Proceedings of the 2000 IEEE International Conference on Robotics and Automations, San Francisco, CA, USA, May 2000, pp. 1906-1911.

Neural Network Controller for Constrained Robot Manipulators

Shenghai Hu, Marcelo H. Ang Jr., and H. Krishnan
Dept of Mechanical and Production Engineering
National University of Singapore
Singapore 119260
mpeangh@nus.edu.sg

Abstract

In this paper, a neural network controller for constrained robot manipulators is presented. A feed-forward neural network is used to adaptively compensate for the uncertainties in the robot dynamics. Training signals are proposed for the feed-forward neural network controller. The neural network weights are tuned on-line, with no off-line learning phase required. It is shown that the controller is able to deal with the uncertainties of the robot, which include modelled undertainties (dynamic parameter uncertainties, etc.) as well as unmodelled uncertainties (frictions, etc). The suggested controller is simple in structure and can be implemented easily. The controller has the Proportional-Integral (PI) type force feedback control structure with a low proportional force feedback gain. Detailed experimental results show the effectiveness of the proposed controller.

1 Introduction

To apply robot manipulators to a wider class of tasks, it is necessary to control not only the position of a manipulator but also the force exerted by its end-effector on an object or environment.

Force control of manipulators has been studied by many researchers [1]-[3]. Constrained motion control has been extensively studied in recent years. In constrained motion control, the robot's end-effector is assumed to be in contact with rigid frictionless surfaces [5]. As a result, kinematic constraints are imposed on the manipulator motion, which correspond to some algebraic constraints among the manipulator state variables. It is necessary to control both the motion of the robot's end effector on the constraint surfaces and the generalized constrained forces.

A general theoretical framework of constrained motion control is rigorously developed in [5]. The proposed controller is based on a modification of the computed torque method. In [4], linear descriptor system theory is applied to design control laws for constrained motion control. The controller is derived based on a linearized dynamic model of the manipulator. In [6], state feedback control and dynamic state feedback control are used to linearize the robot dynamics with respect to motion and contact force subsystems respectively.

The above methods of controller design are based on the knowledge of the exact dynamic model of constrained robot systems. From a practical point of view, in many applications, robot models have many uncertainties in the values of the parameters describing its dynamic properties, such as unknown moments of inertia. In addition, there is also the problem of unmodelled dynamics (e.g., friction). It is hard to estimate the exact form of the model and the values of the dynamic parameters, thus complicating the control design problem significantly. This has motivated the use of adaptive control, sliding mode control, robust control, etc for controller design for constrained robots.

The ability of the neural network (NN) to approximate arbitrary non-linear functions and to learn through examples lends it to many useful applications in control engineering. Many researchers have applied the NN in robot motion control with substantial success [8-10]. Few research works have also dealt with NN controller design in robot force control [11,12,13].

In this paper, we consider the design of NN controllers for force control in constrained robots. A nonlinear transformation similar to [7] is used to decouple the robot dynamics into two subsystems – motion subsystem and force subsystem respectively. A NN control law is proposed based on the decoupled dynamic equations, and a suitable online update law for the NN is derived. Experimental results illustrate the effectiveness of the proposed controller.

2 Dynamic Model of Constrained Robot Manipulators

The dynamic model of a robot in constrained motion is described by a set of nonlinear differential and algebraic equations [5]

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + \tau_f(\dot{q}) = f + u \quad (1)$$

$$\Phi(q) = 0 \quad (2)$$

$$f = J^T(q)\lambda \quad (3)$$

where $q \in R^n$ is the vector of joint positions, $u \in R^n$ is the vector of generalized torque inputs, $M(q) \in R^{n \times n}$ is the inertia matrix, $C(q,\dot{q})\dot{q} \in R^n$ is the vector characterizing Coriolis and centrifugal forces, $g(q) \in R^n$ is the gravitational force vector, and $\tau_f \in R^n$ is the friction force vector. Equation (2) represents a set of m independent kinematic equations that describe the constrained surfaces. These functions are assumed to be twice continuously differentiable with a Jacobian denoted by $J(q) = \dfrac{\partial \Phi}{\partial q}$.

The constraint force vector $f \in R^n$, is expressed in terms of a generalized multiplier $\lambda \in R^m$ by (3).

The actual position of the robot end effector along the free motion z_1 (in task space) can be described as

$$z_1 = h_1(q) \tag{4}$$

while satisfying (2) (i.e., the robot maintains contact on the constrained surface). The actual contact forces z_2 is described as

$$z_2 = \lambda . \tag{5}$$

The manipulator is required to track a time-varying position trajectory $r_1(t)$ and a time-varying force trajectory $r_2(t)$. The control objective is to find a feedback control law so that the constrained manipulator's actual position and force track the desired position and force trajectory $r_1(t)$ and $r_2(t)$ respectively, i.e.

$$z_1(t) - r_1(t) = h_1(q(t)) - r_1(t) \to 0, \text{ as } t \to \infty$$
$$z_2(t) - r_2(t) = \lambda(t) - r_2(t) \to 0 . \quad \text{as } t \to \infty .$$

Here we present the fundamental task space control law [7], in which a nonlinear state transformation is used firstly to decompose the constrained system (1)-(2) into two subsystems, one describing the motion control subsystem, and the other describing the force control subsystem. Secondly, a nonlinear state feedback is determined to exactly linearize these subsystems. Then, linear system control theory is applied to control the resulting linear systems.

The nonlinear coordinate transformation is selected as [7]

$$x_a = \begin{bmatrix} \tilde{x}_1 \\ \bar{x}_1 \end{bmatrix} = \begin{bmatrix} \Phi(q) \\ h_1(q) \end{bmatrix}, \quad \dot{x}_a = \begin{bmatrix} \dot{\tilde{x}}_1 \\ \dot{\bar{x}}_1 \end{bmatrix} = \begin{bmatrix} J(q) \\ P(q) \end{bmatrix} \dot{q} \tag{6}$$

where $P(q) = \dfrac{\partial h_1}{\partial q} = \dfrac{\partial \bar{x}_1}{\partial q}$.

From its inverse, we obtain

$$\dot{q} = Q(q)\dot{x}_a \tag{7}$$

where

$$Q(q) = \begin{bmatrix} J(q) \\ P(q) \end{bmatrix}^{-1} . \tag{8}$$

Furthermore

$$\ddot{q} = Q(q)\ddot{x}_a + \dot{Q}(q)\dot{x}_a \tag{9}$$

Substituting (3), (7), (9) into (1), and multiplying both sides by $Q^T(q)$, the dynamic equation (1) can be expressed in terms of the new coordinates as

$$\bar{M}(q)\ddot{x}_a + \bar{C}(q,\dot{q})\dot{x}_a + \bar{g}(q) + \bar{\tau}_f = Q^T(q)u + Q^T(q)J^T(q)\lambda \tag{10}$$

where

$$\bar{M}(q) = Q^T(q)M(q)Q(q) \tag{11}$$
$$\bar{C}(q,\dot{q}) = Q^T(q)M(q)\dot{Q}(q) + Q^T(q)C(q,\dot{q})Q(q) \tag{12}$$
$$\bar{g}(q) = Q^T(q)g(q) \tag{13}$$
$$\bar{\tau}_f = Q^T(q)\tau_f(\dot{q}) . \tag{14}$$

If we now define the following partitioning matrix

$$\begin{bmatrix} E_1 \\ E_2 \end{bmatrix} = \begin{bmatrix} I_m & 0 \\ 0 & I_{n-m} \end{bmatrix} \tag{15}$$

then, using the fact that $\Phi(q) = 0 \Rightarrow \tilde{x}_1 = 0,\ \dot{\tilde{x}}_1 = \ddot{\tilde{x}}_1 = 0$, we have

$$x_a = \begin{bmatrix} \Phi(q) = 0 \\ \bar{x}_1 \end{bmatrix} = E_2^T \bar{x}_1 . \tag{16}$$

Thus the dynamic equation (10) can be expressed as two subsystems:

$$E_1\bar{M}E_2^T\ddot{\bar{x}}_1 + E_1\bar{C}E_2^T\dot{\bar{x}}_1 + E_1\bar{g} + E_1\bar{\tau}_f = E_1Q^Tu + \lambda \tag{17}$$
$$E_2\bar{M}E_2^T\ddot{\bar{x}}_1 + E_2\bar{C}E_2^T\dot{\bar{x}}_1 + E_2\bar{g} + E_2\bar{\tau}_f = E_2Q^Tu . \tag{18}$$

In obtaining (17) and (18) we have used the fact that $E_1Q^T(q)J^T(q)\lambda = \lambda$, $E_2Q^T(q)J^T(q) = 0$.

Consider the fundamental constrained robot control law [7]

$$\begin{aligned} u &= Q^{-T}\hat{\bar{M}}E_2^T(\ddot{r}_1 - K_v(\dot{\bar{x}}_1 - \dot{r}_1) - K_p(\bar{x}_1 - r_1)) + Q^{-T}\hat{\bar{C}}E_2^T\dot{\bar{x}}_1 \\ &\quad + Q^{-T}\hat{\bar{g}} - J^T(q)r_2 - J^T(q)K_f(r_2 - \lambda) \\ &= \hat{M}QE_2^T(\ddot{r}_1 - K_v(\dot{\bar{x}}_1 - \dot{r}_1) - K_p(\bar{x}_1 - r_1)) + \hat{C}QE_2^T\dot{\bar{x}}_1 + \hat{g} \\ &\quad + \hat{M}\dot{Q}E_2^T\dot{\bar{x}}_1 - J^Tr_2 - J^TK_f(r_2 - \lambda) \end{aligned} \tag{19}$$

where $\hat{M}(q)$, $\hat{C}(q,\dot{q})$ and $\hat{g}(q)$ are the estimates of $M(q)$, $C(q,\dot{q})$ and $g(q)$ respectively, $\hat{\bar{M}}(q)$, $\hat{\bar{C}}(q,\dot{q})$ and $\hat{\bar{g}}(q)$ are estimates of $\bar{M}(q)$, $\bar{C}(q,\dot{q})$ and $\bar{g}(q)$ respectively, K_v, K_p are diagonal positive definite matrices of dimension (n-m), K_f is a diagonal positive definite matrix of dimension m.

Let $e_1(t) = r_1(t) - \bar{x}_1$ and $e_2(t) = \lambda - r_2(t)$.

Substituting the control law (19) into (17), (18), we obtain

$$\begin{aligned} E_2Q^T\hat{M}QE_2^T(\ddot{e}_1 + K_v\dot{e}_1 + K_pe_1) &= E_2Q^T(\Delta MQE_2^T\ddot{\bar{x}}_1 \\ &\quad + \Delta M\dot{Q}E_2^T\dot{\bar{x}}_1 + \Delta CQE_2^T\dot{\bar{x}}_1 + \Delta g + \tau_f) \end{aligned} \tag{20}$$

$$\begin{aligned} (I + K_f)e_2 &= E_1Q^T(\Delta M(q)QE_2^T\ddot{\bar{x}}_1 + \Delta M\dot{Q}E_2^T\dot{\bar{x}}_1 + \Delta C(q,\dot{q})QE_2^T \\ &\quad + \Delta g + t_f) - E_1Q^T\hat{M}(q)QE_2^T(\ddot{e}_1 + K_v\dot{e}_1 + K_pe_1) \end{aligned} \tag{21}$$

where $\Delta M = M(q) - \hat{M}(q)$, $\Delta C = C(q,\dot{q}) - \hat{C}(q,\dot{q})$, and $\Delta g = g(q) - \hat{g}(q)$.

If we define

$$\Psi = \Delta M(q)QE_2^T\ddot{\bar{x}}_1 + \Delta M\dot{Q}E_2^T\dot{\bar{x}}_1 + \Delta C(q,\dot{q})QE_2^T\dot{\bar{x}}_1$$

$$+\Delta g+\tau_f \tag{22}$$

then we can rewrite (20) and (21) as

$$\ddot{e}_1+K_v\dot{e}_1+K_pe_1=(E_2Q^T\hat{M}(q)QE_2^T)^{-1}E_2Q^T\Psi \tag{23}$$

$$(I+K_f)e_2=$$

$$E_1Q^T\Psi-E_1Q^T\hat{M}(q)QE_2^T(E_2Q^T\hat{M}(q)QE_2^T)^{-1}E_2Q^T\Psi \tag{24}$$

In the ideal case where $\Delta M=\Delta C=\Delta g=0$, and $\tau_f=0$ then $\Psi=0$, so equation (23) becomes

$$\ddot{e}_1+K_v\dot{e}_1+K_pe_1=0, \tag{25}$$

and equation (24) becomes

$$(I+K_\lambda)e_2=0 \quad . \tag{26}$$

Thus in the ideal case, the control law (19) applied to a constrained robot results in the closed loop system described by (25), (26). From (25) and (26) we find that with the control law (19), the position error approaches zero exponentially, and the force error is identically equal to zero, so the robot satisfies the required objectives.

Since there are always uncertainties in the robot dynamic model, the ideal dynamic model assumption is not valid in general, that is $\Psi\neq 0$. Thus with the control law (19), the position and force error will not go zero, but are governed by (23) and (24) whose performance is degraded and unpredictable. Thus the fundamental constrained robot force and position control law (19) is not robust in practice. To improve the robustness of the control law (19), a neural network controller can be added to (19) to compensate for effects of model uncertainities in (20) and (21).

3 Proposed NN Controller Scheme

Here a neural network (NN) controller is proposed for the constrained robot system using two-layer feed-forward neural network. Let the NN output be denoted as v. A new control law is defined as

$$u=\hat{M}QE_2^T(\ddot{r}_1-K_v(\dot{\bar{x}}_1-\dot{r}_1)-K_p(\bar{x}_1-r_1))+\hat{M}\dot{Q}E_2^T\dot{\bar{x}}_1+\hat{C}QE_2^T\dot{\bar{x}}_1+\hat{g}$$

$$-J^Tr_2-J^TK_f(r_2-\lambda)+v-J^TK_I\int_0^t(r_2-\lambda)d\tau \tag{27}$$

where $\hat{M}$, $\hat{C}$ and $\hat{g}$ are estimates of $M(q)$, $C(q,\dot{q})$ and $g(q)$, K_v, K_p are diagonal positive definite matrices of dimension (*n*-*m*), K_f, K_I are diagonal matrices of dimension *m*.

Substituting the controller (27) into (17) and (18) yields the corresponding closed loop error system as

$$\ddot{e}_1+K_v\dot{e}_1+K_pe_1=$$

$$(E_2Q^T\hat{M}(q)QE_2^T)^{-1}E_2Q^T(\Psi-v) \tag{28}$$

$$(I+K_\lambda)e_2+K_i\int_0^t e_2d\tau=$$

$$E_1Q^T(\Psi-v)-E_1Q^T\hat{M}QE_2^T(E_2Q^T\hat{M}QE_2^T)^{-1}E_2Q^T(\Psi-v) \tag{29}$$

Define an error signal ζ as

$$\zeta=\ddot{e}_1+K_v\dot{e}_1+K_pe_1. \tag{30}$$

The NN control objective is to generate v to reduce ζ in (30) to zero. Therefore here we use v as the training signal for the NN. The ideal value of v at $\zeta=0$ is

$$v=\Psi=\Delta M\ddot{\bar{x}}_1+\Delta M\dot{Q}E_2^T\dot{\bar{x}}_1+\Delta CQE_2^T\dot{\bar{x}}_1+\Delta g+\tau_f. \tag{31}$$

Clearly minimizing the error signal ζ allows us to achieve the required force and motion tracking control objectives.

4 Neural Network Compensator Design

The two-layer feedforward neural network (with $3(n-m)$ inputs and n outputs) shown in Fig. 1 is used as the compensator. It is composed of an input buffer, a non-linear hidden layer, and a linear output layer.

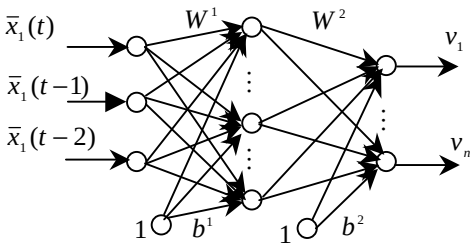

Fig. 1. NN Controller Structure

The inputs $X=[\bar{x}_1^T(t)\quad \bar{x}_1^T(t-1)\quad \bar{x}_1^T(t-2)]^T$ are multiplied by weights w_{ji}^1 and summed at each hidden node, then the nodes are activated through a nonlinear (sigmoid) function $f(\bullet)$ that is bounded in magnitude between 0 and 1; i.e.

$$f(\bullet)=\frac{1}{1+\exp(-(\bullet))}. \tag{32}$$

The activated signals are multiplied by weights w_{kj}^2 and summed at each output node. Thus, the output v_k at a linear output node can be calculated from inputs as

$$v_k=\left[\sum_{j=1}^{n_H}w_{kj}^2\left(\frac{1}{1+\exp(-(\sum_{i=1}^{n_I}x_iw_{ji}^1+b_i^1))}\right)\right]+b_k^2 \quad , \tag{33}$$

where $n_I=3(n-m)$ is the number of inputs, n_H is the number of hidden units, x_i is the ith input of vector X, w_{ji}^1 is the first layer weight between ith input and jth hidden neurons, w_{kj}^2 is the second layer weight between jth hidden neuron and kth output neuron, b_j^1 is a bias weight for jth hidden neuron, and b_k^2 is a bias weight for kth output neuron.

The weight updating law minimizes the objective function J, which is a quadratic function of the training signals ζ, i.e.

$$J=\tfrac{1}{2}\zeta^T\zeta \tag{34}$$

Differentiating (34) yields the gradient of J as follows:

$$\frac{\partial J}{\partial w}=\frac{1}{2}(\frac{\partial \zeta^T}{\partial w}\zeta+\zeta^T\frac{\partial \zeta}{\partial w})=\frac{\partial \zeta^T}{\partial w}\zeta\ , \tag{35}$$

and ζ is obtained from (28) and (30) as

$\zeta=(E_2Q^T\hat{M}(q)QE_2^T)^{-1}E_2Q^T(\Psi-v)$.

Then,

$$\frac{\partial \zeta^T}{\partial w}=-\frac{\partial v^T}{\partial w}((E_2Q^T\hat{M}QE_2^T)^{-1}E_2Q^T)^T\ . \tag{36}$$

The back-propagation update rule for the weights with a momentum term is now chosen as

$$\Delta w=-\eta\frac{\partial J}{\partial w}=\eta\frac{\partial v^T}{\partial w}((E_2Q^T\hat{M}QE_2^T)^{-1}E_2Q^T)^T\zeta \tag{37}$$

$$\begin{aligned}w(t)&=\Delta w+\alpha w(t-1)\\&=\eta\frac{\partial v^T}{\partial w}((E_2Q^T\hat{M}QE_2^T)^{-1}E_2Q^T)^T\zeta+\alpha w(t-1)\end{aligned} \tag{38}$$

where η is the update rate and α is the momentum coefficient. In order to evaluate (38), note specifically that

$$\frac{\partial v^T}{\partial w_{kj}^2}=\left[\frac{\partial v_1}{\partial w_{kj}^2}\ \cdots\ \frac{\partial v_n}{\partial w_{kj}^2}\right]=[0\cdots\underset{\substack{\uparrow\\k-col}}{s_j}\cdots 0] \tag{39}$$

$$\frac{\partial v^T}{\partial b_k^2}=\left[\frac{\partial v_1}{\partial b_k^2}\ \cdots\ \frac{\partial v_n}{\partial b_k^2}\right]=[0\cdots\underset{\substack{\uparrow\\k-col}}{1}\cdots 0] \tag{40}$$

$$\frac{\partial v^T}{\partial w_{ji}^1}=\left[\frac{\partial v_1}{\partial w_{ji}^1}\ \cdots\ \frac{\partial v_n}{\partial w_{ji}^1}\right] \tag{41}$$

where

$$\frac{\partial v_k}{\partial w_{ji}^1}=\sum_{l=1}^{n}\frac{\partial v_k}{\partial s_l}\frac{\partial s_l}{\partial y_l}\frac{\partial y_l}{\partial w_{ji}^1}=w_{kj}^2s_j(1-s_j)x_i \tag{42}$$

$$\frac{\partial v^T}{\partial b_j^1}=\left[\frac{\partial v_1}{\partial b_j^1}\ \cdots\ \frac{\partial v_n}{\partial b_j^1}\right] \tag{43}$$

$$\frac{\partial v_k}{\partial b_j^1}=\sum_{l=1}^{n}\frac{\partial v_k}{\partial s_l}\frac{\partial s_l}{\partial y_l}\frac{\partial y_l}{\partial b_j^1}=w_{kj}^2s_j(1-s_j) \tag{44}$$

$$s_j=\frac{1}{1+\exp\left(-\left(\sum_{i=1}^{n_I}x_iw_{ji}^1+b_j^1\right)\right)} \tag{45}$$

where s_j is the output of j th hidden neuron. Thus, by substituting (39)-(45) into (38), we can update the weights of the neural network compensator.

5 Experimental Results

Real-time implementation of the NN controller has been carried out using the five-bar linkage parallelogram robot with two degrees of freedom (DOF) as shown in Fig. 2. The robot was designed and built at our laboratory for our experimental work. Experiments have been performed on this robot to evaluate the effectiveness of the proposed neural network controller.

The robot dynamic coefficient matrices (1) and its forward kinematics are given by

$$M(q)=\begin{bmatrix}1.747 & -0.467\cos(q_2-q_1)\\ -0.467\cos(q_2-q_1) & 1.439\end{bmatrix} \tag{46}$$

$$C(q_1,q_2,\dot{q}_1,\dot{q}_2)=\begin{bmatrix}0 & 0.476\sin(q_2-q_1)\dot{q}_2\\ -0.476\sin(q_2-q_1)\dot{q}_1 & 0\end{bmatrix} \tag{47}$$

$$g(q_1,q_2)=\begin{bmatrix}0\\0\end{bmatrix} \tag{48}$$

$$\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}0.4\cos(q_1)-0.61\cos(q_2)\\0.4\sin(q_1)-0.61\sin(q_2)\end{bmatrix}. \tag{49}$$

The robot is in contact with a rigid surface as shown in Fig. 2. It satisfies the constraint equation $\Phi(x)=x-0.55=0$ when expressed in task space. The constraint can also be expressed in joint variables as $\Phi(q)=0.4\cos(q_1)-0.61\cos(q_2)-0.55=0$. It then follows that $J(q)=[-0.4\sin(q_1)\ \ 0.61\sin(q_2)]$.

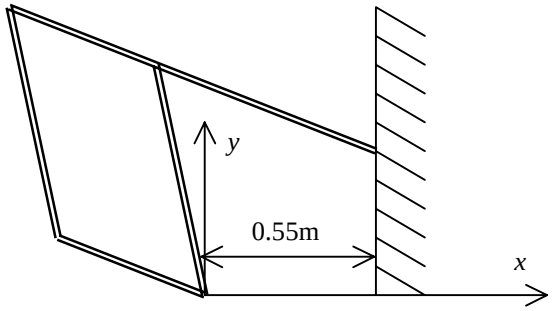

Fig. 2. Configuration of the robot

The nonlinear transformation given by (5) is selected as

$$x_a=\begin{bmatrix}\tilde{x}_1\\ \bar{x}_1\end{bmatrix}=\begin{bmatrix}0.4\cos q_1-0.61\cos q_2-0.55\\ 0.4\sin q_1-0.61\sin q_2\end{bmatrix}. \tag{50}$$

The position output given by (4) is

$$y(t)=\bar{x}_1=h_1(q)=0.4\sin q_1-0.61\sin q_2\ . \tag{51}$$

The velocity and acceleration of the joint motion are approximated using the backward difference method and a low-pass filter.

The algorithm for an estimate of the motion velocity is given by

$$\dot{\bar{x}}_1^*=\frac{\bar{x}_1(t)-\bar{x}_1(t-1)}{t_s}\quad ;\ \dot{\bar{x}}_1=0.8\dot{\bar{x}}_1(t-1)+0.2\dot{\bar{x}}_1^* \tag{52}$$

and the algorithm for an estimate of the motion acceleration is given by

$$\ddot{\bar{x}}_1^*=\frac{\dot{\bar{x}}_1(t)-\dot{\bar{x}}_1(t-1)}{t_s}\quad ;\ \ddot{\bar{x}}_1=0.8\ddot{\bar{x}}_1(t-1)+0.2\ddot{\bar{x}}_1^* \tag{53}$$

where t_s denotes a sampling time interval of $2ms$ for the measurement.

A force filter is also used to filter the noise in the force measurements and is given by

$$f(k)=K_sf+(1-K_s)f(k-1) \tag{54}$$

where f is the measured force from the force sensor.

The NN controller consists of three input neurons, six hidden neurons and two output neurons. All the weights and biases are set to zero initially, the momentum coefficient $\alpha = 0.9$ and the learning rate is chosen as $\eta = 0.01$. The weights are updated at each sampling time on-line. The controller gains in (27) are selected as $K_p = 200$, $K_v = 5$, $K_f = 4$ and $K_i = 10$. Sampling time is 2ms. The performances of the proposed scheme are tested by tracking desired motion and force trajectories under different conditions.

Case 1:

To include model uncertainties, we just choose the nominal dynamic model parameters of the robot as

$$\hat{M}(q) = \begin{bmatrix} 1 & -0.2\cos(q_2 - q_1) \\ -0.2\cos(q_2 - q_1) & 0.5 \end{bmatrix}$$

$$\hat{C}(q_1, q_2, \dot{q}_1, \dot{q}_2) = \begin{bmatrix} 0 & 0.2\sin(q_2 - q_1)\dot{q}_2 \\ -0.2\sin(q_2 - q_1)\dot{q}_1 & 0 \end{bmatrix}$$

$$\hat{g}(q_1, q_2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The robot is required to move along the constraint surface tracking a time varying position with trajectory $r_1(t) = 0.35 - 0.1\sin(0.1\pi t)$, while simultaneously exerting a constant force of $10N$ on the constraint surface. The experiment results are shown in Figs. 3-5. The time responses of the position tracking error $e_1(t)$ with and without NN are shown in Fig.3. Fig. 4 shows the force tracking error $e_2(t)$ of the robot with and without NN. In Fig. 5 we also plot the output of neural network $v(t)$.

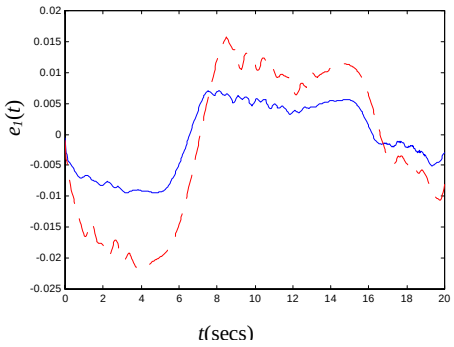

Fig. 3. Plots of position error
(With NN: solid-line, Without NN: dotted-line)

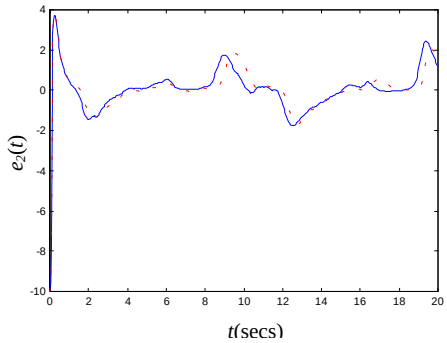

Fig. 4. Plots of force error
(With NN: solid-line, Without NN: dotted-line)

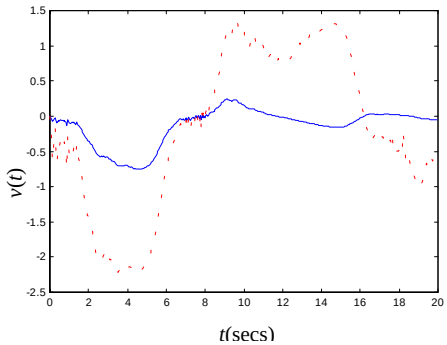

Fig. 5. Outputs of the neural network
(v_1(t): solid-line, v_2 (t): dotted-line)

Case 2:

Here we just choose the nominal dynamic model parameters of the robot as

$$\hat{M}(q) = \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix},\ \hat{C}(q_1, q_2, \dot{q}_1, \dot{q}_2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},\ \hat{g}(q_1, q_2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The control parameters and the desired trajectory are the same as in case 1.

The experimental results are shown in Figs. 6-8. The time responses of tracking error $e_1(t)$ with and without NN are shown in Fig. 6. In Fig. 7 the force error $e_2(t)$ of the robot with and without NN is plotted. In Fig. 8 the output $v(t)$ of neural network is plotted.

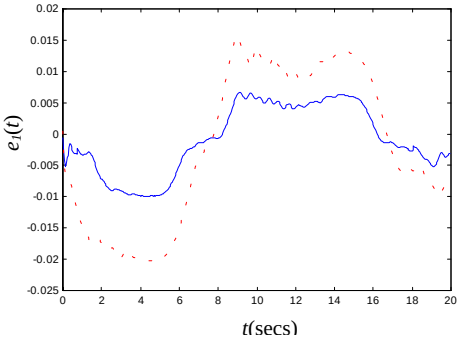

Fig. 6. Plots of position error

(With NN: solid-line, Without NN: dotted-line)

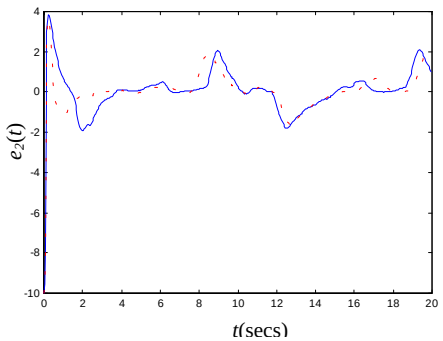

Fig. 7. Plots of force error
(With NN: solid-line, Without NN: dotted-line)

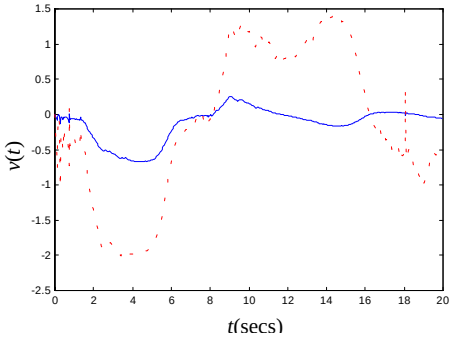

Fig. 8. Outputs of the neural network
(v_1 (t): solid-line, v_2 (t): dotted-line)

From the experimental results above, we find that in presence of the model uncertainties, the suggested NN controller can efficiently compensate for the parameter uncertainties and the unmodelled uncertainties (frictions, etc). The NN controller has good tracking capability, especially in position tracking.

6 Conclusions

A simple and effective neural network controller, which uses the two-layer feed-forward neural network, is proposed for the constrained robot in the task space. Experiments have been carried out on a 2DOF direct-drive robot manipulator to evaluate the performance of the proposed controller. The experimental results show that the proposed neural network controller is able to effectively compensate for the uncertainties in the parameter values of the robot dynamics as well as for unmodelled dynamics.

References

[1] K. J. Salisbury, “Active Stiffness Control of a Manipulator in Cartesian Coordinates,” *Proceedings of the 19th IEEE Conference on Decision and Control*, Albuquerque, New Mexico, pp. 95-100, 1980.

[2] M. H. Raibert and J. J. Craig, "Hybrid Position/Force Control of Manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 102, No. 2, pp.126-133, June 1981.

[3] N. Hogan, "Impedance Control: An Approach to Manipulation: Part I, II, III," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 107, No.1, pp. 1-24, March 1985.

[4] J. K. Mills and A. A. Goldenberg, "Force and Position Control of Manipulators During Constrained Motion Tasks," *IEEE Journal of Robotics and Automation*, Vol.5, No.1, pp. 30-46. February 1989.

[5] N. H. McClamroch and D. W. Wang, "Feedback Stabilization and Tracking of Constrained Robots," *IEEE Transactions on Automation and Control*, Vol. 33, No. 5, pp. 419-426, May 1988.

[6] X. Yun, "Dynamic State Feedback Control of Constrained Robot Manipulator," *Proceedings of the 27th IEEE Conference on Decision and Control*, Austin, Texas, pp. 622-626, Dec. 1988.

[7] X. Wang, "Contact Force and Position Control in Constrained Robots: Theory and Experiments," *M. Eng. Thesis, Dept of Mechanical and Production Engineering, National University of Singapore*, 1997.

[8] Y. H. Kim and F. L. Lewis, “Neural Network Output Feedback Control of Robot Manipulators,” *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 2, pp. 301-309, April 1999.

[9] S. S. Ge, C. C. Hang and L.C. Woon, "Adaptive Neural Network Control of Robot Manipulators in Task Space," *IEEE Transactions on Industrial Electronics*, Vol. 44, No.6, pp. 746-752, December 1997.

[10] S. Jung and T.C. Hsia, "A Study of Neural Network Control of Robot Manipulators," *Robotica*, Vol. 14, pp. 7-15, 1996.

[11] T. Yabuta and T. Yamada, "Force Control Using Neural Networks," A*dvanced Robotics*, Vol. 7, No. 4, pp. 395-408, 1993.

[12] D. Katic and M. Vukobratovic, "Robot Compliance Control Algorithm Based on Neural Network Classification and Learning of Robot-Environment

Dynamic Models," *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, New Mexico, pp. 2632-2637, April 1997.

[13] K. Kiguchi and T. Fukuda, "Robot Manipulator Hybrid Control for an Unknown Environment Using Visco-Elastic Neural Networks," *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium, pp. 1447-1452, May 1998.