Proceedings of the ASME 2017 Dynamic Systems and Control Conference
DSCC2017
October 11-13, 2017, Tysons, Virginia, USA

DSCC2017-5228

NEUROADAPTIVE CONTROL FOR TRAJECTORY TRACKING OF INDIRECT DRIVE ROBOTS

Yu Zhao
Department of Mechanical Engineering
University of California
Berkeley, California
Email: yzhao334@berkeley.edu

Xiaowen Yu
Department of Mechanical Engineering
University of California
Berkeley, California
Email: aliceyu@berkeley.edu

Masayoshi Tomizuka
Department of Mechanical Engineering
University of California
Berkeley, California
Email: tomizuka@me.berkeley.edu

ABSTRACT

Most industrial robots are indirect drive robots, which utilize low torque and high speed motors. Indirect drive robots have gear reducers between the motors and links. Due to the flexibility of transmission units, it is difficult to achieve high accuracy for trajectory tracking. In this paper, a neuroadaptive control, which is essentially a neural network (NN) based adaptive backstepping control approach, is proposed to deal with this problem. The stability of the proposed approach is analysed using Lyapunov stability theory. A data-driven approach is also proposed for the training of the neural network. The effectiveness of the proposed controller is verified by simulation of both single joint and 6-axis industrial robots.

1 INTRODUCTION

Today, most industrial robots are indirect drive for high power/weight ratio and low cost. However, it is hard for indirect drive robots to achieve high trajectory tracking accuracy because of the flexibilities in transmission units in each robot joint [1]. Such flexibilities introduce time-varying mismatches between the positions of actuators and the driven links, which will result in degradation of the tracking performance [2].

Several control approaches have been proposed for robots with flexible transmission units. Some of these approaches require an accurate robot dynamic model: e.g. feedback linearization [3], singular perturbation based approach [4], and the model based feedforward control [5]. Other approaches are model free approaches: e.g. iterative learning control [6] and nonparametric learning control based on Neural Network (NN) techniques [7].

The model based controller relies on a good model. If the model is either too simple to accommodate all complex characteristics in the robots, or too complicated to identify actual dynamics and parameters, the performance of the controller will be poor due to modeling errors. On the other hand, model free approaches can provide reasonable performance most of time because no analytic model is required except that the tuning of the controller may be time consuming or data inefficient. For example, iterative learning control may require many iterations before a good control input can be learned for a specific trajectory while NN always requires large data set for training the controller.

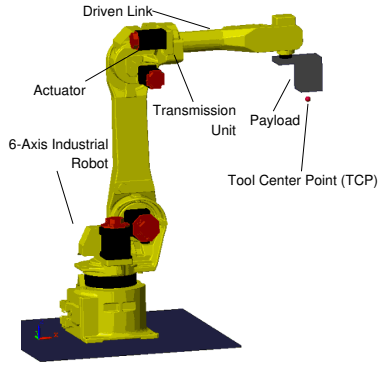

Figure 1. 6-axis indirect drive robot

To address such problems, in this paper, a neuroadaptive controller, which is essentially a neural network based adaptive backstepping control is proposed. Instead of using only dynamic model of the robot, an auxiliary model parameterized by a neural network is used to approximate the modeling error. The controller is designed in two stages: an off-line training stage and an online adaptive training stage. To train for a specific trajectory, it takes only at most two iterations until convergence and only requires the data from the first iteration.

This paper is organized as follows: the proposed controller design is introduced in Section 2; the two stage neural network training is introduced in Section 3; stability analysis is presented in Section 4; the controller is implemented and evaluated by simulation in Section 5; Section 6 concludes the paper.

2 NEUROADAPTIVE CONTROL OF INDIRECT DRIVE ROBOTS

2.1 Dynamical System Model

For a N DOF industrial robot, there are $2N$ generalized coordinates [2, 8, 9]:

$$\Theta = \begin{bmatrix} q & \theta \end{bmatrix}^T \in \mathbb{R}^{2N}$$

where $q \in \mathbb{R}^N$ refers to the link positions, and $\theta \in \mathbb{R}^N$ refers to the actuator positions.

The dynamic model of the system is

$$\begin{aligned} M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) &= d_\ell \\ &\quad + K(R^{-1}\theta - q) + D(R^{-1}\dot{\theta} - \dot{q}) \qquad (1a) \\ B\ddot{\theta} &= \tau + d_m \\ &\quad + R^{-1}\left[K(q - R^{-1}\theta) + D(\dot{q} - R^{-1}\dot{\theta})\right] \qquad (1b) \end{aligned}$$

where $M(q) \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times N}$ are the inertia matrices for link motion and actuator motion respectively; $C(q,\dot{q})\dot{q} \in \mathbb{R}^N$ is Coriolis and centrifugal force; $g(q) \in \mathbb{R}^N$ is the gravity term; $\tau \in \mathbb{R}^N$ is torque generated by actuators; $K \in \mathbb{R}^{N\times N}$ and $D \in \mathbb{R}^{N\times N}$ are diagonal matrices representing stiffness and damping of the transmission units respectively; $R \in \mathbb{R}^{N\times N}$ is diagonal matrix representing gear ratio; $d_\ell \in \mathbb{R}^N$ and $d_m \in \mathbb{R}^N$ are disturbances applied the links and actuators respectively. The input to the system is the actuator torque τ, and output of the system is the link position q.

The disturbances d_ℓ and d_m include complex friction, transmission error mentioned in [10, 11], and actuator-link interaction mentioned in [2]. It is difficult to build a parametric model for all the disturbances, but in general, they could be modelled as $d_\ell \approx d_\ell(q,\dot{q},\theta,\dot{\theta})$, $d_m \approx d_m(q,\dot{q},\theta,\dot{\theta})$.

2.2 Backstepping Control

Backstepping control, as shown in [12], is a control method that is developed for nonlinear dynamical systems with recursive structure. The design of backstepping control involves designing controllers that progressively stabilize a series of subsystems.

An indirect drive robot has such kind of recursive structure. The first subsystem is expressed as (1a), and the second subsystem is expressed as (1b). Design of backstepping control of indirect drive robot consists of two steps. The first step designs the control law for (1a) with the transmission torque $y = K(R^{-1}\theta - q) + D(R^{-1}\dot{\theta} - \dot{q})$ as input, and the second step designs the control law for (1b) with motor torque τ as input. In the first step, the designed controller stabilizes the trajectory tracking error to 0. In the second step, the designed controller stabilizes the transmission torque error to 0.

First step: Letting the reference trajectory of an indirect drive robot be $q_d, \dot{q}_d, \ddot{q}_d$, the trajectory tracking error is defined as

$$e = q_d - q \qquad (2)$$

According to [13], a filtered-error term r can be defined as

$$r = \dot{e} + K_p e \qquad (3)$$

where $K_p \in \mathbb{R}^{N\times N}$ is a positive definite gain matrix with the minimum singular value $\sigma_{\min}(K_p) > 0$. Stabilizing tracking error e is equivalent to stabilizing filtered error r since

$$\|e\| \leq \frac{\|r\|}{\sigma_{\min}(K_p)} \qquad (4)$$

In order to stabilize tracking error e, a desired transmission torque y_d can be designed as [13]

$$y_d = K_r r + M(q)(\ddot{q}_d + K_p\dot{e}) + C(q,\dot{q})(\dot{q}_d + K_p e) + g(q) - d_\ell$$

where $K_r \in \mathbb{R}^{N\times N}$ is a positive definite gain matrix.

Second step: The error between desired transmission torque y_d and the actual interaction torque through flexible transmission unit y is

$$s = y_d - y \qquad (5)$$

In order to design a control law that stabilize the transmission torque error s to 0, we first construct a Lyapunov function L,

$$L = \frac{1}{2}r^T M(q) r + \frac{1}{2}s^T A s$$

where $A = BRD^{-1}$. Since B, R, D are diagonal, positive definite matrices, A is positive definite.

The time derivative of the filtered error in the first step:

$$M(q)\dot{r} = \underbrace{M(q)(\ddot{q}_d + K_p\dot{e}) + C(q,\dot{q})(\dot{q}_d + K_p e) + g(q) - d_\ell}_{f} \\ -C(q,\dot{q})r + \underbrace{y_d - y}_{s} - y_d$$

The time derivative of the transmission torque error:

$$A\dot{s} = \underbrace{A\left[\dot{y}_d - K(R^{-1}\dot{\theta} - \dot{q})\right] + BR\ddot{q} + R^{-1}y - d_m}_{h} - \tau$$

Substituting y_d in the first step, and the time derivatives above. The time derivative of L is:

$$\begin{aligned}\dot{L} &= \tfrac{1}{2}r^T\dot{M}(q)r + r^T M(q)\dot{r} + s^T A\dot{s} \\ &= \tfrac{1}{2}r^T\dot{M}(q)r + r^T[f - C(q,\dot{q})r + s - y_d] + s^T(h - \tau) \\ &= r^T(f - y_d) + s^T(h + r - \tau) \\ &= -r^T K_r r + s^T(h + r - \tau)\end{aligned}$$

Letting $\tau = h + r + K_s s$, where $K_s \in \mathbb{R}^{N\times N}$ is a positive definite gain matrix. The time derivative of L is then negative definite.

$$\dot{L} = -r^T K_r r - s^T K_s s < 0$$

According to Lyapunov stability theory, the dynamical system is asymptotically stable, thus $\lim_{t\to\infty} r = 0$, $\lim_{t\to\infty} s = 0$. According to (4), $\lim_{t\to\infty} e = 0$.

To sum up, backstepping controller can be designed based on an accurate model of the system as

$$y_d = f + K_r r \tag{6a}$$
$$\tau = h + r + K_s s \tag{6b}$$

In this approach, two nonlinear functions f and h are representing the physical model of the dynamic system.

2.3 NN Based Adaptive Backstepping Control

The ideal backstepping controller (6a) and (6b) is impractical since the exact f and h terms are not available due to the complexity of any real physical system. Moreover, though $\dot{y}_d$ and $\ddot{q}$ required in the calculation of h may be estimated using the system model or by finite difference, the estimation could be difficult due to noise. One way to accommodate the uncertainty in f and h is to add robust feature to the controller, but this approach may not be able to make tracking error small if large uncertainty exists. Another way is to use an auxiliary model that approximates the modeling error by an artificial NN.The backstepping controller can then be designed as

$$y_d = f_n + \hat{f} + K_r r \tag{7a}$$
$$\tau = h_n + \hat{h} + r + K_s s \tag{7b}$$

where f_n and h_n are the nominal system model terms obtained using computer aided design software. $\hat{f}$ and $\hat{h}$ are the auxiliary model terms that approximate the difference between the actual system and the nominal model. The difference includes estimation errors in the inertia parameters of the robot, estimation error in the transmission units stiffness and damping parameters, unmodeled complex frictions, and transmission errors.

Radial basis function (RBF) network, also known as the functional-link neural network (FLNN) in [13], is chosen to build this auxiliary model. The reason to use RBF network is that RBF neural network has the ability to approximate an arbitrary nonlinear function with very simple structure (only one hidden layer), as shown in [14].

The terms $\hat{f}$ and $\hat{h}$ can be written as functions of X, where $X \equiv \left[q_d, \dot{q}_d, \ddot{q}_d, q, \theta, \dot{q}, \dot{\theta}\right]^T$ is the augmented state that includes the reference trajectory $\{q_d, \dot{q}_d, \ddot{q}_d\}$. The auxiliary model can then be formulated as

$$\begin{aligned}\hat{f}(X) &= \kappa_1 \sum_{i=1}^{U} w_1^i \phi_i(X) = \kappa_1 W_1^T \Phi(X) \\ \hat{h}(X) &= \kappa_2 \sum_{i=1}^{U} w_2^i \phi_i(X) = \kappa_2 W_2^T \Phi(X)\end{aligned} \tag{8}$$

where $\kappa_1 \in \mathbb{R}$ and $\kappa_2 \in \mathbb{R}$ are two constant parameters that scale the neural network weights. U is the number of neurons used in the RBF network, and $\Phi(X) = \left[\phi_1(X), \cdots, \phi_U(X)\right]^T$ is the vector of activation functions. $W_1, W_2 \in \mathbb{R}^{U\times N}$ are scaled weights of the neural networks, where the i^{th} column of the transposed weight matrices $W_1^T, W_2^T \in \mathbb{R}^{N\times U}$ are w_1^i and w_2^i.

Gaussian radial basis function is one common choice for the activation function in the RBF network. A Gaussian radial basis function used in the i^{th} neuron can be formulated as

$$\phi_i(x) = \exp\left\{-\frac{1}{2}(x - \mu_i)^T \Lambda_i^{-1}(x - \mu_i)\right\} \tag{9}$$

where $\mu_i \in \mathbb{R}^n$ is the center of the Gaussian radial basis function $\phi_i(x)$, and $\Lambda_i \in \mathbb{R}^{n\times n}$ can be called the width parameter. Choosing the center and width of the Gaussian radial basis function $\phi_i(x)$ will be introduced in the section 3 as the initial training

stage. After the center and width parameters are determined, the weights of RBF network can be trained using adaptive control. The adaptation law is designed as

$$\dot{W}_1 = F_1 \kappa_1 \Phi(X) r^T - \gamma_1 F_1 W_1 \tag{10a}$$

$$\dot{W}_2 = F_2 \kappa_2 \Phi(X) s^T - \gamma_2 F_2 W_2 \tag{10b}$$

where $F_1 \in \mathbb{R}^{U\times U}$ and $F_2 \in \mathrm{R}^{U\times U}$ are positive definite gain matrices, $\gamma_1 \in \mathbb{R}$, and $\gamma_2 \in \mathbb{R}$ are two extra gains. The uniform ultimate boundedness, which has once been introduced by [13], will be proved in section 4 for both tracking error and neural network weights estimation error.

3 Two Stage Training Approach for Neural Network

The training of a RBF network using Gaussian radial basis functions can be divided into two stages [15]. The first stage determines the placements of the localized units, i.e. Gaussian units in input space. The second stage then determines the weights of a RBF network. In this paper, these two stages are called initial training stage and online training stage.

3.1 Initial Training Stage

The centers of the Gaussian radial basis functions of a RBF network should be uniformly and densely distributed in the domain of function to guarantee a small approximation error [13]. The width can then be chosen to be the maximum distance between adjacent centers. However, this is hard to realize by simple discretization if the domain of function has high dimensionality because too many neurons/radial basis functions are required to cover the function domain. For example, in section 2.3, the input to the RBF network X could be a 42 dimensional vector if $N = 6$. Even only 2 levels are used for the discretization of each dimension, the required neuron number should be $2^{42} \approx 4.3980 \times 10^{12}$, which is even larger than the number of neurons in a human brain.To avoid this problem, an alternative data-driven approach is proposed in this section.

Since any trajectory of a robot can be parametrized by time as $X = X(t)$, the domain of function can be limited in a one-dimensional manifold in the high dimensional function domain. Instead of choosing centers in the high dimensional function domain, the centers can be determined in the low dimensional manifold, as shown in Fig.2. The required number of neurons can then be reduced.The center and width parameters can be first determined in the low dimensional manifold using clustering approaches like k-means, as shown in [16], then transfer back to the high dimensional space.

Suppose the dimension of the augmented state X is n, then an experiment data set that contains H data points can be denoted as a $H \times n$ matrix as $X_S \equiv [X_1, X_2, \cdots, X_H]^T$. Principle

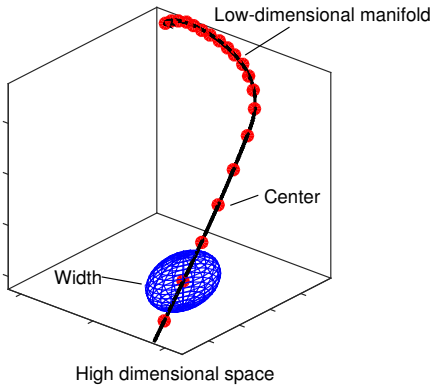

Figure 2. Distributing neurons / radial basis functions in the low-dimensional manifold. Line: low-dimensional manifold. Points: centers of radial basis functions. Ellipsoid: width of radial basis functions.

component analysis (PCA) can be implemented for dimension reduction [17]. Let the singular value decomposition of X_S be

$$X_S = PSQ^T = \sum_{i=1}^{n} p_i \rho_i \nu_i^T \tag{11}$$

where $P = [p_1, \cdots, p_H]$ is an $H \times H$ orthogonal matrix, $Q = [\nu_1, \cdots, \nu_n]$ is an $n \times n$ orthogonal matrix, and S is an $H \times n$ diagonal matrix with $S[i,i] = \rho_i$, where ρ_i is the i-th singular value of X_S. Since X_S is actually representing a low-dimensional manifold, the first $k(k < n)$ singular values will dominate, and $\forall i > k, \rho_i \approx 0$. Thus the data set can be well approximated by a low-rank approximation $\hat{X}_{Sk}$, as in [18],

$$\hat{X}_{Sk} = \sum_{i=1}^{k} p_i \rho_i \nu_i^T \tag{12}$$

Though this approximation is only linear, k could still be much smaller than n. This approximation projects all data points approximately to a hyper plane spanned by $\{\nu_1, \cdots, \nu_k\}$. The center and width of the Gaussian radial basis functions can be designed on the hyper plane then. The centers can be designed to be uniformly distributed along the projection of the low dimensional manifold on the hyper plane, and the widths can be designed to be constants. Suppose there are U neurons in the neural network. Let the $\{\mu_{p1}, \mu_{p2}, \cdots, \mu_{pU}\}$ be the coordinates of the centers of the Gaussian radial basis functions on the hyper plane, and the corresponding width parameters on the hyper plane be $\{\Lambda_{p1}, \Lambda_{p2}, \cdots, \Lambda_{pU}\}$. Let $Q_p \equiv [\nu_1, \nu_2, \cdots, \nu_k]$. The centers $\{\mu_1, \cdots, \mu_U\}$ and the widths $\{\Lambda_1, \cdots, \Lambda_U\}$ of the Gaussian radial basis functions in the original n-dimensional space are

calculated as

$$\begin{aligned}\{\mu_1,\cdots,\mu_U\} &= \{Q_p\mu_{p1},\cdots,Q_p\mu_{pU}\}\\ \{\Lambda_1,\cdots,\Lambda_U\} &= \{Q_p\Lambda_{p1}Q_p^T,\cdots,Q_p\Lambda_{pU}Q_p^T\}\end{aligned} \tag{13}$$

3.2 Online Training Stage

After the initial training stage, the vector of activation functions $\Phi(X)$ is determined. The weights W_1 and W_2 can then be learned to minimize the difference between the nominal model and the actual system.

The optimal RBF network weights W_1^* and W_2^*, which minimize the model difference can be defined as

$$\begin{aligned}W_1^* &= \arg\min_{W_1}(\sup_X\|f(X)-f_n(X)-\kappa_1 W_1^T\Phi(X)\|)\\ W_2^* &= \arg\min_{W_2}(\sup_X\|h(X)-h_n(X)-\kappa_2 W_2^T\Phi(X)\|)\end{aligned} \tag{14}$$

Since the actual dynamic system model f and h are not available, no supervised learning technique can be used to train this neural network. Instead of using supervised learning techniques, the RBF network weights W_1 and W_2 are trained using adaptive control approach as (10a) and (10b) in this paper. It will be proved in section 4 that the network weights are uniformly ultimately bounded.

4 Stability Analysis

This section shows the uniform ultimate boundedness of both trajectory tracking error and the neural network weights. This can be proved by showing the uniform ultimate boundedness of the filtered error r and the weight difference $\tilde{W}_1 = W_1^* - W_1, \tilde{W}_2 = W_2^* - W_2$.

We introduce three assumptions. a) The domains of f is compact and simply connected; b) the domain of h is compact and simply connected; and c) f and h are continuous functions. According to [13, 14], the universal approximation property of radial basis function networks holds. This suggests that the optimal approximation error should be bounded within the domains of the functions, as

$$\begin{aligned}\|\varepsilon_1^*\| &= \|f-f_n-\kappa_1 W_1^{*T}\Phi(x)\| \le \varepsilon_{N1}\\ \|\varepsilon_2^*\| &= \|h-h_n-\kappa_2 W_2^{*T}\Phi(x)\| \le \varepsilon_{N2}\end{aligned} \tag{15}$$

where ε_1^* and ε_2^* are the optimal approximation errors, and ε_{N1}, ε_{N2} are the upper bounds of ε_1^* and ε_2^*. Furthermore, $\kappa_1 W_1^*$ and $\kappa_2 W_2^*$ can be chosen to be constant and bounded matrices, as

$$\begin{aligned}\kappa_1\|W_1^*\|_F &\le W_{B1}\\ \kappa_2\|W_2^*\|_F &\le W_{B2}\end{aligned} \tag{16}$$

where $\|\cdot\|_F$ is the Frobenius norm; W_{B1} and W_{B2} are upper bounds of the norm of neural network weights.

To analyse the stability, a Lyapunov function candidate can be chosen as

$$\begin{aligned}V = &\tfrac{1}{2}r^T M(q)r + \tfrac{1}{2}s^T As + \tfrac{1}{2}\mathrm{tr}\{\tilde{W}_1^T F_1^{-1}\tilde{W}_1\}\\ &+\tfrac{1}{2}\mathrm{tr}\{\tilde{W}_2^T F_2^{-1}\tilde{W}_2\}\end{aligned} \tag{17}$$

where $M(q)$ is the inertia matrix, r, s, and A are defined in section 2.2, F_1 and F_2 are defined in the adaptation law (10a), (10b). Since the optimal neural network weights W_1^* and W_2^* are constant matrices,

$$\dot{\tilde{W}}_1 = -\dot{W}_1 \tag{18a}$$

$$\dot{\tilde{W}}_2 = -\dot{W}_2 \tag{18b}$$

With the proposed controller (7a), (7b), the proposed adaptation law (10a), (10b), and (18a), (18b), the time derivative of the Lyapunov function candidate is

$$\begin{aligned}\dot{V} = &\tfrac{1}{2}r^T\dot{M}(q)r + r^T M(q)\dot{r} + s^T A\dot{s}\\ &+\mathrm{tr}\{\tilde{W}_1^T F_1^{-1}\dot{\tilde{W}}_1\} + \mathrm{tr}\{\tilde{W}_2^T F_2^{-1}\dot{\tilde{W}}_2\}\\ = &\tfrac{1}{2}r^T[\dot{M}(q)-2C(q,\dot{q})]r\\ &-r^T K_r r + r^T\varepsilon_1^* + \kappa_1 r^T\tilde{W}_1^T\Phi(X)\\ &-s^T K_s s + s^T\varepsilon_2^* + \kappa_2 s^T\tilde{W}_2^T\Phi(X)\\ &-\mathrm{tr}\{\kappa_1\tilde{W}_1^T\Phi(X)r^T - \gamma_1\tilde{W}_1^T W_1\}\\ &-\mathrm{tr}\{\kappa_2\tilde{W}_2^T\Phi(X)r^T - \gamma_2\tilde{W}_2^T W_2\}\end{aligned}$$

Using the skew symmetric property of $\dot{M}(q)-2C(q,\dot{q})$, the linearity of trace, and the property $\mathrm{tr}(AB)=\mathrm{tr}(BA)$, the time derivative of V can be further manipulated as

$$\begin{aligned}\dot{V} = &-r^T K_r r - s^T K_s s + r^T\varepsilon_1^* + s^T\varepsilon_2^* - \gamma_1\mathrm{tr}\{\tilde{W}_1^T\tilde{W}_1\}\\ &+\gamma_1\mathrm{tr}\{W_1^{*T}\tilde{W}_1\} - \gamma_2\mathrm{tr}\{\tilde{W}_2^T\tilde{W}_2\} + \gamma_2\mathrm{tr}\{W_2^{*T}\tilde{W}_2\}\\ = &-\begin{bmatrix}r\\ s\end{bmatrix}^T\begin{bmatrix}K_r & 0\\ 0 & K_s\end{bmatrix}\begin{bmatrix}r\\ s\end{bmatrix} + \begin{bmatrix}r\\ s\end{bmatrix}^T\begin{bmatrix}\varepsilon_1^*\\ \varepsilon_2^*\end{bmatrix} - \gamma_1\mathrm{tr}\{\tilde{W}_1^T\tilde{W}_1\}\\ &+\gamma_1\mathrm{tr}\{W_1^{*T}\tilde{W}_1\} - \gamma_2\mathrm{tr}\{\tilde{W}_2^T\tilde{W}_2\} + \gamma_2\mathrm{tr}\{W_2^{*T}\tilde{W}_2\}\end{aligned}$$

Let $W_i^v = [w_i^{1T},\cdots,w_i^{UT}]^T$ be the vectorized $W_i (i=1,2)$, where $W_i^T\in\mathbb{R}^{N\times U}$, w_i^j is the j^{th} column of W_i^T. The time derivative of the V can be written as

$$\dot{V} = \begin{bmatrix}r\\ s\\ \tilde{W}_1^v\\ \tilde{W}_2^v\end{bmatrix}^T\begin{bmatrix}\varepsilon_1^*\\ \varepsilon_2^*\\ \gamma_1 W_1^{*v}\\ \gamma_2 W_2^{*v}\end{bmatrix} - \begin{bmatrix}r\\ s\\ \tilde{W}_1^v\\ \tilde{W}_2^v\end{bmatrix}^T\begin{bmatrix}K_r & 0 & 0 & 0\\ 0 & K_s & 0 & 0\\ 0 & 0 & \gamma_1 I & 0\\ 0 & 0 & 0 & \gamma_2 I\end{bmatrix}\begin{bmatrix}r\\ s\\ \tilde{W}_1^v\\ \tilde{W}_2^v\end{bmatrix} \tag{19}$$

According to (15), the optimal neural network approximation error is bounded. According to (16), W_i^{*v} are bounded since $\|W_i\|_F = \sqrt{W_i^{vT} W_i^v} = \|W_i^v\| (i = 1,2)$. Then

$$\begin{aligned} &\left\| [\varepsilon_1^{*T}, \varepsilon_2^{*T}, \gamma_1 W_1^{*vT}, \gamma_2 W_2^{*vT}]^T \right\| \\ &= \sqrt{\|\varepsilon_1^*\|^2 + \|\varepsilon_2^*\|^2 + \gamma_1^2 \|W_1^{*v}\|^2 + \gamma_2^2 \|W_2^{*v}\|^2} \\ &\leq \sqrt{\varepsilon_{N1}^2 + \varepsilon_{N2}^2 + \gamma_1^2 W_{B1}^2 / \kappa_1^2 + \gamma_2^2 W_{B2}^2 / \kappa_2^2} \\ &\triangleq b_\varepsilon \end{aligned}$$

Let σ_l be

$$\sigma_l = \min_{\|x\|=1} x^T \begin{bmatrix} K_r & 0 & 0 & 0 \\ 0 & K_s & 0 & 0 \\ 0 & 0 & \gamma_1 I & 0 \\ 0 & 0 & 0 & \gamma_2 I \end{bmatrix} x > 0$$

Let $\eta \triangleq [r^T, s^T, \tilde{W}_1^{vT}, \tilde{W}_2^{vT}]^T$. Then from (19),

$$\begin{aligned} \dot{V} &\leq -\sigma_l \|\eta\|^2 + b_\varepsilon \|\eta\| \\ &= \|\eta\| (b_\varepsilon - \sigma_l \|\eta\|) \end{aligned}$$

Therefore

$$\dot{V} \leq -\delta \|\eta\| < 0, \forall \|\eta\| \geq (b_\varepsilon + \delta)/\sigma_l > 0 \tag{20}$$

where $\delta > 0$ can be any small number.

In addition, according to [13], $M(q)$ is positive definite and bounded, thus the Lyapunov function candidate V can be bounded by quadratic functions.

$$0 < \sigma_1 \|\eta\|^2 \leq V \leq \sigma_2 \|\eta\|^2 \tag{21}$$

where σ_1 is a positive number smaller than one half of the minimum singular values of $M(q)$, A, F_1^{-1}, F_2^{-1}, and σ_2 is a positive number larger than one half of the maximum singular values of the four gain matrices.

Ref to [13] and [19], uniform ultimate boundedness of η can be guaranteed by (20) and (21). Thus there exists t_0, such that $\forall t \geq t_0, \|\eta\| \leq \sqrt{\frac{\sigma_2}{\sigma_1}} \frac{b_\varepsilon + \delta}{\sigma_l}$. Since δ can be any small number, this inequality is reduced to $\|\eta\| \leq \sqrt{\frac{\sigma_2}{\sigma_1}} \frac{b_\varepsilon}{\sigma_l}$ eventually. This upper bound of η can be made arbitrarily small by increasing K_r, K_s, κ_1, κ_2, and decreasing γ_1, γ_2. Thus the trajectory tracking error e and neural network weight estimation error $\tilde{W}_1, \tilde{W}_2$ are uniformly ultimately bounded as $(i = 1,2)$

$$\|e\| \leq \frac{\|r\|}{\sigma_{\min}(K_p)} \leq \frac{\|\eta\|}{\sigma_{\min}(K_p)} \leq \frac{1}{\sigma_{\min}(K_p)} \sqrt{\frac{\sigma_2}{\sigma_1}} \frac{b_\varepsilon}{\sigma_l} \tag{22a}$$

$$\|\tilde{W}_i\|_F \leq \|\eta\| \leq \sqrt{\frac{\sigma_2}{\sigma_1}} \frac{b_\varepsilon}{\sigma_l} \tag{22b}$$

5 Simulation Results

The proposed controller is implemented to control a 6-axis robot in simulation. In the simulation, rigid body dynamics of the robot, joint flexibility, motor dynamics, complex friction are all taken into account. The reference trajectory in the simulation is designed to have high velocity and acceleration. The reference trajectory is illustrated in Fig. 3, and the acceleration is shown in Fig. 4.

A benchmark controller for industrial robot and the proposed controller are implemented in the simulation. Nominal dynamical model are used in both controllers. The modelling error includes: a) link inertia and center-of-gravity; b) friction; c) stiffness and damping of transmission units.

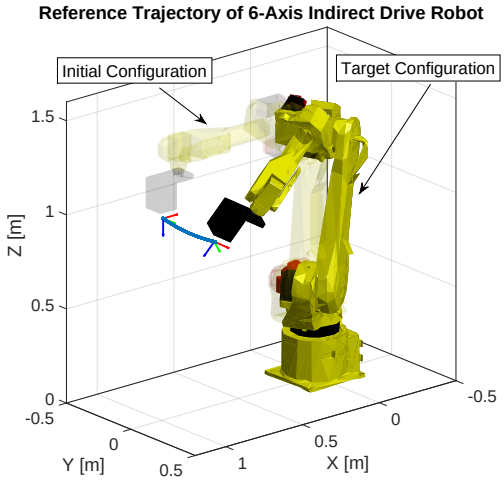

Figure 3. Reference trajectory of 6-axis indirect drive robot

The benchmark controller consists of two parts: torque feedforward control part and feedback control part. The feedforward part utilizes a nominal rigid body dynamics model of the robot to compensate the nonlinear dynamics of the 6-axis robot. The feedback control part utilizes a well tuned proportional−integral−derivative (PID) controller. The bench-

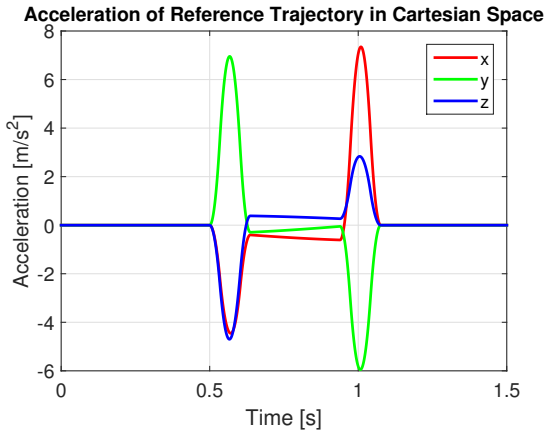

Figure 4. Acceleration of the reference trajectory in Cartesian space

mark controller has the form

$$\tau = R^{-1}\left[(M_n(q_d)+B_nR^2)\ddot{q}_d + C_n(q_d,\dot{q}_d)\dot{q}_d + g_n(q_d)\right] + K_P(Rq_d-\theta) + K_D(R\dot{q}_d-\dot{\theta}) + K_I\int_0^t (Rq_d-\theta)$$

where the subscript $_n$ denotes the nominal model. K_P, K_D, and K_I are the PID gains.

The proposed neuroadaptive controller is designed as (7a), (7b). Two iterations are required for this approach. The first iteration is mainly used to collect data for designing center and width of each Gaussian basis function in the RBF network. In the first iteration, only the nominal model is used and there is no auxiliary model available, i.e., in the first iteration,

$$y_d = f_n + K_r r$$
$$\tau = h_n + r + K_s s$$

For the second iteration of the proposed controller, RBF network is used to build the auxiliary model as (8). In the second iteration, both nominal model and auxiliary model are used in the controller,

$$y_d = f_n + \kappa_1 W_1^T \Phi(X) + K_r r$$
$$\tau = h_n + \kappa_2 W_2^T \Phi(X) + r + K_s s$$

The data from the first iteration is used in the initial training stage before running the second iteration, i.e. determining the center and width parameters for the RBF network. 50 neurons are used in this neural network. The 2-D projection of center and width of the radial basis functions are shown in Fig. 5. The online training stage takes place during the second iteration. The neural network weights are trained adaptively as designed in (10a) and (10b).

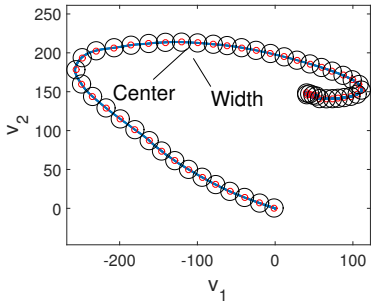

Figure 5. Low dimensional manifold from experiment data with designed center and width of radial basis functions.

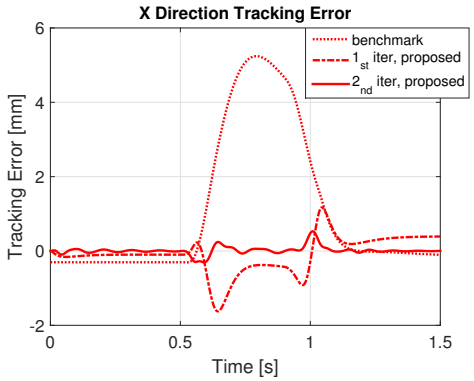

Figure 6. Cartesian space tracking error in X direction

The trajectory tracking for the benchmark controller and the proposed controller are shown in Fig. 6, 7, and 8. Due to modelling error, the trajectory tracking error is large for the benchmark controller and the first iteration of the proposed controller. But the error is effectively reduced in the second iteration.

6 Conclusion

In this paper, a neural network based adaptive backstepping control approach is proposed to improve trajectory tracking accuracy of indirect drive robots. Artificial neural network is used to approximate the difference of actual system and the physical model used for control. A two stage training approach, which consists of an offline data-driven initial training stage and an online training stage, was proposed to train the radial basis function network used in the controller. In the initial training stage,

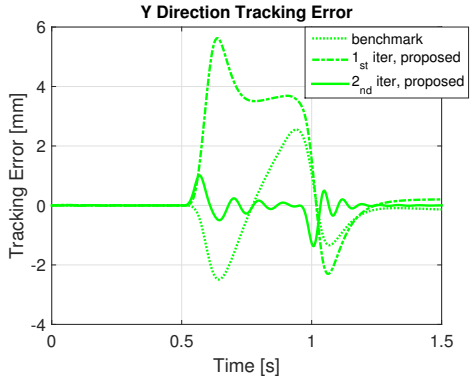

Figure 7. Cartesian space tracking error in Y direction

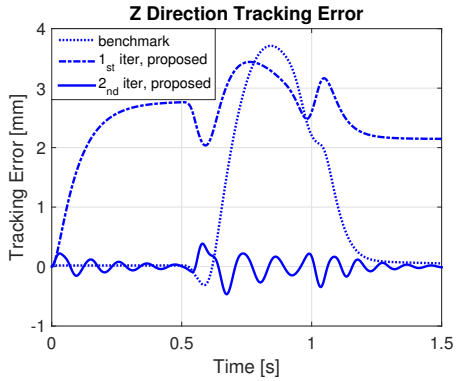

Figure 8. Cartesian space tracking error in Z direction

a model based backstepping controller is first implemented for data collection. The center and width parameters of neurons are then designed based on the motion data. In the second stage, the same trajectory is used and the weights of neural network are tuned online to improve the controller performance. Comparing to other learning control techniques like iterative learning control, the approach proposed in this paper requires only at most two iterations for a specific trajectory, which is more efficient. It is proved that the trajectory tracking error and the neural network weight estimation error are uniform ultimate bounded. The effectiveness of the proposed controller is demonstrated using simulation on a six axis indirect drive robot.

REFERENCES

[1] Litak, G., and Friswell, M. I., 2003. "Vibration in gear systems". *Chaos, Solitons & Fractals,* ***16***(5), pp. 795–800.

[2] De Luca, A., and Book, W., 2008. "Robots with flexible elements". In *Springer Handbook of Robotics*. Springer, pp. 287–319.

[3] De Luca, A., and Lucibello, P., 1998. "A general algorithm for dynamic feedback linearization of robots with elastic joints". In Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, Vol. 1, IEEE, pp. 504–510.

[4] Spong, M., Khorasani, K., and Kokotovic, P. V., 1987. "An integral manifold approach to the feedback control of flexible joint robots". *IEEE Journal on Robotics and Automation,* ***3***(4), pp. 291–300.

[5] De Luca, A., 2000. "Feedforward/feedback laws for the control of flexible robots". In Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, Vol. 1, IEEE, pp. 233–240.

[6] Chen, W., and Tomizuka, M., 2014. "Dual-stage iterative learning control for mimo mismatched system with application to robots with joint elasticity". *Control Systems Technology, IEEE Transactions on,* ***22***(4), pp. 1350–1361.

[7] Wang, C., Zhao, Y., Chen, Y., and Tomizuka, M., 2015. "Nonparametric statistical learning control of robot manipulators for trajectory or contour tracking". *Robotics and Computer-Integrated Manufacturing,* ***35***, pp. 96–103.

[8] Spong, M. W., 1987. "Modeling and control of elastic joint robots". *Journal of dynamic systems, measurement, and control,* ***109***(4), pp. 310–318.

[9] de Wit, C. C., Siciliano, B., and Bastin, G., 2012. *Theory of robot control*. Springer Science & Business Media.

[10] Gandhi, P. S., 2001. "Modeling and control of nonlinear transmission attributes in harmonic drive systems". PhD thesis, Rice University.

[11] Han, C.-H., Wang, C.-C., and Tomizuka, M., 2008. "Suppression of vibration due to transmission error of harmonic drives using peak filter with acceleration feedback". In Advanced Motion Control, 2008. AMC'08. 10th IEEE International Workshop on, pp. 182–187.

[12] Kokotovic, P. V., 1992. "The joy of feedback: nonlinear and adaptive". *IEEE Control systems,* ***12***(3), pp. 7–17.

[13] Lewis, F., Jagannathan, S., and Yesildirak, A., 1998. *Neural network control of robot manipulators and non-linear systems*. CRC Press.

[14] Park, J., and Sandberg, I. W., 1991. "Universal approximation using radial-basis-function networks". *Neural computation,* ***3***(2), pp. 246–257.

[15] Fritzke, B., 1994. "Fast learning with incremental rbf networks". *Neural processing letters,* ***1***(1), pp. 2–5.

[16] Friedman, J., Hastie, T., and Tibshirani, R., 2001. *The elements of statistical learning*, Vol. 1. Springer series in statistics Springer, Berlin.

[17] Jolliffe, I., 2002. *Principal component analysis*. Wiley Online Library.

[18] Calafiore, G., and El Ghaoui, L., 2014. *Optimization Models*. Control systems and optimization series. Cambridge University Press, October.

[19] Khalil, H. K., and Grizzle, J., 2002. *Nonlinear systems, vol. 3*, Vol. 8. Prentice hall Upper Saddle River, pp. 168–174.