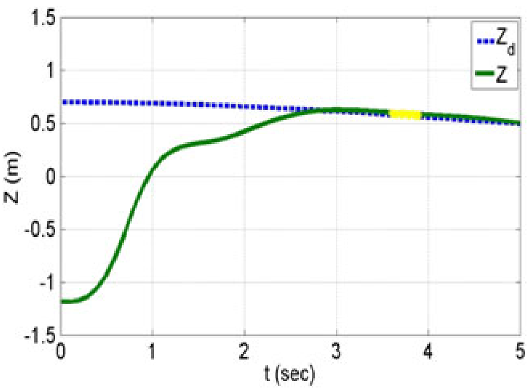

Figure 26. Path of moving target and end effector along the z-axis.

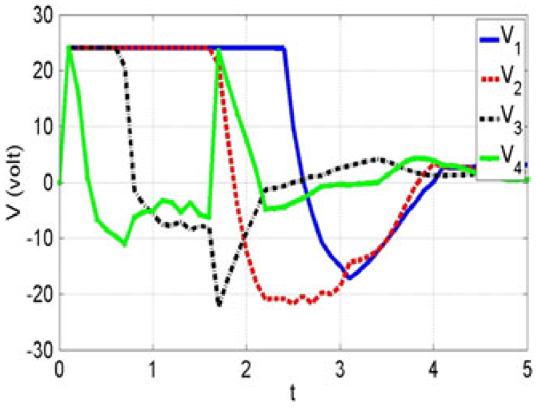

Figure 27. Input voltages of DC servomotors.

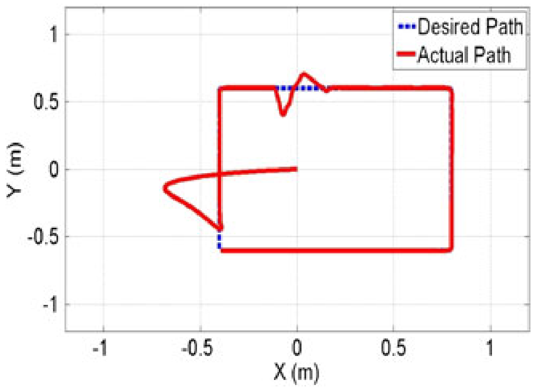

Figure 28. Desired and actual end-effector path.

quickly that it collides with a link of the robot, or when the trajectory of the moving target is out of the workspace of the robot, then there is no solution to the optimization problem.

Next, in order to show the robustness of the proposed method against changes in system parameters, the mass of link 4 is increased by 20 kg at $t = 20$ s. Although a 20-kg change in the mass of the fourth link might be unrealistic, nevertheless, the aim is to show the ability of the proposed method even against considerable changes in the system parameters. Simulation results for this case are shown in Figure 28. As this figure shows, the controller can adapt itself very quickly to the changes in the system parameters in a relatively short time without any collision with obstacles.

7. CONCLUSION

In this paper, an adaptive NMPC method for path tracking and obstacle avoidance of robot manipulators was proposed. For this reason, two terms were introduced in the cost function of NMPC, one for the tracking problem and the other for the obstacle avoidance. Moreover, singularities were avoided by introducing constraints to joints velocities. In addition, to avoid collisions with moving obstacles and capturing moving target, the future position of obstacles and the moving target in 3D space was predicted using NNs. Using online training for NNs, no knowledge about obstacles and motion of the moving object was required. Furthermore, using NNs for model prediction in NMPC, no knowledge about system parameters was necessary, and system robustness against changes in system parameters was achieved. To guarantee stability of the closed-loop system, the adaptation law for the NNs weights in the prediction model was obtained using the Lyapunov's direct method. Simulations were carried out on a 4-DoF special and redundant robot manipulator. It was shown that the proposed method is a viable tool for path tracking and obstacle avoidance for robot manipulators.

APPENDIX A

Sequential (or successive) quadratic programming is one of the fastest and most efficient methods for the numerical solution of constrained nonlinear optimization problems. This method closely mimics the Newton's method for the constrained optimization just as it is performed for the unconstrained optimization. In this method, an approximation is made recursively of the Hessian of the Lagrangian function using a quasi-Newton updating method. This is then used to generate a quadratic programming subproblem whose solution is used to form a search direction for a line search procedure. In the following, a brief introduction of this method is given. For more details, the reader may refer to books on optimization programming methods (e.g., [42]). The function *fmincon* in MATLAB software uses SQP for solving nonlinear constraint optimization problems.

Consider a nonlinear programming problem of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{h}(x) \geqslant 0 \\ & \mathbf{g}(x) = 0 \end{aligned} \tag{A.1}$$

where $\mathbf{x} = [\, x_1 \ \cdots \ x_n \,]^T \in R^n$, $f(\mathbf{x}) \in R^n \to R$ is the objective functional, and functions $\mathbf{h}(\mathbf{x}) \in R^n \to R^m$ and $\mathbf{g}(\mathbf{x}) \in R^n \to R^p$ describe the inequality and equality constraints, respectively.

The Lagrangian for this problem is defined as

$$L(\mathbf{x}, \lambda, \sigma) := f(\mathbf{x}) - \lambda^T \mathbf{h}(\mathbf{x}) - \sigma^T \mathbf{g}(\mathbf{x}) \tag{A.2}$$

where λ and σ are the Lagrange multipliers. At an iterate $\mathbf{x}_k$, the basic SQP algorithm defines an appropriate search direction $\mathbf{d}(\mathbf{x})$ as a solution to the quadratic programming subproblem

$$\begin{aligned} \min_{\mathbf{d}} \quad & L(\mathbf{x}_k, \lambda_k, \sigma_k) + \nabla_x L(\mathbf{x}_k, \lambda_k, \sigma_k)^T \mathbf{d}(\mathbf{x}_k) + \frac{1}{2}\mathbf{d}(\mathbf{x}_k)^T \nabla^2_{\mathbf{xx}} L(\mathbf{x}_k, \lambda_k, \sigma_k)^T \mathbf{d}(\mathbf{x}_k) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}_k) + \nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x}_k)^T \mathbf{d}(\mathbf{x}_k) \geqslant 0 \\ & \mathbf{g}(\mathbf{x}_k) + \nabla_{\mathbf{x}} g(\mathbf{x}_k)^T \mathbf{d}(\mathbf{x}_k) = 0 \end{aligned} \tag{A.3}$$

where $\nabla_{\mathbf{x}} L(\mathbf{x},\lambda,\sigma) = \nabla_{\mathbf{x}} f(\mathbf{x}) - \nabla_{\mathbf{x}}\mathbf{h}(\mathbf{x})\lambda - \nabla_{\mathbf{x}}\mathbf{g}(\mathbf{x})\sigma$. For a functional $f(\mathbf{x}) \in R^n \to R$, the $\nabla_{\mathbf{x}} f(\mathbf{x})$ denotes the gradient of $f(\mathbf{x})$ at $\mathbf{x} \in R^n$, that is,

$$\nabla_{\mathbf{x}} f(\mathbf{x}) := \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \frac{\partial f(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}^T \tag{A.4}$$

Moreover, $\nabla^2_{\mathbf{xx}} f(\mathbf{x})$ is referred to as the Hessian matrix of $f(\mathbf{x})$ at $\mathbf{x} \in R^n$, that is, the matrix of second partial derivatives is determined by

$$\nabla^2_{x_i x_j} f(\mathbf{x}) := \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_i}, \quad 1 \leqslant i,j \leqslant n \tag{A.5}$$

For the vector-valued functions $\mathbf{h}(\mathbf{x}) \in R^n \to R^m$, the symbol ∇ is used to find the Jacobian of $\mathbf{h}$, that is,

$$\nabla_{\mathbf{x}}\mathbf{h}(\mathbf{x}) := \begin{bmatrix} \nabla_{\mathbf{x}} h_1(\mathbf{x}) & \nabla_{\mathbf{x}} h_2(\mathbf{x}) & \cdots & \nabla_{\mathbf{x}} h_m(\mathbf{x}) \end{bmatrix} \tag{A.6}$$

APPENDIX B

$$\begin{aligned}
C(1,1) &= \begin{pmatrix} -\frac{1}{3}m_2 l_2^2 \mathrm{s}_{22} - m_3 l_2^2 \mathrm{s}_{22} - m_3 l_2 l_3 \mathrm{s}_{223} - \frac{1}{3}m_3 l_3^2 \mathrm{s}_{2233} - m_4 l_3^2 \mathrm{s}_{2233} - m_4 l_2^2 \mathrm{s}_{22} \\ -2m_4 l_2 l_3 \mathrm{s}_{223} - m_4 l_4 l_3 \mathrm{s}_{22334} - m_4 l_4 l_2 \mathrm{s}_{2234} - \frac{1}{3}m_4 l_4^2 \mathrm{s}_{223344} \end{pmatrix} \dot\theta_1 \dot\theta_2 \\
&+ \begin{pmatrix} -m_3 l_2 l_3 \mathrm{s}_{23}\mathrm{c}_2 - \frac{1}{3}m_3 l_3^2 \mathrm{s}_{2233} - m_4 l_3^2 \mathrm{s}_{2233} - 2m_4 l_3 l_2 \mathrm{s}_{23}\mathrm{c}_2 - m_4 l_3 l_4 \mathrm{s}_{22334} \\ -m_4 l_2 l_4 \mathrm{s}_{234}\mathrm{c}_2 - \frac{1}{3}m_4 l_4^2 \mathrm{s}_{223344} \end{pmatrix} \dot\theta_1 \dot\theta_3 \\
&+ \left(-m_4 l_4 l_3 \mathrm{s}_{234} c_{23} - m_4 l_4 l_2 \mathrm{s}_{234} c_2 - \frac{1}{3} m_4 l_4^2 \mathrm{s}_{223344} \right) \dot\theta_1 \dot\theta_4 \\
C(2,1) &= \begin{pmatrix} \frac{1}{6}m_2 l_2^2 \mathrm{s}_{22} + \frac{1}{2}m_3 l_2^2 \mathrm{s}_{22} + \frac{1}{2}m_3 l_2 l_3 \mathrm{s}_{223} + \frac{1}{6}m_3 l_3^2 \mathrm{s}_{2233} + \frac{1}{2}m_4 l_3^2 \mathrm{s}_{2233} \\ +\frac{1}{2}m_4 l_2^2 \mathrm{s}_{22} + m_4 l_2 l_3 \mathrm{s}_{223} + \frac{1}{2}m_4 l_4 l_3 \mathrm{s}_{22334} + \frac{1}{2}m_4 l_2 l_4 \mathrm{s}_{2234} \\ +\frac{1}{6}m_4 l_4^2 \mathrm{S}_{223344} \end{pmatrix} \dot\theta_1^2 \\
&+ \left(-\frac{1}{2}m_3 l_2 l_3 \mathrm{s}_3 - m_4 l_3 l_2 \mathrm{s}_3 \right) \dot\theta_3^2 + \left(-\frac{1}{2}m_4 l_4 l_3 \mathrm{s}_4 \right) \dot\theta_4^2 + (-m_4 l_3 l_4 \mathrm{s}_4)\, \dot\theta_2 \dot\theta_4 \\
&+ (-m_3 l_3 l_2 \mathrm{s}_3 - 2m_4 l_3 l_2 \mathrm{s}_3)\, \dot\theta_2 \dot\theta_3 + (-m_4 l_3 l_4 \mathrm{s}_4)\, \dot\theta_3 \dot\theta_4 \qquad \text{(B.1)} \\
C(3,1) &= \begin{pmatrix} \frac{1}{2}m_3 l_3 l_2 \mathrm{s}_{23}\mathrm{c}_2 + \frac{1}{6}m_3 l_3^2 \mathrm{s}_{2233} + \frac{1}{2}m_4 l_3^2 \mathrm{s}_{2233} + m_4 l_2 l_3 \mathrm{s}_{23}\mathrm{c}_2 + \frac{1}{2}m_4 l_4 l_3 \mathrm{s}_{22334} \\ +\frac{1}{2}m_4 l_4 l_2 \mathrm{s}_{234} c_2 + \frac{1}{6}m_4 l_4^2 \mathrm{s}_{223344} \end{pmatrix} \dot\theta_1^2 \\
&+ \left(\frac{1}{2}m_3 l_2 l_3 \mathrm{s}_3 + m_4 l_3 l_2 \mathrm{s}_3 \right) \dot\theta_2^2 + (-m_4 l_3 l_4 \mathrm{s}_4)\, \dot\theta_2 \dot\theta_4 + \left(-\frac{1}{2}m_4 l_3 l_4 \mathrm{s}_4 \right) \dot\theta_4^2 \\
&+ (-m_4 l_3 l_4 s_4)\, \dot\theta_3 \dot\theta_4 \\
C(4,1) &= \left(\frac{1}{2}m_4 l_4 l_3 \mathrm{s}_{234} c_{23} + \frac{1}{2}m_4 l_4 l_2 \mathrm{s}_{234} \mathrm{c}_2 + \frac{1}{6}m_4 l_4^2 \mathrm{s}_{223344} \right) \dot\theta_1^2 + \left(\frac{1}{2}m_4 l_4 l_3 \mathrm{s}_4 \right) \dot\theta_2^2 \\
&+ \left(\frac{1}{2}m_4 l_4 l_3 \mathrm{s}_4 \right) \dot\theta_3^2 + (m_4 l_4 l_3 \mathrm{s}_4)\, \dot\theta_3 \dot\theta_4
\end{aligned}$$

$$\begin{aligned}
G(1,1) &= 0 \\
G(1,2) &= \left(\frac{1}{2}\mathrm{m}_2 g l_2 + \mathrm{m}_3 g l_2 + \mathrm{m}_4 g l_2 \right) \mathrm{c}_2 + \left(\frac{1}{2}\mathrm{m}_3 g l_3 + m_4 g l_3 \right) \mathrm{c}_{23} + \frac{1}{2}\mathrm{m}_4 g l_4 \mathrm{c}_{234} \\
G(3,1) &= \left(\frac{1}{2}\mathrm{m}_3 g l_3 + m_4 g l_3 \right) c_{23} + \frac{1}{2}\mathrm{m}_4 g l_4 \mathrm{c}_{234} \qquad \text{(B.2)} \\
G(4,1) &= \frac{1}{2}\mathrm{m}_4 g l_4 c_{234}
\end{aligned}$$

$$
\begin{aligned}
M(1,1) &= \frac{2}{3}m_1l_1^2 + \left(\frac{1}{3}m_2l_2^2 + m_3l_2^2 + m_4l_2^2\right)\mathrm{c}_2^2 + \left(\frac{1}{3}m_3l_3^2 + m_4l_3^2\right)\mathrm{c}_{23}^2 + \frac{1}{3}m_4l_4^2c_{234}^2 \\
&\quad + (m_3l_3l_2 + 2m_4l_3l_2)\,c_{23}\mathrm{c}_2 + m_4l_4l_3c_{234}c_{23} + m_4l_4l_2c_{234}\mathrm{c}_2 \\
M(2,2) &= \frac{1}{3}m_2l_2^2 + m_3l_2^2 + \frac{1}{3}m_3l_3^2 + m_4l_3^2 + m_4l_2^2 + \frac{1}{3}m_4l_4^2 + m_4l_4l_2 + m_4l_3l_4\mathrm{c}_4 \\
&\quad + (m_3l_3l_2 + 2m_4l_2l_3)\,\mathrm{c}_3 \\
M(3,3) &= \frac{1}{3}m_3l_3^2 + m_4l_3^2 + \frac{1}{3}m_4l_4^2 + m_4l_3l_4\mathrm{c}_4 \\
M(4,4) &= \frac{1}{3}m_4l_4^2 \\
M(2,3) &= M(3,2) = \frac{1}{3}m_3l_3^2 + m_4l_3^2 + \frac{1}{2}m_4l_4l_2 + \frac{1}{3}m_4l_4^2 + \left(\frac{1}{2}m_3l_3l_2 + m_4l_2l_3\right)\mathrm{c}_3 \qquad \text{(B.3)} \\
&\quad + m_4l_4l_3c_4 \\
M(2,4) &= M(4,2) = \frac{1}{2}m_4l_4l_2 + \frac{1}{3}m_4l_4^2 + \frac{1}{2}m_4l_4l_3\mathrm{c}_4 \\
M(3,4) &= M(4,3) = \frac{1}{3}m_4l_4^2 + \frac{1}{2}m_4l_3l_4\mathrm{c}_4 \\
M(1,2) &= M(2,1) = 0 \\
M(1,3) &= M(3,1) = 0 \\
M(1,4) &= M(4,1) = 0
\end{aligned}
$$

where l_i and $m_i(i = 1, \ldots, 4)$ are the length and mass of the ith link, respectively, θ_i and $\dot{\theta}_i$ are the angular position and the angular velocity of the ith joint, respectively, and $c_i := \cos(\theta_i), s_i := \sin(\theta_i), c_{ij} := \cos(\theta_i + \theta_j), s_{ij} := \sin(\theta_i + \theta_j)$, and so forth.

REFERENCES

1. Spong MW, Hutchinson S, Vidyasagar M. *Robot Modeling and Control*. John Wiley & Sons: New York, 2006.
2. Craig JJ. *Introduction to Robotics: Mechanics and Control*. Addison- Wesley: Reading, Massachusetts, 1989.
3. Duleba I, Sasiadek JZ. Nonholonomic motion planning based on Newton algorithm with energy optimization. *IEEE Transactions on Control Systems Technology* 2003; **11**(3):355–363.
4. Zha XF. Optimal pose trajectory planning for robot manipulators. *Mechanism and Machine Theory* 2002; **37**(10):1063–1086.
5. Visioli A, Legnani G. On the trajectory tracking control of industrial SCARA robot manipulators. *IEEE Transactions on Industrial Electronics* 2002; **49**:224–232.
6. Galicki M. Path-constrained control of a redundant manipulator in a task space. *Robotics and Autonomous Systems* 2006; **54**:234–243.
7. LaValle SM. *Planning Algorithms*. Cambridge University Press: Cambridge, UK, 2006.
8. Croft EA, Fenton RG, Benhabib B. Optimal rendezvous-point selection for robotic interception of moving object. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 1998; **28**:192–204.
9. Chu J. Robotic interception of moving object using an augmented ideal proportional navigation guidance technique. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans* 2000; **30**:238–250.
10. Schwarzer F, Saha M, Latombe JC. Adaptive dynamic collision checking for single and multiple articulated robots in complex environments. *IEEE Transactions on Robotics* 2005; **21**:338–353.
11. Yoshikawa T. Analysis and control of robot manipulators with redundancy. *International Symposium on Robotic Research*, Cambridge, MA, USA, 1984; 735–747.
12. Garga DP, Kumarb M. Optimization techniques applied to multiple manipulators for path planning and torque minimization. *Engineering Applications of Artificial Intelligence* 2002; **15**:241–252.
13. Chettibi T, Lehtihet HE, Haddada M, Hanchi S. Minimum cost trajectory planning for industrial robots. *European Journal of Mechanics A/Solids* 2004; **23**:703–715.

14. Hu X, Wang H, Ho M. Using nonlinear constrained optimization methods to solve manipulators path planning with hybrid genetic algorithms. *IEEE International Conference on Robotics and Biomimetics*, Shatin, China, 2005; 718–723.
15. Xu W, Liang B, Li C, Qiang W, Xu Y, Lee KK. Non-holonomic path planning of space robot based on genetic algorithm. *IEEE International Conference on Robotics and Biomimetics*, Kunming, China, 2006; 1471–1476.
16. Huang P, Yan J, Xu Y, Xu W, Liang B. Genetic algorithms-based minimum torque path planning for space manipulator. *6th World Congress on Intelligent Control and Automation*, Dalian, China, 2006; 3575–3579.
17. Sullivan JCW, Pipe AG. Path planning for redundant robot manipulators: a global optimization approach using evolutionary search. *IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, CA, USA, 1998; 2396–2400.
18. Chiacchio P, Chiaverini S, Sciavicco L, Siciliano B. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *International Journal of Robotics Research* 1991; **10**:410–426.
19. Chen JL, Liu JS, Lee WC, Liang TC. On-line multi-criteria based collision-free posture generation of redundant manipulator in constrained workspace. *Robotica* 2002; **20**:625–636.
20. Han Y. Simultaneous translational and rotational tracking in dynamic environments: theoretical and practical viewpoints. *IEEE Transactions on Robotics and Automation* 2004; **20**:309–318.
21. Lewis FL, Abdallah CT, Dawson DN. *Control of Robot Manipulators Theory and Practice*. Marcel Dekker Inc.: New York, 2004.
22. Jafarov EM, Parlakçi MNA, Istefanopulos Y. A new variable structure PID-controller design for robot manipulators. *IEEE Transactions on Control Systems Technology* 2005; **13**:122–130.
23. Oya M, Su C, Kobayashi T. State observer-based robust control scheme for electrically driven robot manipulators. *IEEE Transactions on Robotics* 2004; **20**:796–804.
24. Barambones O, Etxebarria V. Robust neural control for robotic manipulators. *Automatica* 2006; **38**:235–242.
25. Corradini ML, Fossi V, Giantomassi A, Ippoliti G, Longhi S, Orlando G. Discrete time sliding mode control of robotic manipulators: development and experimental validation. *Control Engineering Practice* 2012; **20**: 816–822.
26. Fujita M, Kawai H, Spong MW. Passivity-based dynamic visual feedback control for three-dimensional target tracking: stability and L2-gain performance analysis. *IEEE Transactions on Control Systems Technology* 2007; **15**:40–52.
27. Sun F, Li L, Li H, Liu H. Neuro-fuzzy dynamic-inversion-based adaptive control for robotic manipulators: discrete time case. *IEEE Transactions on Industrial Electronics* 2007; **54**:1342–1351.
28. Hsua S, Fu L. A fully adaptive decentralized control of robot manipulators. *Automatica* 2006; **42**:1761–1767.
29. López-Araujo DJ, Zavala-Río A, Santibáñez V, Reyes F. Output-feedback adaptive control for the global regulation of robot manipulators with bounded inputs. *International Journal of Control, Automation and Systems* 2013; **11**:105–115.
30. Han HG, Wu XL, Qiao JF. Real-time model predictive control using a self-organizing neural network. *IEEE Transactions on Neural Networks and Learning Systems* 2013; **24**:1425–1436.
31. Yan Z, Wang J. Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks. *IEEE Transactions and Neural Networks and Learning Systems* 2013. Online: DOI 10.1109/TNNLS.2013.2275948.
32. Ławryńczuk M. Training of neural models for predictive control. *Neurocomputing* 2010; **73**:1332–1343.
33. Salahshoor K, Zakeri S, Haghighat Sefat M. Stabilization of gas-lift oil wells by a nonlinear model predictive control scheme based on adaptive neural network models. *Engineering Applications of Artificial Intelligence* 2013; **26**:1902–1910.
34. Zhicheng X, Bin Z, Qingbin J. Application of neural network for nonlinear predictive control. *Advanced Materials Research* 2012; **562-564**:1964–1967.
35. Abdolvand M, Fatehi MH. Model-base predictive control for vibration suppression of a flexible manipulator. *International Conference on Control (UKACC)*, Cardiff, 2012; 562–567.
36. Ganglo JA, Mathelin MF. High speed visual servoing of a 6 DOF manipulator using multivariable predictive control. *Advanced Robotics* 2003; **17**:993–1021.
37. Koivo AJ, Houshangi N. Real-time vision feedback for servoing robotic manipulator with self-tuning controller. *IEEE Transactions on Systems, Man, and Cybernetics* 1991; **21**:134–142.
38. Kara K, Missoum TE, Hemsas KE, Hadjili ML. Control of a robotic manipulator using neural network based predictive control. *17th IEEE International Conference on Electronics, Circuits, and Systems*, Athens, 2010; 1104–1107.
39. Henmi T, Ohta T, Deng M, Inoue A. Tracking control of a two-link planar manipulator using nonlinear model predictive control. *International Journal of Innovative Computing, Information and Control*; **6**:2977–2984.
40. Allgower F, Findeisen R, Nagy ZK. Nonlinear model predictive control: from theory to application. *Journal of Chinese Institute of Chemical Engineers* 2004; **35**:299–315.
41. Findeisen R, Imsland L, Allgower F, Foss BA. State and output feedback nonlinear model predictive control: an overview. *European Journal of Control* 2003; **9**:190–206.
42. Fletcher R. *Practical Methods of Optimization*. John Wiley & Sons: New York, 1987.
43. Haykin SS. *Neural Networks: A Comprehensive Foundation*. Prentice Hall: New Jersey, 1999.

44. Fanaei A, Farrokhi M. Robust adaptive neuro-fuzzy controller for hybrid position/force control of robot manipulators in unknown environment. *Journal of Intelligent and Fuzzy System* 2006; **17**:125–144.
45. Ogunfunmi T. *Adaptive Nonlinear System Identification*. Springer: New York, 2007.
46. Mayne DQ, Rawlings JB, Rao CV, Scokaert POM. Constrained model predictive control: stability and optimality. *Automatica* 2000; **36**:789–814.

SUPPORTING INFORMATION

Additional supporting information may be found in the online version of this article at the publisher's web site.