A Neural Network Controller for Trajectory Control of Industrial Robot Manipulators

Zhao-Hui Jiang
Department of Mechanical Systems Engineering, Hiroshima Institute of Technology
2-1-1 Miyake, Saeki-ku, Hiroshima 731-5193, Japan
Email: jiang@cc.it-hiroshima.ac.jp

Taiki Ishita
Yushin Corporation, Tokyo, Japan

Abstract—This paper addresses the issue of trajectory tracking control based on a neural network controller for industrial manipulators. A new control scheme is proposed based on neural network technology and linear feedback approach for tracking a planned trajectory. In detail, the control system is designed with two parallel subsystems designed separately. One is a linear controller, and another one is a neural network controller. The former is designed for trajectory tracking error regulation, the later for force/torque generation required by the designed dynamic trajectory. A leaning law for online weight updating of the neural network controller is derived based on simplified dynamic model of the robot. A Direct Drive (DD) SCARA type industrial robot arm AdeptOne is used as an application example for trajectory tracking control experiments. Simulations and experiments are carried out on AdeptOne robot. From the simulation and experimental results, the effectiveness and usefulness of the proposed control system are confirmed.

Index Terms—robot manipulator, trajectory control, neural network controller

I. INTRODUCTION

For trajectory control of a robot manipulator, the model-based control design requires a correct dynamic model and precise parameters of the robot. Practically speaking, every dynamic model contains some degrees of incorrectness and every parameter associates with some degrees of error. The incorrectness and error eventually result positioning and/or trajectory tracking errors, and even cause instability of the system. In the past two decades, intensive research activities have been devoted on the design of robust control systems and adaptive control systems for the robot in order to overcome the control system drawback caused by model errors and uncertain parameters, and a great number of research results have been reported, for example[1]~[5]. However, almost parts of them associate with complicated control system design approaches and difficulties in the control system implementation for industrial robot manipulators.

Recent years, neural network technology attracts many attentions in the design of robot controllers. It has been pointed out that multi-layered neural network can be used for the approximation of any nonlinear function. Other advantages of the neural networks often cited are parallel distributed structure, and learning ability. They make such the artificial intelligent technology attractive not only in the application areas such as pattern recognition, information and graphics processing, but also in intelligent control of nonlinear and complicated systems such as robot manipulators[6]~[10]. A new field in robot control using neural network technology is beginning to emerge to deal with the issues related to the dynamics in the robot control design. A neural network based dynamics compensation method has been proposed for trajectory control of a robot system [11]. A combined approach of neural network and sliding mode technology for both feedback linearization and control error compensation has been presented [12]. Sensitivity of a neural network performance to learning rate in robot control has been investigated [13].

In this paper, we present a new and simple control system consisting of a traditional controller and a neural network controller with parallel structure for trajectory tracking control of industrial robot manipulators. First, a linear feedback controller is designed. Second, a neural network with three layers is designed and added to the control system in the parallel way to the linear feedback controller. Finally, a learning scheme used to train the weights of each layer of the neural network is derived by minimizing a criterion prescribed in a quadratic form of the error between a planned trajectory and response of the robot. Control system implementation issue is discussed. Both the motivation function of the neural network and dynamic model used in the calculation of the learning law are simplified to meet practical needs. An industrial manipulator AdeptOne is adopted as an experimental test bed. Trajectory tracking control simulations and experiments are carried out. The results demonstrate effectiveness and usefulness of the proposed control system.

This paper is organized as follows. Sections II presents formulation of the dynamic model and problem statement.
The detail control system design and neural network learning law derivation are given in Section III. Trajectory tracking control simulations are presented in Section IV. In Section V, the role that the neural network controller played in the control system is discussed. Section VI presents experimental results using the test bed AdaptOne Robot. Finally, conclusions are given in Section VII.

II. DYNAMIC MODELING AND PROBLEM STATEMENT

A. Dynamics of Robot Manipulators

The torque-based dynamics of robot manipulator is the popular dynamic model that is widely used for control design and simulation of robot manipulators. For a n-link robot manipulator, the dynamic model can be easily derived and expressed systematically with Lagrange formulation as follows

\[ M(\theta) \ddot{\theta} + H(\theta, \dot{\theta}) \dot{\theta} + g(\theta) = \tau \]  

where, \( \theta \in \mathbb{R}^n \) and \( \tau \in \mathbb{R}^n \) are joint variable and torque, \( M(\theta) \in \mathbb{R}^{n \times n} \) is inertia matrix, \( H(\theta, \dot{\theta}) \dot{\theta} \in \mathbb{R}^n \) contains Coriolis and centrifugal forces, and \( g(\theta) \in \mathbb{R}^n \) denotes gravitational force.

Remarks: In motion equation (1), \( M(\theta) \) is a symmetric matrix, and \( M(\theta) - 2H(\theta, \dot{\theta}) \) is a skew symmetric matrix.

B. The Problem Statement and Our Objective

Theoretically speaking, for joint trajectory tracking control of an industrial robot, dynamic model based control system methods, such as the computed torque control method under the dynamic model given by (1), can be used. System implementation, however, is difficult to perform because of the existence of the uncertainties in the parameters in the dynamics and in the formulation of the dynamics itself. On the other hand, PID controllers are usually building-in almost all industrial robot manipulators. As the significant drawback, however, it is well known that PID control cannot guarantee precise tracking results for given dynamic trajectories since such the control system is essentially driven by trajectory errors themselves. Some approaches for approximating part of dynamics of a robot by using neural network technology have been proposed instead of the dynamic model based control. Complicity of robot dynamics, however, requires complicated structure of neural network, and the complicated neural network makes the implementation difficult in real time control under both software and hardware environment of industrial manipulators.

In this research, we aim at high precision trajectory tracking control of the industrial robot manipulators using simple and applicable control method. We design a control strategy with both technologies of linear feedback control a neural network for taking the advantages of both simplicity on design and implementation of a linear controller, and learning capacity of neural network control. The main idea is to establish a control system with the linear feedback controller and a neural network control scheme which are parallel to each other in the control system for achieving precise tracking control of dynamic trajectories. The detail description of the control system design will be discussed in the next section.

III. NEURAL NETWORK CONTROL SYSTEM DESIGN

A. Structure of the Control System

For tracking planned joint trajectories, we design a manifold to describe the desired tracking performance of the robot as

\[ s = 0 \]  

where \( s = \dot{e} + \lambda e \), \( e = \theta - \Theta_d \), and \( \dot{e} = \dot{\theta} - \dot{\Theta}_d \) and \( \Theta_d \) are the planned joint trajectories, \( \lambda \in \mathbb{R}^{m \times n} \) is a selected positive constant matrix.

Control system is designed with a linear feedback controller and two neural network controllers. The structure of the control system is shown in Figure 1.

![Figure 1. Black diagram of a robot control system with a linear controller and neural network controller.](image)

B. The Control Scheme

The control scheme is given as follows.

\[ \tau = \tau_i + \tau_n \]  

\[ \tau_i \] is control input of the linear controller, and can be simply described as below.

\[ \tau_i = -Ks \]  

where \( K \in \mathbb{R}^{m \times n} \) is a positive-definite gain matrix.

\( \tau_n \) is the control input generated by the neural network controller to be designed. The structure of neural network controller is shown in Figure 2. The detail mathematical description of the neural network is given by

\[ \tau_n = V f(Wq) \]  

where

\[ q = [\theta_1, \theta_2, \ldots, \theta_n, \dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_n]^T \in \mathbb{R}^{2n} \]
denotes input vector with elements being each joint variable, velocity; \( \tau_n = [\tau_{n1}, \tau_{n2} \cdots \tau_{nm}] \in \mathbb{R}^m \) is output vector, \( \mathbf{W} \in \mathbb{R}^{2w \times d} \) and \( \mathbf{V} \in \mathbb{R}^{d \times m} \) with their elements being expressed by \( w_j \) and \( v_{jk} \), are weight matrices from input nodes to the hidden layer and from hidden layer to the output layer; \( f(\cdot) \in \mathbb{R}^l \) is an activation function vector of the hidden layer with elements being selected as a saturation function, such as a sigmoid function; \( l \) is the number of hidden nodes. Though the dimension of robot joint inputs equals joint numbers \( n \), here we denote it as \( m \), i.e. \( m = n \) in order to describe the network controller design without confusion. In this research, we select the activation function vector with its individual element being a hyperbolic function as follows.

\[
f_j(z_j) = \frac{e^{z_j} - e^{-z_j}}{e^{z_j} + e^{-z_j}} \quad (j = 1,2,\ldots,l)
\]

(7)

where \( z_j \) is the summation of input signals to \( j \)th node of the hidden layer and can be given as

\[
z_j = [w_{1j} \ w_{2j} \cdots w_{nj}] \cdot q
\]

(8)

C. Neural Network Learning Law

For tracking control of a robot with a designed dynamic trajectory, only the linear controller is not enough to ensure a proper tracking precision. For this reason, we design the neural network controller such that it takes the important part on which the linear controller has shown its limitation and/or powerlessness. In doing so, the neural network controller should be trained in such the way: the trajectory tracking error getting smaller and smaller while training. First, we choose a performance criterion of the whole control system with a quadratic form with respect to the manifold given in (2), as follows.

\[
E = \frac{1}{2} \tau^T s
\]

(9)

The weights’ learning algorithm is derived based on the back-propagation approach. The tuning law is to give weights’ increments to be proportional to the negative gradient of the performance criterion with respect to the weights. For updating of the weights between the hidden layer and the output layer, we define an increment as

\[
\Delta v_{jk} = -\gamma_{jk} \frac{\partial E}{\partial v_{jk}} \quad (j=1,2,\ldots,l; \ k=1,2,\ldots,m)
\]

(10)

where, \( j \) and \( k \) indicate the one between \( j \)th node of the hidden layer and \( k \)th node of the output layer , and \( \gamma_{jk} \) is a constant of proportionality, to be designed as a learning rate.

Whereas for the weights between the input layer and hidden layer, we give

\[
\Delta w_{ij} = -\eta_{ij} \frac{\partial E}{\partial w_{ij}} \quad (i=1,2,\ldots,2n; \ j=1,2,\ldots,l)
\]

(11)

where \( \eta_{ij} \) is a learning rate to be designed by the user.

Using the chain rule and noting that the weights are independent with \( \tau \), the partial derivative of (10) can be expressed as follows,

\[
\frac{\partial E}{\partial v_{jk}} = \frac{\partial E}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \tau_{mk}} \frac{\partial \tau_{mk}}{\partial v_{jk}}
\]

(12)

In detail, \( \frac{\partial E}{\partial \mathbf{q}} \) can be given as

\[
\frac{\partial E}{\partial \mathbf{q}} = s^T \Gamma \in \mathbb{R}^{1 \times 2n}
\]

(13)

and

\[
\Gamma = [I, \lambda] \in \mathbb{R}^{n \times 2n}
\]

(14)

where, \( I \in \mathbb{R}^{n \times n} \) is a unite matrix. Similarly,

\[
\frac{\partial \mathbf{q}}{\partial \tau_{mk}} = \mathbf{b}_k \in \mathbb{R}^{2n \times 1}
\]

(15)

and

\[
\frac{\partial \tau_{mk}}{\partial v_{jk}} = f_j
\]

(16)

where \( f_j \) is the output of \( j \)th node of the hidden layer.

Similarly, for (11), we use the chain rule to obtain

\[
\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \tau_{ni}} \frac{\partial \tau_{ni}}{\partial \tau_{mk}} \frac{\partial \tau_{mk}}{\partial \tau_{mj}} \frac{\partial \tau_{mj}}{\partial \tau_{mj}} \frac{\partial \tau_{mj}}{\partial v_{ij}}
\]

(17)

The first partial derivative of the right side of above equation can be calculated using (13) and (14). The second partial derivative is a Jacobian matrix of output vector \( \mathbf{q} \) of the robot with respect to the control torque and can be described as

\[
\frac{\partial \mathbf{q}}{\partial \tau_n} = [b_1, b_2, \cdots, b_m] \equiv \mathbf{b} \in \mathbb{R}^{2n \times m}
\]

(18)

It is quite clear that \( \mathbf{b} \) depends on the dynamics of the robot, and needs to be specified for the control implementation. This issue will be discussed in the next
subsection. It should be noted that $b_k$ in (15) is the $k$th column vector of $b$. The third term of the right side of (17) is the partial derivative of the output of the neural network with respect to the output of the $j$th node of the hidden layer. Since the relation between the two layers is linear, it can be determined as the $j$th row vector of the weight matrix between the hidden layer and output layer, i.e.

$$\frac{\partial \tau_n}{\partial f_j} = [v_{j1}, v_{j2}, \ldots, v_{jn}]^T \equiv v_j \in R^n \quad (19)$$

The fourth partial derivative term of right side of (17) can be determined directly using partial derivative $\frac{\partial f_j(z_j)}{\partial z_j} = \frac{\partial f_j(z_j)}{\partial z_j}$ of the hyperbolic function $f_j(z_j)$ as below.

$$\frac{\partial f_j(z_j)}{\partial z_j} = \frac{4}{(e^{z_j} + e^{-z_j})^2} \quad (20)$$

Noting (8) the last partial derivative is obtained as below.

$$\frac{\partial z_j}{\partial w_{ij}} = \delta_i \quad (21)$$

D. An Implementation Issue

In the design of the neural network controller, since we aimed at trajectory tracking performance of the system, we designed the performance criterion using error’s quadratic form of the inputs of the neural network other than using error’s quadratic form of the outputs of the neural network, though the later is much usual in neural network design. It eventually results the use of dynamics of the system in deriving the learning law with back propagation method since the inputs and outputs of the robot system and neural network controller are contrary to each other.

Using (6) as a state variable vector, dynamic model (1) can be rewritten as

$$\dot{q} = a + B\tau \quad (23)$$

where

$$a = \begin{bmatrix} 0 \\ -M^{-1}(\theta)(H(\theta, \dot{\theta})\dot{\theta} + g(\theta)) \end{bmatrix} \in R^{2n},$$

$$B = \begin{bmatrix} 0 \\ M^{-1}(\theta) \end{bmatrix} \in R^{2\times n}.$$ Generally, the solution of (23) can be given by

$$q = \int_{t_0}^{t} a dt + \int_{t_0}^{t} B\tau dt \quad (24)$$

Noting $\tau = \tau_t + \tau_n$, one can numerically calculate $b$ in a real time control process as

$$b = \frac{\partial q}{\partial \tau_n} \approx \int_0^{L} B d\tau \approx \sum_{i=1}^{l}(t_i - t_{i-1})B \quad (25)$$

where $t_i$ indicates the $i$th sampling time.

IV. TRAJECTORY TRACKING CONTROL SIMULATIONS

The simulated robot system is a SCARA type industrial robot manipulator AdeptOne XL shown in Figure 11. Since the third joint is prismatic and dynamically independent with other joints, control subsystem for the third joint can be designed independently and easily. Focusing on control of the most complex part of the robot, we do not take the third joint into consideration in the simulation as well as in the experiments that are presented in Section 6. The fourth joint is extremely light-weight designed comparing with other joints and its link length is zero. Therefore, the fourth joint does not cause dynamic coupling to the others, and we do not take it into consideration in the simulations and experiments as well.

The joint trajectory tracking control simulations were carried out based on a simplified dynamic model of (1). The neural network controller was designed with three layers, four nodes for the input layer and hidden layer respectively, and two nodes for the output layer. The learning scheme was designed using the method given in section III. The desired joint trajectories are designed using triangle functions with amplitudes to be 45 and 30 degrees for joint1 and joint2. The manifold parameter matrix is determined as $\lambda = diag(0.78, 0.33)$, and feedback gain matrix of the linear controller (4) were determined as, $K = diag(0.80, 0.30)$. Learning rates in (10) and (11) were chosen as $\gamma_{ji} = 0.07$, $\gamma_{ij} = 0.04$ ($j = 1, \ldots, 4$), $\eta_i = 0.01$ ($i = 1, \ldots, 4$).

Simulations were taken place under Matlab environment. Figure 3 and Figure 4 show a result of trajectory tracking control simulation using the linear controller only, where the solid lines in the first and second parts of figures present the tracking results of the joint angles and velocities whereas the broken lines show the planned trajectories correspondingly.

Figure 5 and Figure 6 give the planned joint trajectories and tracking control results using the controller combined with the linear feedback controller and neural network controller at the fourth time learning of the neural network controller. The broken lines indicate the planned trajectories that are not easy to be seen since they are almost completely covered by the solid lines i.e. the tracking results.

From the simulation results, it is seen that using the combined control system with linear controller and neural network controller high precise joint trajectory tracking performance can be achieved under learning process of the weights of the neural network.
The robot control system using the linear feedback controller given by (4) is essentially driven by error with respect to the designed manifold (2), i.e. the combined trajectory tracking error of the joint variables and velocities with parameter $\lambda$ being the weight between the variables and velocities. In dynamic trajectory tracking of a robot under linear feedback control, the linear controller plays two important roles: one is to regulate the motion for guaranteeing stability of the robot system; another one is to generate force/torque required by the dynamic trajectory for driving the robot such that it would follow the planned trajectory. The latter needs a big enough tracking error in order to generate the actuating force/torque required by the trajectory for the robot. That is why linear feedback control cannot achieve precise tracking performance for a designed dynamic trajectory. This fact is confirmed by the trajectory tracking simulation results with linear feedback control only as given in Figure 3 and Figure 4.

From the simulation results given in Figure 3 to Figure 6, one can see that the tracking error is largely reduced under the combined control with linear and neural network controllers at the fourth time learning. Now, let us find how the control input of the individual controller changed during the learning process of the neural network controller so that we may clarify what kind of the role the neural network controller plays. The time history of control inputs are shown in Figure 7 to Figure 10. Figures 7 and 8 present the control inputs of the linear controller of joint 1 and 2, respectively. Figures 9 and 10 show the control inputs of their neural network controller counterparts. Each figure contains three parts in its top, centre, and bottom indicating the input without learning, at the first time learning, and at the fourth time learning, respectively.

Comparing Figure 7 with Figure 9 and Figure 8 with Figure 10, it can be found that control input by the neural...
network controller increases whereas the control input generated by the linear controller decreases when the learning time increases. At the fourth time learning, it seems that the neural network controller took the linear controller over and played the main role in generating actuation voltages for the robot.

Standing on the dynamics point of view, with the same tracking accuracy to a designed dynamic trajectory, the whole control input should be the same judging with the same unit of the input (whatever it counted by torque/force or voltage) regardless what kind of control method is used for the trajectory tracking control. In robot control with neural network, it is a popular way to use a neural network to approximate dynamics of the robot rather than use it as a controller itself. From the simulation results, it can be concluded that the neural network controller proposed in this paper not only plays the role as a controller but also play the main role in generating force/torque required by dynamic trajectories just as an approximated dynamic model using neural network in computed torque control.

Although the results given here are limited on those simulations in the learning up to four times, we have carried out much more simulation, and learning even for 20 times, for example. The results show that after some specified time the learning effect judged with the trajectory tracking accuracy will remain unchanged.

VI. TRAJECTORY TRACKING CONTROL EXPERIMENTS USING AN ADAPETONE ROBOT MANIPULATOR

A. The Experimental Test Bed

The experimental test bed used in this research is an AdeptOne XL robot manipulator shown in Figure 11. It is a high performance SCARA type Direct Drive (DD) industrial robot manipulator possessed with 4 joints. Except the third joint being a prismatic joint, other joints are revolute. Though a closed-loop servo system is built-in by Adept Technology Corporation on the basis of servo units and servo motors, using the Advanced Servo Library the user is allowed to access the D/A converter directly to establish a user-designed close-loop servo system for the development of more advanced control system by V+ language. We developed control software on such the software and hardware environment.

B. The Trajectory Tracking Control Experiments

The joint trajectory tracking control experiments were carried out under almost same conditions of the simulations except the feedback gain matrices were chosen as $\lambda = \text{diag}(1.15, 1.50)$, $K = \text{diag}(1.30, 0.40)$, and amplitudes of the trajectories of joint 1 and 2 are planned as 25 and 20 degrees.

The experimental results are shown in Figures 12 to 15. Figures 12 and 13 present the planned joint and...
velocity trajectories and the tracking results for joints 1 and 2 under the conditions without learning, with the first time learning and sixth time learning. Figure 14 and 15 describe time history of the control inputs according to the responses given in Figures 12 and 13 for joints 1 and 2, respectively.

From the experimental results, it can be seen that although the trajectory tracking accuracy is a little bit worse comparing with the simulation results, the trajectory tracking error becomes less and less when learning time increases. It confirms the effectiveness and usefulness of the proposed control method.

VII. CONCLUSIONS

In this paper, we have proposed a dynamic trajectory tracking control method for industrial robot manipulators using a linear feedback controller and a neural network controller. The manifold was designed to prescribe the desired trajectory tracking performance of the robot system. Then the linear controller was designed on the feedback with respect to the manifold. The neural network controller was designed as a three layers feed-forward network, and was added to the system in the parallel way to the linear controller to establish the control system. The learning law of weights of the neural network was derived using a simplified dynamic model of the robot based on the back propagation approach. Dynamic trajectory tracking control simulations and
experiments were carried out using an industrial manipulator AdeptOne XL robot as the test bed. The results showed the effectiveness and usefulness of the proposed control method.

From the simulation and experiment, it was clarified that according increase in the learning time the neural network controller took the linear controller over on playing the main role in the generating of the actuating force/torque required by the dynamic trajectory. From this fact, an important conclusion can be made that the neural network controller can play the same role as the computed torque played, even the neural network is not used to approximate the dynamics of the robot in the so-called computed torque method, and no such the computed torque approach is used in the proposed control method. It also was clarified that the learning effect of the neural network has some limitation, i.e. after some specified time of learning, trajectory tracking accuracy remains unchanged.

REFERENCES


Zhao-Hui Jiang received Ph.D. degree in Precision Engineering from Tohoku University, Sendai, Japan, in 1988.
He is currently a professor in the Department of Mechanical Systems Engineering, Hiroshima Institute of Technology.
Professor Jiang is a member of IEEE, SICE, Robotics Society of Japan, and Japanese Society of Mechanical Engineering.

Taiki Ishita received his B.S. and M.E. degrees both in Mechanical Engineering from Hiroshima Institute of Technology, Hiroshima, Japan, in 2005 and 2007, M.E. respectively.
He is now working in the Yushin Corporation, Tokyo, Japan as an engineer.