Adaptive Neuro-Predictive Position/Velocity Control of Robot Manipulators in Work Space

Mohammad Farrokhi[1] and Horiyeh Mazdarani [2]

[1] Department of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran
farrokhi@iust.ac.ir

[2] Department of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran
h_mazdarani@elec.iust.ac.ir

Abstract:

This paper proposes an adaptive Nonlinear Model Predictive Controller (NMPC) for hybrid position/velocity control of robot manipulators. Robot dynamics have generally uncertainties, including parameters variations, unknown nonlinearities of the robot, payload variations, and torque disturbances form the environment. The cost function of the NMPC is defined in such a way that by adjusting its weighting parameters, the end-effector of the robot tracks a predefined geometry path in Cartesian space with a constant velocity. Moreover, to eliminate the uncertainties, a neural network with Levenberg-Marquardt training algorithm is used to estimate adaptively the model of the robot. The closed-loop stability is demonstrated using Lyapunov theory. The validity of the proposed control method is shown by simulation results on a 3-DOF robot manipulator actuated by DC servomotors.

Keywords: Robot manipulators; Hybrid position/velocity control; Model predictive control; Neural-network model; Levenberg-Marquardt algorithm, Lyapunov stability.

1. Introduction[1]

Nowadays, robot manipulators are widely used in many tasks such as industry, medicine, military and space. One of the main reasons for using manipulators is to replace human in doing long and repetitive operations and very precise or even unhealthy tasks. Welding and laser operations are two examples of such industrial tasks, which can be performed by these robots. In particular, in many applications, robot manipulators are needed to track a predefined path with constant velocity.

Several approaches have been proposed in literatures for controlling various parameters of manipulators such as position, velocity, and force. These control schemes can be classified into two main categories: dynamic control and adaptive control. For example, a robust sliding-mode controller with PID sliding function for tracking control of a robot manipulator was proposed in [1]. Liuzzo and Tomei designed an adaptive learning PD controller, which learns the input reference signals by identifying their Fourier coefficients [2]. A saturated nonlinear PID regulator for industrial robot manipulators was proposed by Santibanez et al. [3]. And an adaptive control solution for the velocity problem of manipulators has been presented in [4].

In recent decades, intelligent algorithms such as Neural Networks (NN) and fuzzy logic have been used in the controller designs to overcome various uncertainty problems in the system. For instance, an adaptive neuro controller for trajectory tracking of robot manipulators has been developed in [5]. A robust adaptive tracking controller for a general class of strict-feedback uncertain nonlinear systems that uses a fuzzy logic to approximate the uncertainties was presented in [6]. Nagata and Watanabe presented an integrated recurrent neural network with large variability of teaching signals as feedforward controller for robot manipulators [7]. And a fuzzy logic-based cost controller for position-velocity control in nonlinear systems has been given in [8].

Over the last few decades, a new class of control approach based on the so-called Model Predictive Control (MPC) algorithm has been proposed. The MPC (also referred to as moving-horizon control or receding-horizon control) is an optimal control method that can handle the control and state constraints. In principle, MPC is a feedback control scheme, for which in each

[1] Submission date: 04, 07, 2011
Acceptance date: 30, 09, 2012
Corresponding author: Mohammad Farrokhi,
Electrical Engineering Department- Iran University of Science and Technology- Tehran- Iran

sampling period, a finite-horizon optimization problem is solved. Properties of MPC are well explained in many references such as [9--12]. However, in general, many systems are inherently nonlinear. Therefore, the class of MPC was later extended to nonlinear systems and is called Nonlinear MPC (NMPC) [10]. NMPC is especially popular in process control, where the slow system response makes the on-line optimal control computations feasible. The application of MPC in the field of robotics was limited until a decade ago. A simplified approach of predictive control of robotic manipulators that directly introduces the complete dynamic model into the cost function was presented by Duchaine et al. [13]. Another literature presents a software implementation of NMPC algorithm to control the position of a 6-DOF manipulator [14]. A new formulation of the MPC for continuous-time nonlinear systems has been developed in [15], where the proposed method allows real-time optimization technique. More examples in this regard can be found in [16--18]. Moreover, various stability results for MPC schemes of different kinds of systems have been developed by many researchers; for example see [15, 19--21].

In this paper, NMPC is used for hybrid position-velocity control of robot manipulators. In the proposed technique, a Multi-Layer Perceptron (MLP) NN with on-line Levenberg-Marquardt training algorithm is used to estimate the model of the actuated robot in the optimization procedure of the NMPC. In this way, the controller can cope with uncertainties in parameters, varying payloads, etc. Moreover, by introducing an appropriate cost function, the simultaneous position-velocity control will be feasible and easy to implement. Another advantage of the proposed method is that it can be applied in the work space of the robot mainly due to the interesting abilities of the MPC. The proposed controller also ensures stability of the closed-loop system through Lyapunov theorem. The proposed method is applied to a 3 DOF manipulator. Simulation results show effectiveness of the proposed method in hybrid position/velocity control of such systems.

This paper is organized as follows: Section 2 describes the manipulators dynamics and the NMPC algorithm. Section 3 and section 4 present the adaptive NMPC method and its stability analysis, respectively. Section 5 shows simulation results, followed by conclusion in Section 6.

2. Problem Formulation

2.1. Dynamic of Robot Manipulator

Using the Euler-Lagrange formulation, the dynamics of robot manipulators with rigid serial links can be written as [22, 23]

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \mathbf{f}(\dot{\mathbf{q}}) = \boldsymbol{\tau} \quad (1)$$

where $\mathbf{q} \in R^{n\times 1}$ is the vector of joint angles, $\mathbf{M}(\mathbf{q}) \in R^{n\times n}$ is the symmetric and positive definite inertia matrix, $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) \in R^n$ is the centrifugal and Coriolis force vector, $\mathbf{g}(\mathbf{q}) \in R^n$ is the gravity vector, $\mathbf{f}(\dot{\mathbf{q}}) \in R^n$ is the vector of joints friction of the links, $\boldsymbol{\tau} \in R^n$ is the torque vector of joints, and n is the degree of freedom. Friction for joint i can be expressed as [22]

$$f(\dot{q}) = F_v \dot{q}_i + F_d \mathrm{sgn}(\dot{q}_i) \quad (2)$$

where F_v and F_d are the coefficients of the viscous and dynamic frictions, respectively.

The dynamics of the DC servomotors, which drive the links are expressed as [22]

$$\begin{aligned} \boldsymbol{\tau}_e &= \mathbf{K}_T \mathbf{i}_a \\ \boldsymbol{\tau}_e &= \mathbf{J}_m \ddot{\mathbf{q}}_m + \mathbf{B}_m \dot{\mathbf{q}}_m + \boldsymbol{\tau}_m \\ \mathbf{V}_t &= \mathbf{R}_a \mathbf{i}_a + \mathbf{L}_a \dot{\mathbf{i}}_a + \mathbf{K}_E \dot{\mathbf{q}}_m \end{aligned} \quad (3)$$

where $\boldsymbol{\tau}_e \in R^n$ is the vector of electromagnetic torque, $\mathbf{K}_T \in R^{n\times n}$ is the diagonal matrix of the motor torque constant, $\mathbf{i}_a \in R^n$ is the vector of armature currents, $\mathbf{J}_m \in R^{n\times n}$ is the diagonal matrix of the moment inertia, $\mathbf{B}_m \in R^{n\times n}$ is the diagonal matrix of torsional damping coefficients, $\mathbf{q}_m, \dot{\mathbf{q}}_m, \ddot{\mathbf{q}}_m \in R^n$ denote the vectors of motor shaft positions, velocities and accelerations, respectively, $\boldsymbol{\tau}_m \in R^n$ is the vector of load torque, $\mathbf{v}_t \in R^n$ is the vector of armature input voltages and $\mathbf{R}_a, \mathbf{L}_a, \mathbf{K}_E \in R^{n\times n}$ are the diagonal matrix of armature resistance, armature inductance and the back electromotive force coefficients, respectively.

In order to apply the DC servomotors for actuating an *n*-link robot manipulator, the relationship between the robot joint and the motor shaft is

$$R_i = \frac{q_i}{q_{mi}} = \frac{\tau_{mi}}{\tau_i}, \quad i = 1, \dots, n \quad (4)$$

where R_i is the *i*th element of the diagonal and positive definite matrix, $\mathbf{R} = \mathrm{diag}(R_1, R_2, \dots, R_n) \in R^{n\times n}$ which in fact is the gear ratios for *n* joints. According to the fact

that the armature inductance is small and negligible, Eq. (3) can be expressed as [22]

$$\mathbf{J}_m\ddot{\mathbf{q}}_m + \left[\mathbf{B}_m + \frac{\mathbf{K}_E\mathbf{K}_T}{\mathbf{R}_a}\right]\dot{\mathbf{q}}_m + \boldsymbol{\tau}_m = \frac{\mathbf{K}_T}{\mathbf{R}_a}\mathbf{v}_t \quad (5)$$

Substituting (1) and (3) in (5), the governed equation of *n*-link robot manipulator, including actuator dynamics, can be obtained as

$$(\mathbf{J}_m + \mathbf{R}^2\mathbf{M})\ddot{\mathbf{q}} + \left(\mathbf{B}_m + \frac{\mathbf{K}_E\mathbf{K}_T}{\mathbf{R}_a}\right)\dot{\mathbf{q}} + \mathbf{R}^2(\mathbf{c}+\mathbf{g}+\mathbf{f}) = \frac{\mathbf{R}\mathbf{K}_T}{\mathbf{R}_a}\mathbf{v}_t \quad (6)$$

Based on (6), the armature input voltages are considered as the control efforts.

2.2. Model Predictive Control Algorithm

The MPC is a model-based control technique where the control action is computed by means of a receding horizon (RH) strategy, which requires the solution of an optimization problem at each sampling time. In RH strategy, the prediction of the system output is used to calculate the control law; unlike classical control schemes, which use the past outputs of the system. The MPC schemes that are based on the nonlinear model of the system or consider non-quadratic cost function and nonlinear constrains on the inputs and states, are called Nonlinear MPC (NMPC) [24]. From the theoretical point of view, the MPC algorithm can be expressed as follow.

Consider the following nonlinear state-space model:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \quad (7)$$

where $\mathbf{x}_t \in R^n$ and $\mathbf{u}_t \in R^m$ are the system state and control input, respectively. In this paper, it is assumed that function f in (7) is continuous over $R^n \times R^m$. Assume that the control objective is to regulate the system state to the origin under some input and state constraints represented by a compact set $U \subseteq R^m$ and a convex set $X \subseteq R^n$, respectively, both containing the origin. Denoting the prediction horizon and the control horizon by N_P and $N_C \leq N_P$, respectively, the cost function can be defined as

$$J(U, x_{t|t}, N_P) = \boldsymbol{\Phi}(x_{t+N_P|t}) + \sum_{j=0}^{N_P-1} L(, u_{t+j|t}) \quad (8)$$

where denotes *j*-step ahead state prediction using (7), given the input sequence $u_{t|t}, \dots, u_{t+j-1|t}$ and the initial state $x_{t|t} = x_t$. Moreover, $U = [u_{t|t}^T, \dots, u_{t+N_C-1|t}^T]^T$ is the vector of the control moves that has to be optimized. The remaining predicted control moves can be computed with different strategies, e.g. by setting. Then, the NMPC control law is obtained by applying the following RH strategy as $[u_{t|t}^* \quad u_{t+N_C-1|t}^*]$:

1. At time instant *t*, get x_t.
2. Solve the optimization problem (8) subject to *U*.
3. Apply the first element of the solution sequence *U* to the optimization problem as the actual control action $u_t = u_{t|t}$.
4. Repeat from step 1 at time $t + 1$.

3. NMPC Design for Position-Velocity Control

The purpose of the position-velocity control of robot manipulators is to obtain a control law that forces the end-effector to track a given geometry path in the Cartesian space with desired velocity. Block diagram of the proposed NMPC is shown in Fig. 1. According to the NMPC algorithm, an appropriate cost function must have direct relation with the position and velocity errors of the end-effector in comparison with desired values. These errors can be expressed by Euclidean distance between the end-effector position and velocity in Cartesian space and their references as

$$D_P = \sqrt{(x-x_r)^2 + (y-y_r)^2 + (z-z_r)^2} \quad (9)$$

$$D_V = \sqrt{(\dot{x}-\dot{x}_r)^2 + (\dot{y}-\dot{y}_r)^2 + (\dot{z}-\dot{z}_r)^2} \quad (10)$$

where, x, y, z and $\dot{x}, \dot{y}, \dot{z}$ are the end-effector position and velocity components in the Cartesian space, respectively, and x_r, y_r, z_r and $\dot{x}_r, \dot{y}_r, \dot{z}_r$ are the corresponding reference position and velocity components, respectively.

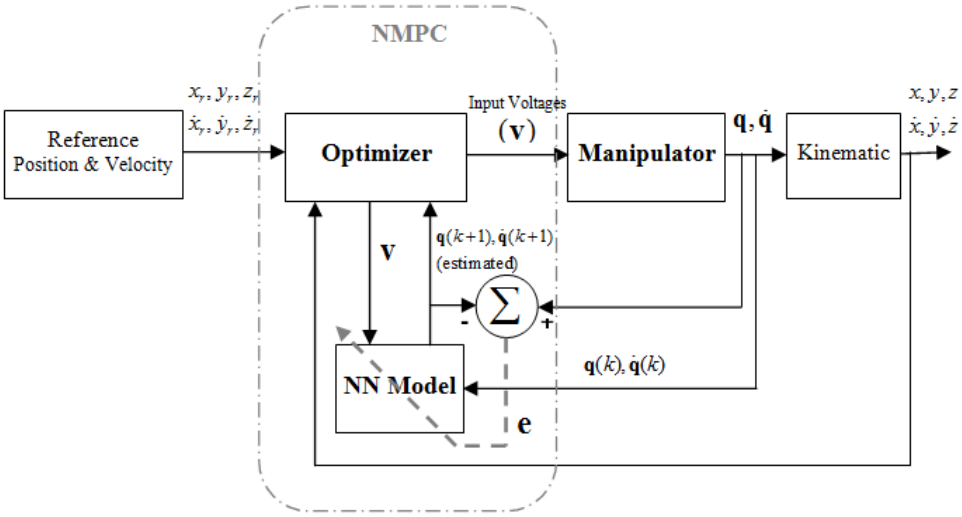


Fig. 1. Block diagram of proposed NMPC

One of the proper candidates for the cost function in the considered problem in this paper can be written as

$$J = \sum_{j=1}^{N_P} D_{P_{k+j|k}}^T D_{P_{k+j|k}} + D_{V_{k+j|k}}^T R D_{V_{k+j|k}} \qquad (11)$$

where $Q \geq 0$ and $R \geq 0$ are the weighting parameters. The notation $a(m|n)$ indicates the value of a at instant m predicted at instant n. Since the range of variations of D_P and D_V are not the same, the two terms in (11) are normalized to [0 1] using a simple map as follows:

$$\sum_{j=1}^{N_P} Q\left(\frac{e^{D_{P_{k+j|k}}} - e^{D_{P_{min}}}}{e^{D_{P_{max}}} - e^{D_{P_{min}}}}\right) + R\left(\frac{e^{D_{V_{k+j|k}}} - e^{D_{V_{min}}}}{e^{D_{V_{max}}} - e^{D_{V_{min}}}}\right) \qquad (12)$$

where $[D_{Pmin}, D_{Pmax}]$ and $[D_{Vmin}, D_{Vmax}]$ are the range of variations for D_P and D_V respectively. In the proposed control scheme, a Multi-Layer Perceptron (MLP) NN with one hidden layer is used to estimate the model of the manipulator, with actuators dynamics, for predicting the robot behavior. MLP is the most commonly used NN architecture due to its structural flexibility, good representational capabilities and availability of a large number of training algorithms [25]. For this purpose, six NNs with similar structure (as shown in Fig. 2) are used to estimate the system states (joint angels and velocities) at the next sampling time. The inputs to these NNs are all system states (three joint angles and three joint velocities) plus the corresponding control signal of that joint at the current sampling time (Fig. 2). An on-line training method is employed to adapt the NN model to uncertainties such as payload variations and parameters uncertainties. Hence, the NN does not need any off-line training. However, some mild off-line training could give the controller the benefit not to apply irrational control signals to the robot at the beginning of operation. The Levenberg-Marquardt algorithm has been used for off-line and on-line training. This is a numerical optimization technique that has been designed for minimizing functions that are sums of squares of other nonlinear functions; hence, it is well suited for training NNs such as MLP, where the performance index is the sum of squared errors. Moreover, this algorithm provides a balanced compromise between the speed of the Newton's method and the guaranteed convergence of the steepest descent scheme [25]. Appendix of this paper provides a brief review of the Levenberg-Marquardt training algorithm. The NNs are trained off-line using data obtained from the approximate robot dynamics via input voltages as

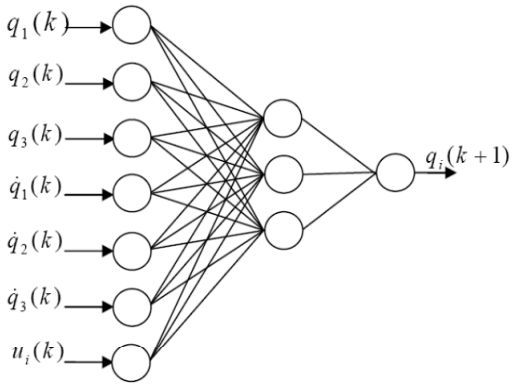


Fig. 2. The NN model for estimating the *i*th joint angle

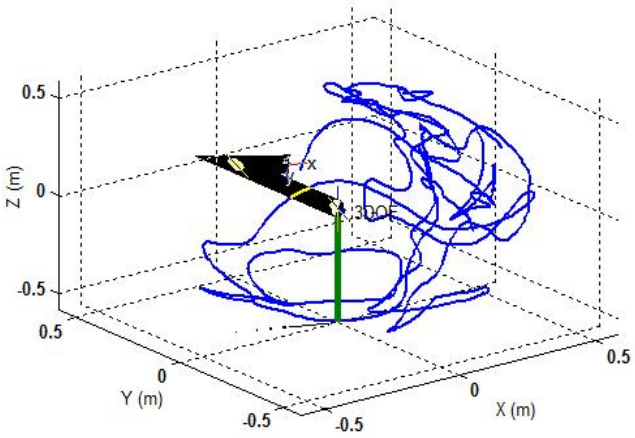


Fig. 3. The end-effector motion with input voltages as Eq. (13)

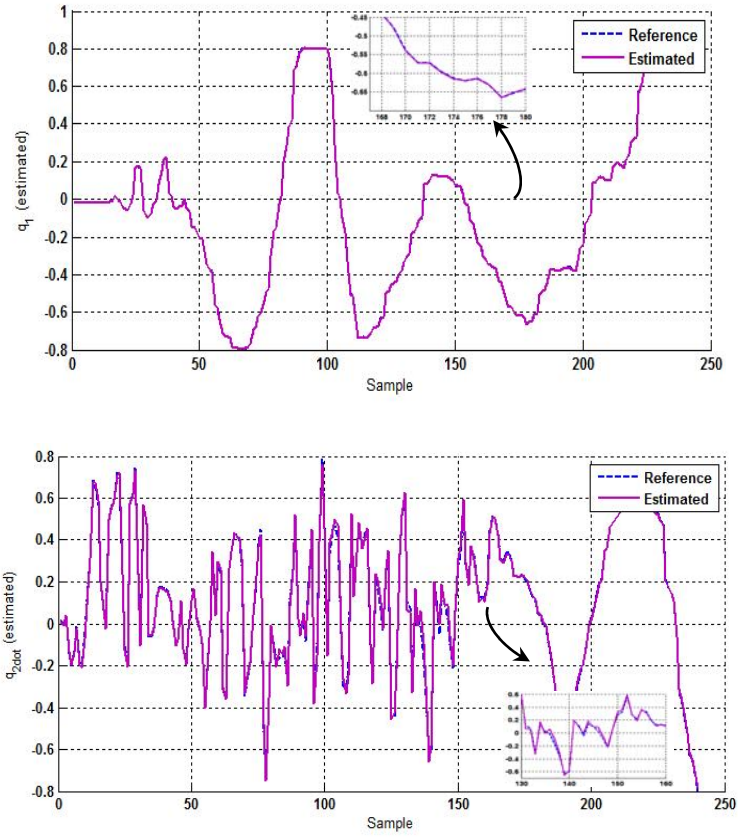


Fig. 4. Test results of two NNs

Table 1. NNs Mean-Squared errors

NNs / Errors	NN 1 (q_1)	NN 2 (q_2)	NN 3 (q_3)	NN 4 $(\dot{q}_1)$	NN 5 $(\dot{q}_2)$	NN 6 $(\dot{q}_3)$
Mean-Squared errors of NNs train	0.0026	0.0021	0.0181	0.1106	0.2115	0.252
Mean-Squared errors of NNs test	0.0005	0.0004	0.0104	0.0242	0.0624	0.1023

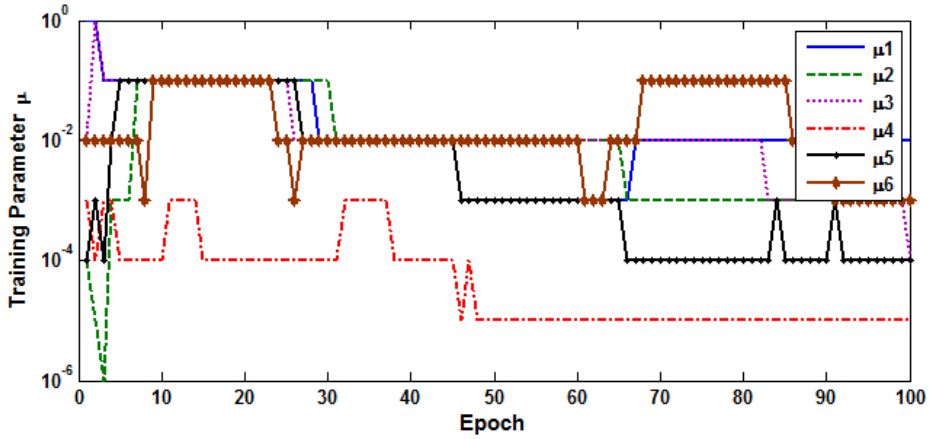


Fig. 5. Training parameter variations of the NNs

$$
\begin{aligned}
&u_1(t)\\
&= 16e^{\frac{-(t-30)^2}{150}}\cos(0.6t)\\
&+ 10f(t,[30\ 90\ 30\ 120])\sin(0.09t)\\
&+ 3f(t,[30\ 160\ 20\ 200])(\sin(1.5t)+0.08)\\
&- 2.5f(t,[10\ 180\ 10\ 210])\\
&+ 14e^{0.03t-9}(\sin(0.3t)+1.5\sin(t)+\sin(3t)\\
&+ 0.8)
\end{aligned}
$$

$$
\begin{aligned}
&u_2(t)\\
&= 15e^{\frac{-(t-20)^2}{160}}(\sin(t)+0.6)\\
&+ 8.5f(t,[30\ 90\ 30\ 80])(\sin(0.8t)+0.05)\\
&+ f(t,[10\ 160\ 10\ 190])\\
&+ 5f(t,[30\ 130\ 20\ 170])(\sin(0.5t)\\
&+ 1.5\sin(t)+\sin(3t)+0.6)\\
&+ 12e^{0.01t-3}(\sin(0.1t)+0.2)
\end{aligned}
$$

$$
\begin{aligned}
&u_3(t)\\
&= 6e^{\frac{-(t-15)^2}{145}}(\sin(0.5t)+1.5\sin(t)+\sin(3t)\\
&+ 0.49)\\
&+ 2f(t,[10\ 60\ 20\ 120])\ (\sin(0.5t)-0.35)\\
&+ 3f(t,[20\ 130\ 20\ 160])(\sin(0.4t)+0.8)\\
&+ 11e^{0.05t-15}(\cos(0.5t)+0.4)\\
&+ 3.5f(t,[10\ 190\ 10\ 220])(\sin(t)-0.3)
\end{aligned}
$$

$$f(t,[s_1\ c_1\ s_2\ c_2]) = \text{gauss2mf}(t,[s_1\ c_1\ s_2\ c_2]) \quad (13)$$

where gauss2mf is a MATLAB function that generates a semi-Gaussian signal. Using these inputs, the end-effector passes various points in the task space as shown in Fig. 3. The NNs mean squared errors in two cases (train and test) are given in Table 1. Moreover, Fig. 5 shows the test results of two sample networks. The training parameter μ (see Appendix) of the Levenberg-Marquardt changes through the learning procedure by factor υ. Fig. 5 shows the values of this parameter of six NNs, where $\upsilon = 10$ and $\mu(0) = 0.001$.

Constraints in the optimization problem can be considered as follows. One of these constraints is the limitation of the amplitude of input voltages; the other constraint is the limitation of the joint velocities, which is considered based on the fact that in a singular configuration, for the case of limited velocity for the end-effector, the joint velocities are infinite. Therefore, the following constraints must be taken into account:

$$\begin{aligned}\mathbf{v}_{\min} \le \mathbf{v} \le \mathbf{v}_{\max}\\ \dot{\mathbf{q}}_{\min} \le \dot{\mathbf{q}} \le \dot{\mathbf{q}}_{\max}\end{aligned} \quad (14)$$

where $\mathbf{v}_{\min}$ and $\mathbf{v}_{\max}$ are the lower and the upper bound of the joints voltages, and $\dot{\mathbf{q}}_{\min}$ and $\dot{\mathbf{q}}_{\max}$ are the lower and upper bound of the joints velocities, respectively.

For better performance of the robot, the weighting parameters Q and R can be adjusted adaptively as follows. When the initial position of the end-effector is far from the desired path, the velocity control can be ignored since the end-effector must move quickly to the desired path. On the other hand, when the end-effector is at or near the desired path, the velocity control plays an important role in the control action. Therefore, weighting parameters are adjusting on-line by referring to the position and velocity errors. That is, when the position error is very large, the Q must be increased and R must be decreased. Conversely, when the position error is small, Q must be decreased and R must be increased.

4. Stability Analysis

The stability analysis for the proposed finite horizon NMPC is given in this section.

The governed equation of an n-link robot

manipulator, including actuator dynamics, can be shown generally as a nonlinear discrete system as

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \quad (15)$$

where $\mathbf{x}(k) \in R^{6\times1}$ is the vector of system states, $\mathbf{u}(k) \in R^{3\times1}$is the vector of DC motors armature input voltages, and $\mathbf{f}$is a nonlinear continuous function which has equilibrium at the origin $(\mathbf{f}(\mathbf{0},\mathbf{0}) = \mathbf{0})$.

According to the manipulator physical restrictions, the system constraints could be shown as follows:

$$\begin{aligned} |x_i| &\le \bar{x}_i \quad i = 1,2,\dots,6 \\ |u_i| &\le \bar{u}_i \quad i = 1,2,3 \end{aligned} \quad (16)$$

where x_i and u_i are the maximum allowed value for the states and inputs, respectively. Hence, the allowable region for the inputs is defined as

$v \triangleq \{\mathbf{u} = [u_1, u_1, u_1]^T : |u_i| \le \bar{u}_i, i = 1,2,3\}$.

Two observations can be made for the optimization problem:

- Since the value of $\mathbf{x}(k)$ is measured at time instant k, it has no influence on the solution of the on-line optimization and only affects the optimal value of the cost function.
- For the optimization problem, it can be shown that the influence of the control effort on the last stage of the prediction horizon is not taken into account in the cost function and so its optimal value is achieved at 0.

Based on these two observations, the cost function of the MPC problem (Eq. (12)) can be rewritten as:

$$J(\mathbf{x}(k), \mathbf{U}(k|k)) = \sum_{j=0}^{N_P-1} L(\mathbf{x}(k+j+1|k), \mathbf{u}(k+j|k)) \quad (17)$$

where $\mathbf{x}(k+j|k); j = 1,\dots,N_P$ denotes the predicted state at time instant $k+j$ based on the state measurements at time instant k; i.e., $\mathbf{x}(k)$ and $\mathbf{U}(k|k) = [\mathbf{u}(k|k),\dots,\mathbf{u}(k+N_P-1|k)]$ denotes the control sequence achieved at time instant k. Some assumptions should be made for this definition of the cost function:

- The nonlinear function $L(.)$ is positive definite $(L(\mathbf{x},\mathbf{u}) \ge 0)$.
- The cost function is continuously differentiable and radically bounded.

When the NMPC algorithm is applied to control the system in (15) at time instant k and subject to (16), the minimization problem is achieved as

$$J^*\big(\mathbf{x}^*(k), \mathbf{U}^*(k|k)\big) = \min_{\mathbf{u}(k|k),\dots,\mathbf{u}(k+N_P-1|k)} J(\mathbf{x}(k), \mathbf{U}(k|k)) \quad (18)$$

By solving this problem on-line, the optimal values of the control sequences, state trajectories, and cost function are obtained as

$$\mathbf{U}^*(k|k) = [\mathbf{u}^*(k|k),\dots,\mathbf{u}^*(k+N_P-1|k)] \quad (19)$$

$$\mathbf{X}^*(k|k) = [\mathbf{x}^*(k+1|k),\dots,\mathbf{x}^*(k+N_P|k)] \quad (20)$$

$$J^*(\mathbf{x}^*(k), \mathbf{U}^*(k|k)) = \sum_{j=0}^{N_P-1} L(\mathbf{x}^*(k+j+1|k), \mathbf{u}^*(k+j|k)) \quad (21)$$

Since there are constraints for the control as well as state variables, it is unlikely to achieve global stability and feasibility for the NMPC problem. Hence, the feasibility region in this paper is defined in the followings.

Definition 1: a feasibility region χ for the MPC problem (15) with performance index (17) is defined as a set such that for any $\mathbf{x}(k) \in \chi$ there exist control $\mathbf{u}(k) \in v$ such that

$$\begin{cases} \mathbf{x}(k+1) \in \chi \\ L\big(\mathbf{x}(k+2|k), \mathbf{u}(k+1|k)\big) \\ -L(\mathbf{x}(k+1|k), \mathbf{u}(k|k)) \le 0 \end{cases} \quad (22)$$

Remark 1: Condition 2 in the above definition is quite reasonable assumption; as otherwise, the overall cost would increase no matter what control action is taken. In other words, this condition says that the value of the cost function at the end of the minimization procedure should not be larger than its value at the beginning, which is inevitable.

Definition 2: The NMPC problem is feasible at time instant k if the following conditions hold:

$$\forall \mathbf{x}^*(k|k) = \mathbf{X}(k) \in: \mathbf{X}^*(k|k) = [\mathbf{x}^*(k+1|k),\dots,\mathbf{x}^*(k+N_P|k)] \in \chi \quad (23)$$

Theorem 1: consider the NMPC problem for the system (15) and (16) with performance index (17). The closed-loop system under the NMPC law stemming from the on-line optimization is stable about the origin with the stability region χ if the NMPC problem is feasible as defined in definition 2.

Proof: according to the feasibility of the MPC problem as defined in definition 2 and

considering the cost function, one can easily conclude that the optimal cost function is also positive definite and continuously differentiable. Hence, it can be used as Lyapunov candidate

$$V\big(\mathbf{x}(k)\big) = J^*\big(\mathbf{x}^*(k), \mathbf{U}^*(k|k)\big) \tag{24}$$

Then,

$$\Delta V\big(\mathbf{x}(k)\big) = V(\mathbf{x}(k+1)) - V\big(\mathbf{x}(k)\big) \tag{25}$$

where

$$\begin{aligned} V\big(\mathbf{x}(k+1)\big) &= J^*\big(\mathbf{x}^*(k+1), \mathbf{U}^*(k+1|k+1)\big) \\ &= \min_{\mathbf{U}(k+1|k+1)} \sum_{j=0}^{N_P-1} L(\mathbf{x}(k+j \\ &\quad + 2|k+1), \mathbf{u}(k+j+1|k \\ &\quad + 1)) \end{aligned} \tag{26}$$

$$\begin{aligned} \Delta V\big(\mathbf{x}(k)\big) = \sum_{j=0}^{N_P-1} \{ & L(\mathbf{x}^*\big(k+j \\ & + 2|k \\ & + 1), \mathbf{u}^*(k+j+1|k+1)\big) \\ & - L(\mathbf{x}^*\big(k+j \\ & + \ \ 1|k), \mathbf{u}^*(k+j|k)\big)\} \end{aligned} \tag{27}$$

At time instant k, after on-line solving the MPC problem (18), the optimal control sequence within the prediction horizon is given as $\mathbf{U}^*(k|k) = [, \dots, \mathbf{u}^*(k+N_P-1|k), \mathbf{u}^*(k+N_P|k)]$ where $\mathbf{u}^*(k+N_P|k) = 0$.

At time instant $k+1$, after applying the optimal control $\mathbf{u}^*(k|k)$, one has

$$\mathbf{x}(k+1) = \mathbf{x}^*(k+1|k) \tag{28}$$

The new control sequence $\tilde{\mathbf{U}}(k+1|k+1)$ can be selected as

$$\begin{aligned} &\tilde{\mathbf{U}}(k+1|k+1) \\ &= [\tilde{\mathbf{u}}(k+1|k+1), \dots, \tilde{\mathbf{u}}(k+|k+1)] \end{aligned} \tag{29}$$

Where $\tilde{\mathbf{u}}(k+i|k+1) = \mathbf{u}^*(k+i|k); i = 1, \dots, N_P$.

The corresponding state trajectory, based on $\mathbf{x}(k+1)$ as the initial state, is given by

$$\begin{aligned} &\tilde{\mathbf{X}}(k+1|k+1) \\ &= \ [\tilde{\mathbf{x}}(k+2|k+1), \dots, \tilde{\mathbf{x}}(k+N_P+1|k+1)] \end{aligned} \tag{30}$$

Therefore, the cost function corresponding to $\tilde{\mathbf{U}}(k+1|k+1)$ and $\tilde{\mathbf{X}}(k+1|k+1)$ can be expressed as

$$\begin{aligned} &\tilde{J}\left(\tilde{\mathbf{x}}(k+1), \tilde{\mathbf{U}}(k+1|k+1)\right) \\ &\quad = \sum_{j=0}^{N_P-1} L(\tilde{\mathbf{x}}\big(k+j \\ &\quad + 2|k \\ &\quad + 1), \tilde{\mathbf{u}}(k+j+1|k+1)\big) \end{aligned} \tag{31}$$

By comparing (26) and (31), one can easily conclude that

$$V\big(\mathbf{x}(k+1)\big) \le \tilde{J}\left(\tilde{\mathbf{x}}(k+1), \tilde{\mathbf{U}}(k+1|k+1)\right) \tag{32}$$

Hence,

$$\begin{aligned} \Delta V\big(\mathbf{x}(k)\big) \le \ & \tilde{J}\left(\tilde{\mathbf{x}}(k+1), \tilde{\mathbf{U}}(k+1|k+1)\right) \\ & - V(\mathbf{x}(k)) \end{aligned} \tag{33}$$

Moreover, it follows from (15), (29) and (30) that

$$\begin{cases} \tilde{\mathbf{x}}(k+i|k+1) \\ \quad = \mathbf{x}^*(k+i|k); i = 2, \dots, N_P \\ \tilde{\mathbf{x}}(k+N_P+1|k+1) \\ = \mathbf{f}(\mathbf{x}^*(k+N_P|k), \mathbf{u}^*(k+N_P|k)) \end{cases} \tag{34}$$

Substituting (34) into (33) gives

$$\begin{aligned} &\Delta V\big(\mathbf{x}(k)\big) \\ &= \sum_{j=0}^{N_P-1} \{L(\tilde{\mathbf{x}}\big(k+j \\ &+ 2|k+ \ \ \ 1), \tilde{\mathbf{u}}(k+j+1|k+1)\big)\} \\ &- \sum_{j=0}^{N_P-1} \{L(\mathbf{x}^*\big(k+ \ \ \ j+1|k), \mathbf{u}^*(k+j|k)\big)\} \end{aligned} \tag{35}$$

This can be written as

$$\begin{aligned} &\Delta V\big(\mathbf{x}(k)\big) \\ &\le \ \{L\big(\tilde{\mathbf{x}}(k+N_P+1|k+1), \tilde{\mathbf{u}}(k+N_P|k+1)\big)\} \\ &+ \ \sum_{j=0}^{N_P-2} \{L(\tilde{\mathbf{x}}\big(k+j \\ &+ \ \ 2|k+1), \tilde{\mathbf{u}}(k+j+1|k+1)\big)\} \\ &- \ \ \sum_{j=1}^{N_P-1} \{L(\mathbf{x}^*\big(k+j+1|k), \mathbf{u}^*(k+j|k)\big) \\ &- \ \ \ L(\mathbf{x}^*\big(k+1|k), \mathbf{u}^*(k|k)\big)\} \end{aligned} \tag{36}$$

Substituting (29) and (34) into (36) yields

$$\begin{aligned} &\Delta V\big(\mathbf{x}(k)\big) \\ &= \ \{L\big(\tilde{\mathbf{x}}(k+N_P+1|k+1), \tilde{\mathbf{u}}(k+N_P|k+1)\big)\} \\ &- L(\mathbf{x}^*\big(k+1|k), \mathbf{u}^*(k|k)\big) \end{aligned} \tag{37}$$

Hence,

$$\begin{aligned} &\Delta V\big(\mathbf{x}(k)\big) \\ &\le L\{\mathbf{f}\big(\mathbf{x}^*(k+N_P|k), \mathbf{u}^*(k+N_P|k)\big), \mathbf{u}^*(k+N_P|k)\} \\ &- \ \ L(\mathbf{x}^*\big(k+1|k), \mathbf{u}^*(k|k)\big) \end{aligned} \tag{38}$$

Finally,

$$\Delta V\big(\mathbf{x}(k)\big) \le L_{N_P} - L_0 \tag{39}$$

where L_{N_P} and L_0 are the nonlinear function $L(.)$at the beginning of the optimization procedure and at the last stage of the prediction horizon, respectively. Therefore, by considering

the feasibility property of the MPC problem given in Definition 2, and by referring to Remark 1, it follows that

$$\Delta V\big(\mathbf{x}(k)\big) \le 0 \quad (40)$$

Therefore, $\Delta V\big(\mathbf{x}(k)\big)$ is negative inside the feasibility region χ and according to the Lyapunov theory, the closed-loop system under the MPC is stable. ❑

Table 2. Denavit-Hartenberg parameters

Joint	α	a	d	$\boldsymbol{\theta}°$
1	90	0	0	θ_1
2	0	l_1	0	θ_2
3	0	l_2	0	θ_3

Table 3. Manipulator links parameters

Link	1	2	3
l (m)	0.1	0.418	0.165
m (kg)	12.6	10.1	2.4

Table 4. DC servomotors parameters

Motor	R_a	K_E	K_T	B_m	J_m	R	V_t
1	5.3	0.28	0.48	64×10^{-4}	0.51	1:100	12
2	5.3	0.28	0.48	64×10^{-4}	0.51	1:100	12
3	5.3	0.28	0.48	64×10^{-4}	0.51	1:10	12

6. Simulation Results

A 3-DOF (waist, shoulder and elbow) type manipulator actuated by DC servomotors is used for simulations. Using the Denavit-Hartenberg parameters (Table 2), the position of the end-effector in Cartesian space can be calculated. Other parameters of the robot and DC servomotors are given as Tables 3 and 4, respectively. The constraints on the motor voltages and joint velocities are selected as [12, 12] V and [35, 35] deg/s, respectively. Moreover, the following adaptation rules are employed for adjusting the weighting parameters in the const function

$$\begin{cases} IF\ D_P \ge 1.4\times10^{-3}, THEN\ \ Q=30, R=0 \\ IF\ D_P < 1.4\times10^{-3}, THEN\ \ Q=0, R=30 \end{cases} \quad (41)$$

As discussed before, the model of the robot with the servomotors dynamics is estimated by MLPNN during the on-line optimization. For this purpose, six NNs with similar structure (one hidden layer with three neurons) are used to estimate the system states at the next sampling time. All NNs have tangent hyperbolic nonlinear function in their hidden layer; they are trained off-line using data obtained from the approximate robot dynamics. The learning parameters of NNs are fixed at $\mu = 20$ for the online training. For the sake of comparison, computed torque controller is used to compare the results with the proposed method. This controller is adjusted to have critical damping response by selecting the proportional and derivative coefficients as $K_P = 12$ and $K_V = 7$.

The proposed controller is applied for two cases:

1) For the first case, controller parameters are selected as $N_P = 5, N_C = 2$. There is no payload in this case and the sampling time is 0.2 sec. The desired trajectory for this case is a rectangular path in the Cartesian space. The end-effector must follow this path with the constant velocity of 1 cm/sec. The reason this path has been selected is that controlling the end-effector at sharp 90 degrees angles is challenging for most controllers. Figs. 6-9 show the simulation results. As these figures show, the manipulator end-effector can follow the desired path and the velocity with good accuracy. Moreover, the motor voltages and the angular velocities are within the predefined bounds. Furthermore, Fig. 10 shows the results of the computed torque method in comparison with the proposed method. As this figure shows, the end-effector has followed the desired path with high precision by the computed torque controller (because it uses the exact robot dynamic in controller computation). However, at the corner, the end-effector deviates slightly from the path and then returns to it. This behavior caused from the fact that the computed torque controller has no information about the future of the path and makes the end-effector to produce some overshoots. In addition, the proposed controller leads the end-effector toward the desired path from its initial state faster than the computed torque method.

One major concern among researchers in using the MPC methods is the computational time, especially for the optimization part of the controller. As Fig. 11 shows, the computation time is less than the data sampling time. Hence, the proposed controller can be used in real-time

applications.

The stability of the proposed controller was shown in Section 4 by selecting the optimal cost function as the Lyapunov candidate. For better visualization of the stability problem, the value of the Lyapunov function (Eq. (10)) and its upper bound (Eq. (39)) are plotted from the beginning of the manipulator motion until the moment it passes the first corner of the path (Figs. 12 and 13). As discussed before, the weighting parameters are adjusting on-line by considering the position error. Fig. 12 shows how these parameters vary during the path. At the beginning, when the end-effector position is far from the desired path, the velocity control is ignored and the end-effector moves quickly to the desired path. When the end-effector is near the desired path, the velocity control plays an important role in the control action. At $t = 9.4$ sec, when the end-effector approaches 90^o angle, the position error increases. Hence, the weighting parameters in the cost function change in order to bring the end-effector quickly to the desired path. Fig. 13 shows $\Delta V(\mathbf{x}(k))$; i.e., the difference of the Lyapunov function. As this figure shows, $\Delta V(\mathbf{x}(k))$ is mostly negative except when the end-effector passes through the corner of the rectangular path, where it has a small positive value for a few samples. Fig. 14 shows how the end-effector passes through the 90^o angle (i.e., from $t = 9.4$ sec to $t = 9.4$ sec, which is equivalent to the pointed area in Figs. 12 and 13). This is mainly due to sudden changes in weighting parameters R and Q in the cost function. However, the proposed algorithm is able to bring the closed-loop system quickly to the stable region.

2) In the second case, an 8 Kg payload is added to the end-effector at $t = 25$ sec. The desired trajectory for this case is a 3D circle in the Cartesian space and the desired velocity is defined as 2 cm/sec. Due to the smoothness of the desired trajectory in this case, the controller parameters are selected as $N_P = 4, N_C = 2$. Other control parameters are the same as before. The simulation results are shown in Figs. 15-18. As these figures show, the payload variation causes a disturbance to the robot operation. However, the adaptation capability of the NN can cope with it very quickly and return the end-effector to its desired path, yielding an adaptive and robust control method. Fig. 19 shows three weights of one of the NNs, where the weights variations due to the payload of the robot can be observed. On the other hand, as Fig. 20 shows, the computed torque controller is week in this case, as when the payload is added this controller cannot follow the desired path. A comparison between the sum-of-squared position errors of two controllers is provided in Fig. 21. Table 4 shows a numerical comparison between these two control methods.

7. Conclusion

This paper presented an approach for the hybrid position/velocity control of robot manipulators. The proposed control scheme is independent of the robot dynamics and is robust against any changes in the system parameters. To this end, a NN with on-line training capability has been employed. While this NN can provide appropriate model of the robot, it can deal with any changes in robot parameters. Moreover, by on-line adjustment of the weighting parameters in the cost function of the NMPC, one can obtain a trade-off between the position and velocity control. Another important feature of the proposed control scheme is its ability for real-time applications, due to the relatively low computational time in each sample. Also, the stability of the close-loop system has been proven through Lyapunov theory. Simulation results on a 3-DOF actuated manipulator demonstrated the effectiveness of the proposed method.

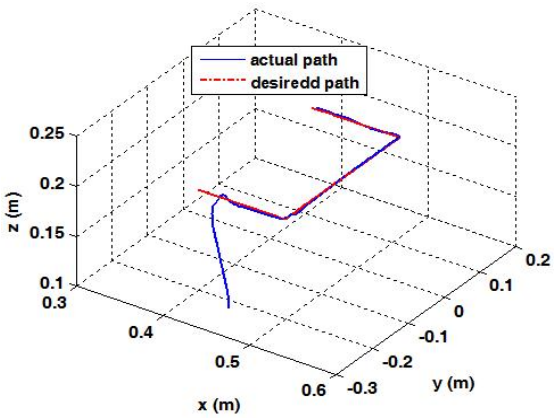


Fig. 6. Desired and actual end-effector path in workspace

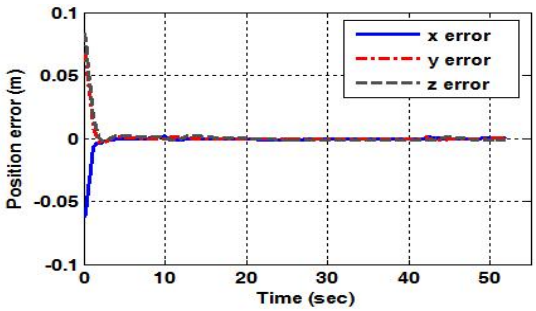


Fig. 7. Position errors in Cartesian space

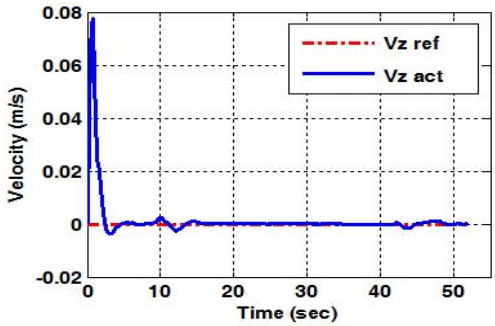
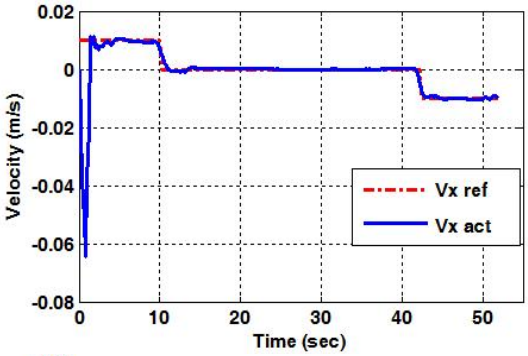
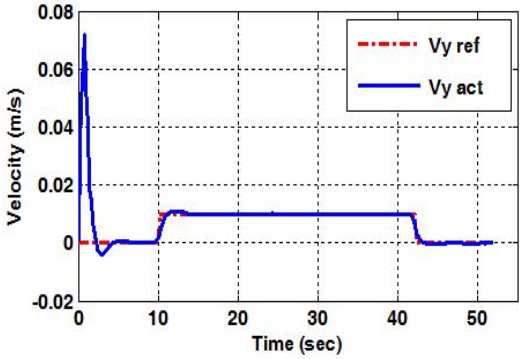
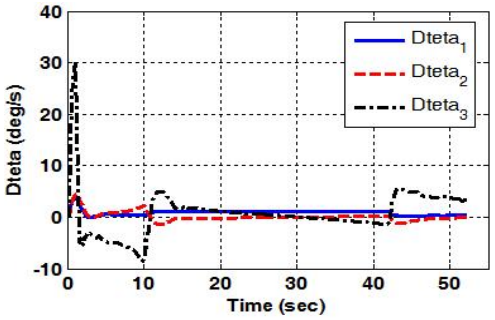


Fig. 8. End-effector velocities in Cartesian space







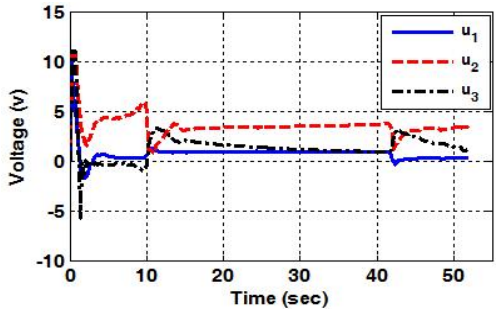


Fig. 9. Joint velocities and motor voltages

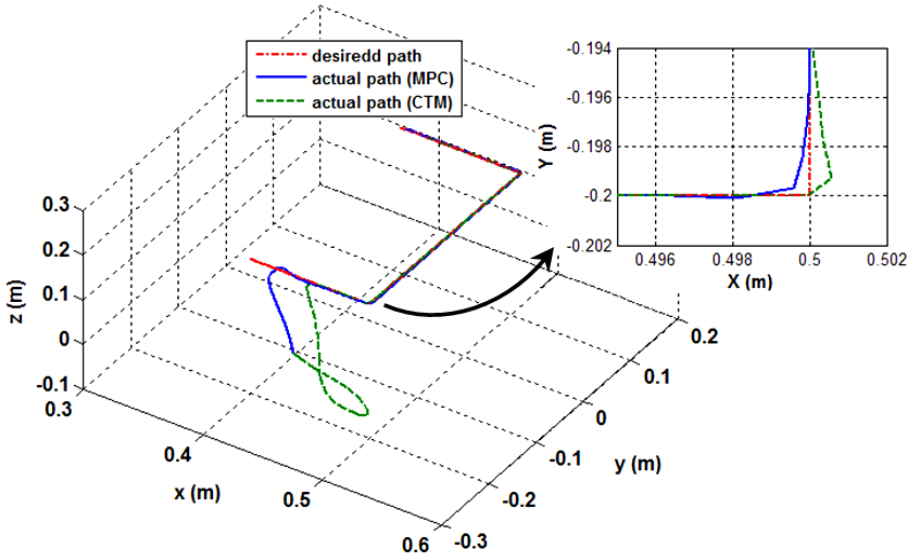


Fig. 10. Desired and actual end-effector path in workspace (solid line: MPC, dashed line: Computed Torque Method)

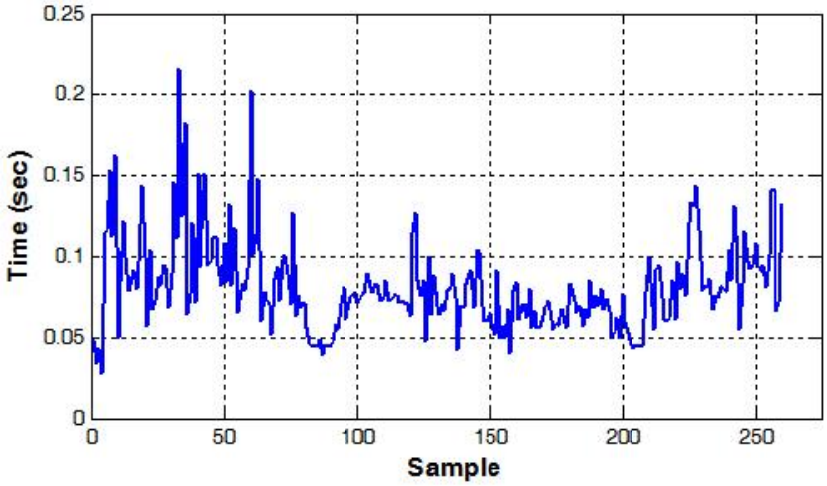


Fig. 11. Computational time during each sample

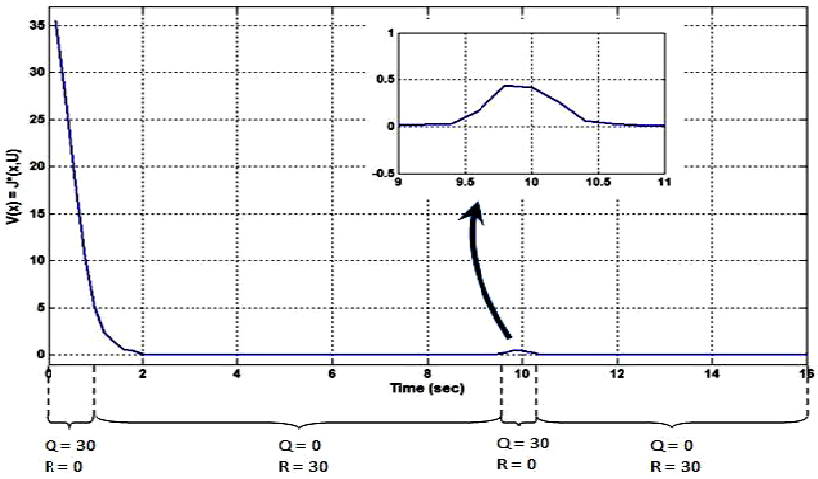


Fig. 12. Value of Lyapunov function

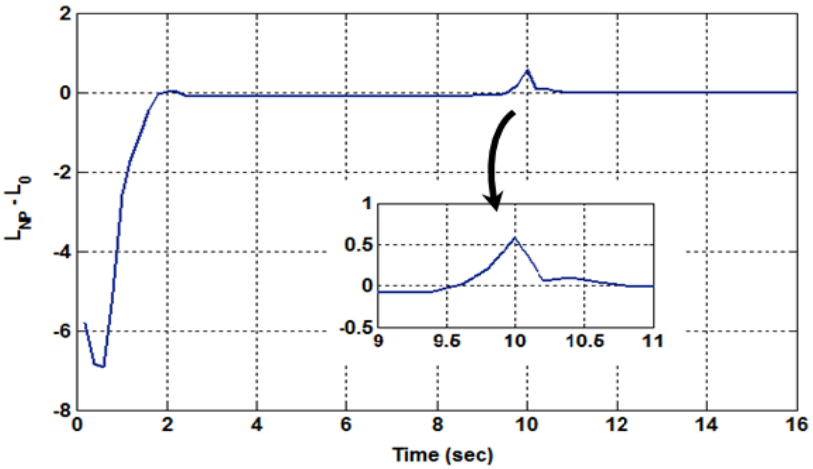


Fig. 13. Upper bound of Lyapunov function

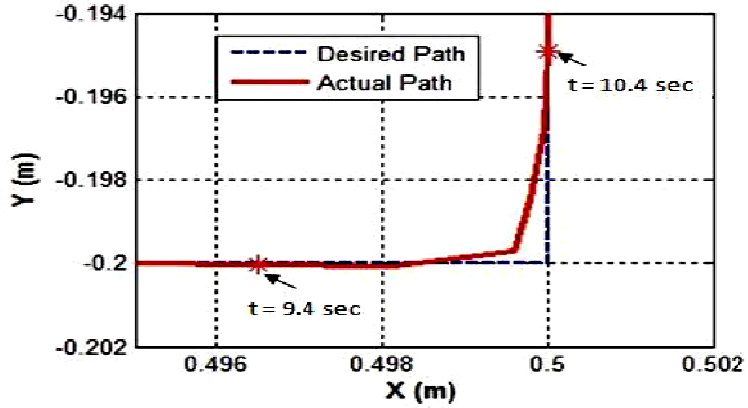


Fig. 14. Desired and actual end-effector path in x-y plane at the corner of the rectangular path

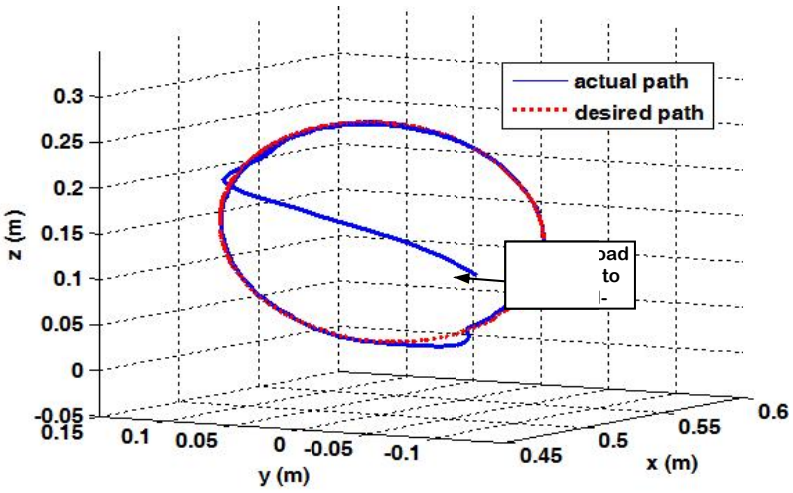


Fig. 15. Desired and actual end-effector path in workspace

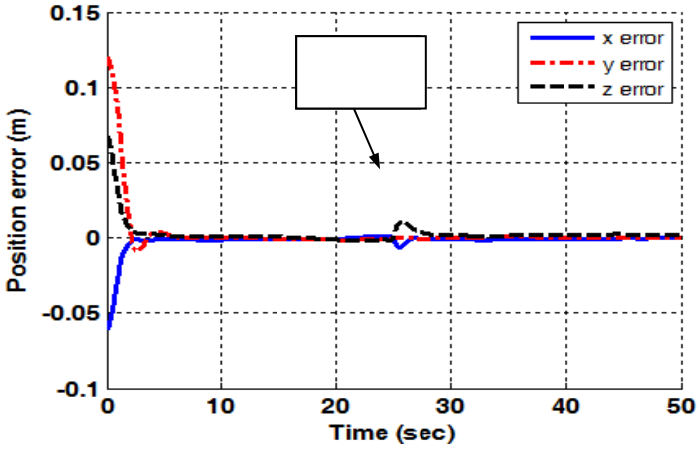


Fig. 16. Position errors in Cartesian space

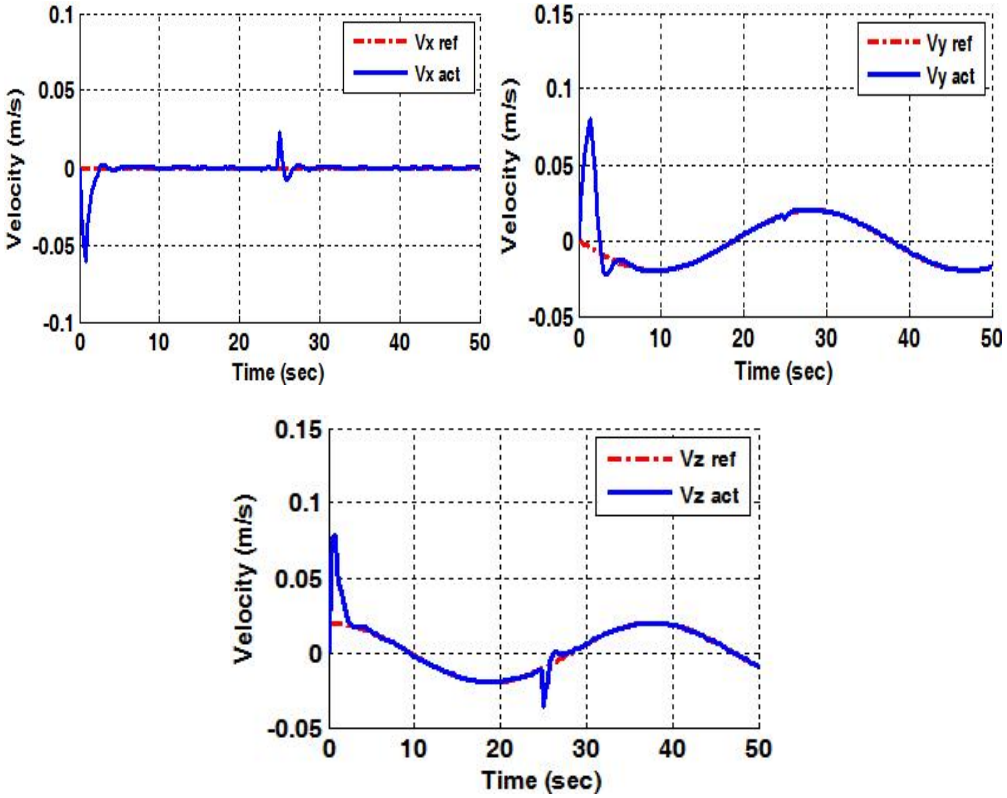


Fig. 17. End-effector velocities in Cartesian space

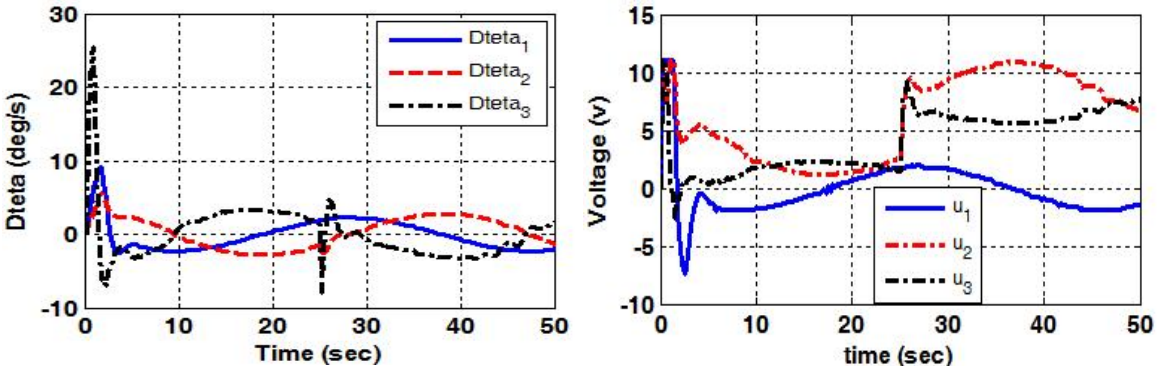


Fig. 18. Joint velocities and motor voltages

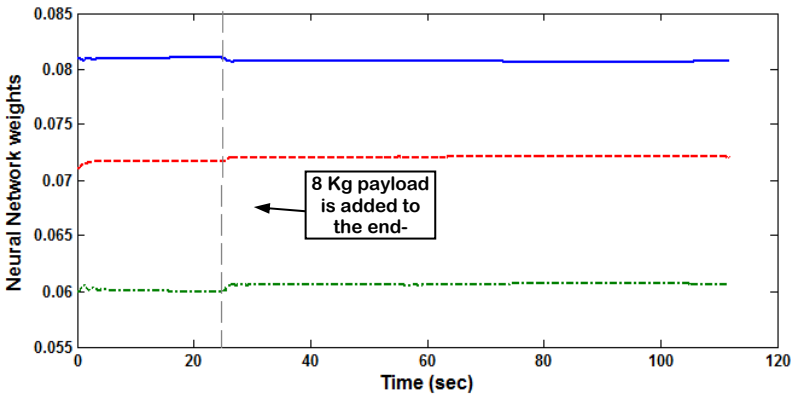


Fig. 19. three weights of the NN which estimates the third joint angle

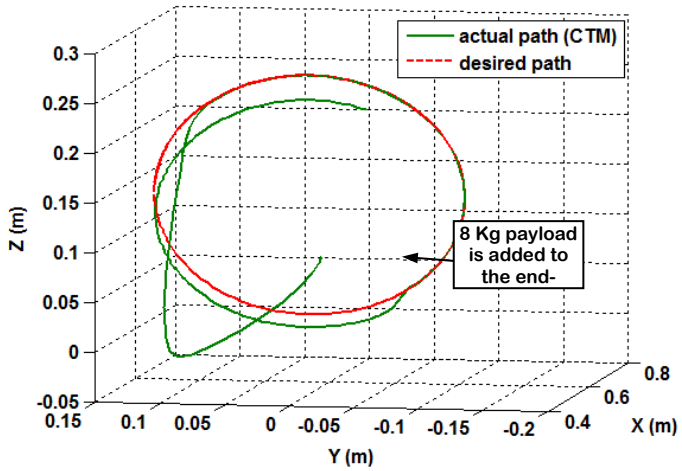


Fig. 20. Desired and actual end-effector path in workspace by computed torque method

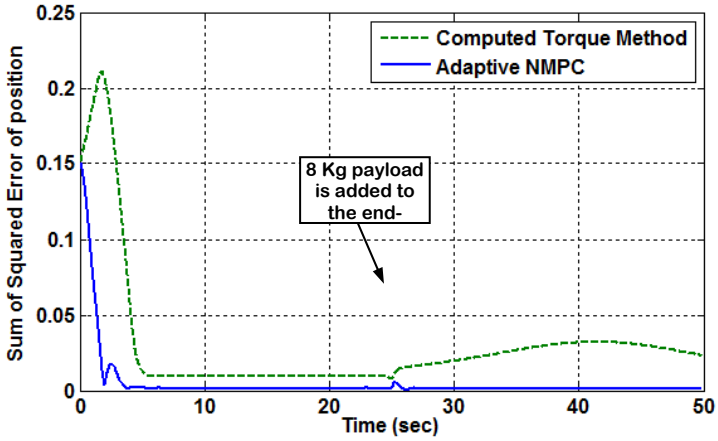


Fig. 21. Error tracking of two controllers

Table 5. Comparison between two control algorithms

Performance Factors		Adaptive Neuro-Predictive Controller	Computed Torque Method
Rectangular path (no payload was added)	Mean of squared errors after reaching the path	0.0016	0.0165
	Transition time (before reaching the path)	3.4	5.6
Circular path (8 Kg payload was added to the end-effector at t = 25 sec)	Mean of squared errors after reaching the path	0.0018	0.0248
	Transition time (before reaching the path)	4 sec	6 sec

References

[1] T. C. Kuo, Y. J. Huang, "A Sliding mode PID-controller design for robot manipulators", Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation Espoo, Finland; pp. 625-629, 2005.

[2] S. Liuzzo, P. A. Tomei, "A global adaptive learning control for robotic manipulators", Automatica, Vol. 44, No. 5, pp. 1379-1384, 2008.

[3] V. Santibanez, K. Camarillo, J. Moreno-Valenzuela, R. Campa, "A practical PID regulator with bounded torques for robot manipulators. International Journal of Control, Automation, and Systems", Vol. 8, No. 3, pp. 544-555, 2010.

[4] D. Seo, M. R. Akella, "Non-certainty equivalent adaptive control for robot manipulator systems",

System and Control Letters, Vo. 58, No. 4, pp. 304-308, 2009.
[5] S. Purwar, I. N. Kar, A. N. Jha, "Adaptive output feedback tracking control of robot manipulators using position measurements only", Expert Systems with Applications , Vol. 34, No. 4, pp. 2789–2798, 2008.
[6] Y. Yang, G. Feng, J. Ren, "A combined backstepping and small-gain approach to robust adaptive fuzzy control for strict-feedback nonlinear systems", IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans Vo. 34, No. 3, pp. 406-420, 2004.
[7] F. Nagata, K. Watanabe, "Adaptive learning with large variability of teaching signals for neural networks and its application to motion control of an industrial robot", International Journal of Automation and Computing, Vol. 8, No. 1, pp. 54-61, 2011.
[8] D. Senthilkumar, C. Mahanta, "Fuzzy guaranteed cost controller for trajectory tracking in nonlinear systems", Nonlinear Analysis: Hybrid Systems, Vol. 3 , No. 4, pp. 368-379, 2009.
[9] E. F. Camacho, C. Bordons, Model Predictive Control in the Process Industry. Springer Verlage: New York, 1995.
[10] R. Findeisen, F. Allgower, An Introduction to Nonlinear Model Predictive Control, Technical Report, University of Stuttgart: Germany, 2002.
[11] F. Allgower, R. Findeisen, Z. K. Nagy, "Nonlinear model predictive control: from theory to application", Journal of Chinese Institute of Chemical Engineers, Vol. 35, No. 3, pp. 299-315, 2004.
[12] B. Kouvaritakis, M. Cannon, "Nonlinear predictive control: Theory and practice. IEE control Engineering Series Vol. 61, 2001.
[13] V. Duchaine, S. Bouchard, C. M. Gosselin, "Computationally efficient predictive robot control", IEEE/ASME Transactions on Mechatronics, Vol. 12, No. 5, pp. 570-578, 2007.
[14] W. Wroblewski, "Implementation of a model predictive control algorithm for a 6dof manipulator - simulation results," 4th International Workshop on Robot Motion and Control, Puszczykowo, Poland, pp. 209-212, 2004.
[15] D. Dehaan, M. Guay, "A real-time framework for model-predictive control of continuous-time nonlinear systems", IEEE Transactions on Automatic Control, Vol. 52, No. 11, pp. 2047-2057, 2007.
[16] K. V. Zmeu, E. A. Shipitko, A. S. Perevozchikov, "Linear neural model-based predictive controller design for flexible link robot. Proceedings of IEEE International Symposium on Intelligent Control, pp. 293-298, 2004.
[17] T. Henmi, M. Deng, A. Inoue, "Adaptive control of a two-link planar manipulator using nonlinear model predictive control", Proceedings of IEEE International Conference on Mechatronics and Automation, Xi'an, China, pp. 1868-1873, 2010.
[18] R. Hedjar, R. Toumi, P. Boucher, D. Dumur, "Finite horizon nonlinear predictive control by the Taylor approximation: Application to robot tracking trajectory", International Journal of Applied Mathematics and Computer Science, Vol. 15, No. 4, pp. 527–540, 2005.
[19] W. H. Chen, "Stability analysis of classic finite horizon model predictive control", International Journal of Control, Automation, and Systems, Vol. 8, No. 2, pp. 187-197, 2010.
[20] G. Grimm, M. J. Messina, S. E. Tuna, A. R. Teel, "Model predictive control: for want of a local control Lyapunov function, all is not lost", IEEE Transactions on Automatic Control Vol. 50, No. 5, pp. 546-558, 2005.
[21] L. Grune, "Computing stability and performance bounds for unconstrained NMPC schemes", Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, USA, pp. 1263-1268, 2007.
[22] F. L. Lewis, C. T. Abdallah, D. N. Dawson, Control of Robot Manipulators Theory and Practice. Marcel Dekker Inc, 2004.
[23] T. Yoshikawa, Foundation of Robotics, Analysis and Control, MIT Press: Boston, USA, 1990.
[24] R. Hedjar, P. Boucher, "Nonlinear receding-horizon control of rigid link robot manipulators", International Journal of Advanced Robotic Systems, Vol. 2, No. 1, pp. 15-24, 2005.
[25] M. T. Hagan, H. B. Demuth, M. Beale, Neural Network Design, PWS Publishing Company: Boston, USA, 1995.

Appendix

Levenberg-Marquardt algorithm can be expressed in 4 steps as follows:

$$F(\mathbf{x}) = \sum_{i=1}^{N} (v_i)^2 = \mathbf{v}^T\mathbf{v} \tag{A-1}$$

$$\Delta\mathbf{x}_k = -[\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k), \quad \mu > 0$$

$$\mathbf{v}^T = [v_1\ v_2\ \dots\ v_N] = [e_{1,1}\ e_{2,1}\ \dots\ e_{S^M,1}\ e_{1,2}\ \dots\ e_{S^M,Q}]$$

$$\mathbf{x}^T = [x_1\ x_2\ \dots\ x_n] = \left[w^1_{1,1}\ w^1_{1,2}\ \dots\ w^1_{S^1,R}\ b^1_1\ \dots\ b^1_{S^1}\ w^2_{1,1}\ \dots\ b^M_{S^M}\right]$$

$$N = Q \times S^M \quad \text{and}$$
$$n = S^1(R+1) + S^2(S^1+1) + \cdots + S^M(S^{M-1}+1)$$

$$\mathbf{J}(\mathbf{x}) = \frac{\partial\mathbf{v}(\mathbf{x})}{\partial\mathbf{x}} \tag{A-2}$$

1. Compute the sum of squared errors over all inputs, $F(\mathbf{x})$, using (A-1).
2. Solve (A-2) to obtain $\Delta\mathbf{x}_k$.

3. Recompute the sum of squared errors using $\mathbf{x}_k + \Delta\mathbf{x}_k$. If this new sum of squares is smaller than that computed in step 1, then divide μ by $\vartheta > 1$, let $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$ and go back to step 1. If the sum of squares is not reduced, then multiply μ by ϑ and go back to step 2.

In the above algorithm, $\mathbf{W}^i$ is the weight matrix of the i^{th} layer, $\mathbf{b}^i$is the bias vector of the i^{th} layer, R is the number of inputs, $S^i$is the number of neurons in the i^{th} layer, and Q and N are the number of the input/target pairs and the number of elements in $\boldsymbol{v}(\mathbf{x})$, respectively.