Backstepping Intelligent Control Using Partially Known Model Applied to a Two-Link Flexible-Joint Robot Manipulator

Withit Chatlatanagulchai

Department of Mechanical Engineering, Faculty of Engineering,
Kasetsart University, Bangkok 10900, Thailand
fengwtc@ku.ac.th

Peter H. Meckl

School of Mechanical Engineering, College of Engineering,
Purdue University, West Lafayette 47906, USA
meckl@ecn.purdue.edu

Abstract

We propose a controller consisting of a backstepping structure, nonlinear damping control, a neuro-fuzzy intelligent system, and a plant in strict-feedback form. The neuro-fuzzy intelligent system is used to learn a portion of the unknown plant off-line. Nonlinear damping provides robustness against uncertainties in plant estimation and external disturbances, while a backstepping structure allows control inputs to be inserted into each subsystem. The proposed controller is applied to a two-link flexible-joint robot manipulator with good tracking results.

Keywords: Backstepping, Nonlinear Damping, Flexible-Joint Robot, Intelligent System, Neuro-Fuzzy System

Introduction

Model independent control is used to develop controllers independent at the mathematical model of the system being controlled. For example, a control, for the trajectory of a robot that does not use the mathematical model of the robot in the control algorithm. In [1] we present a three-layer neural network to learn unknown plant functions, in [2] a control algorithm was devised first before using neural networks to learn the unknown part of the control algorithm, and in [3] we incorporated a nonlinear observer into the control system.

The advantages of using model-independent control techniques are that the control algorithm does not depend on the accuracy of the plant model, the control algorithm can be implemented for other systems that have slightly different setup without having to redesign the controller or to identify the new system, and the control algorithm is suitable for controlling complicated systems, for example, nanosystems, where the plant model from physical laws is difficult to obtain.

However, model-based control techniques have another advantage over model-independent control techniques in that if the plant model used in the control algorithm is accurate, the former normally deliver better control performance and are especially suitable for fast movement, whereas it is almost impossible to have neural networks trained and be ready for such fast movement.

This motivates the idea of combining the advantages of both model-independent control techniques and model-based control techniques. There are two possible research directions. First, instead of initializing the neural network weights at zero values every time, we could incorporate of a pre-train in algorithm to train the neural network weights to start at more-meaningful initial values. Second, in the case where we can easily and accurately identify some parts of the plant model, we can incorporate this so-called partially known model into our control system and use an intelligent system to learn the unknown part of the plant model. For the latter case, some work has already been done and is presented here.

In this paper, the objective is that a plant to closely track a desired trajectory. Since the control algorithm is model-free, we do not use the plant's mathematical model in the algorithm. However, devising a control algorithm for the closed-loop system to be stable does require knowledge of the plant. We assume the actual plant to be a multi-input-multi-output nonlinear system in strict-feedback form. A portion of the plant is assumed perfectly known whereas the rest of the plant, typically more complex, is unknown. An adaptive neuro-fuzzy inference system (ANFIS) is used as the intelligent system to learn the unknown part of the plant model. The learning is performed off-line. The controller has a backstepping control [4] structure. The backstepping control allows control inputs to be virtually inserted into each subsystem. The control law for each subsystem contains proportional gain, cancellation terms, and a nonlinear damping term. The nonlinear damping term provides robustness against uncertainties that arise from the plant estimation as well as from external disturbance.

We applied the proposed controller to a two-link flexible-joint robot manipulator. Joint flexibility exists in most manipulators from driving components, joint material, and as a component to reduce joint damage from collision. The experiment conducted in [5] suggests that designers should consider joint flexibility in both modeling and control design because joint resonant frequencies, which are located within the control bandwidth, can be excited and cause severe oscillations.

Controller design of a two-link flexible-joint robot manipulator is challenging for two main reasons. First, its Euler-Lagrange model is much more complicated than the model of a rigid-joint or one-link flexible-joint robot manipulator. Second, the number of degrees of freedom is twice the number of control inputs. The control inputs do not directly act on the links. Instead, the control inputs directly act on the motors that connect to the links via flexible-joint dynamics. This results in the loss of some important structural properties that apply for rigid-joint robot manipulators, such as matching property between nonlinearities and the inputs, and passivity from inputs to link velocities.

The proposed controller suits the flexible-joint robot well in that complicated part of the model can be learned by the intelligent system and backstepping structure allows control inputs to go into each subsystem to handle uncertainties and nonlinearities. The controller delivers good tracking performance.

The paper is organized as follows. Section 2 describes the proposed controller. It starts with a discussion of the adaptive neuro-fuzzy inference system (ANFIS), which is the intelligent system used in the proposed controller. Then, the applicable plant model and the backstepping control design are presented. Boundedness of the tracking error is then proved. Section 3 contains the application of the proposed controller to a two-link flexible-joint robot manipulator. It starts with the robot model, followed by the off-line training of ANFIS and the tracking results. Section 4 presents conclusions.

Control System Design

1. Adaptive Neuro-Fuzzy Inference System

Neuro-fuzzy systems use neural network structure to provide fuzzy systems with automatic tuning ability. Both membership functions and fuzzy rules can be tuned based on specified performance objectives. Figure 1 depicts a kind of neuro-fuzzy system called an adaptive neuro-fuzzy inference system (ANFIS). ANFIS was introduced in [6] and was proved to be a universal approximator in [7]. Therefore, ANFIS can be used to estimate a smooth function $f\left(x_1, x_2, \ldots, x_i\right)$ with arbitrary accuracy.

Suppose there are j fuzzy rules. Layer 1 performs fuzzification by assigning a membership function to each input. If f_i is a function of $x_1,\ldots,x_i$, then $x_1,\ldots,x_i$ are inputs to the ANFIS. There are many types of membership functions. We use a bell-shaped function, which has the form

$$\mu_{i,j}(x_i) = 1/\left(1+\left(\left(x_i - c_{i,j}\right)/a_{i,j}\right)^{2b_{i,j}}\right),$$

where $a_{i,j}, b_{i,j}, c_{i,j}$ are premise parameters.

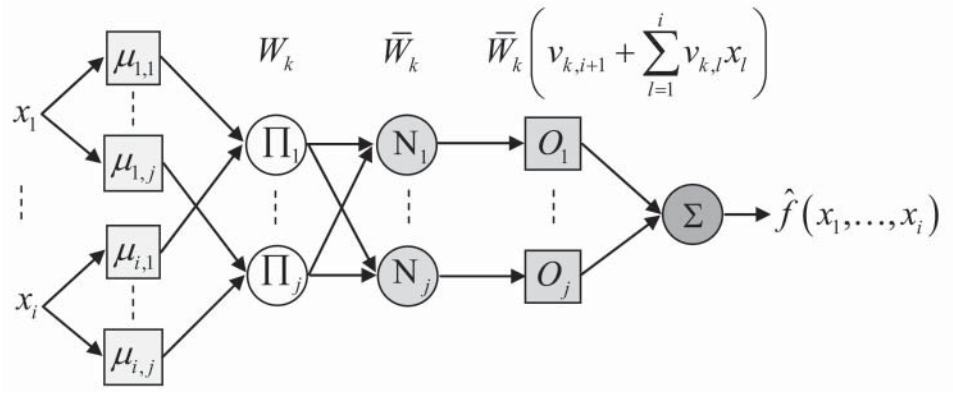

Figure 1: Diagram of an adaptive neuro-fuzzy inference system.

The output of Layer 2 represents the firing strength of each fuzzy rule by performing an "and" operation,

$$W_k = \prod_{l=1}^{i} \mu_{l,k}, \quad k = 1,\ldots,j.$$

Layer 3 normalizes firing strengths, with output

$$\bar{W}_k = W_k / \left(\sum_{l=1}^{j} W_l\right), \quad k = 1,\ldots,j.$$

Layer 4 represents the output of each fuzzy rule,

$$\bar{W}_k\left(v_{k,i+1}+\sum_{l=1}^{i} v_{k,l}x_l\right), \quad k = 1,\ldots,j,$$

where $v_k = \left[v_{k,1}, v_{k,2},\ldots, v_{k,i+1}\right] \in \mathbb{R}^{i+1}$ is a vector containing consequent parameters. Layer 5 performs defuzzification,

$$f = \sum_{k=1}^{j} \bar{W}_k\left(v_{k,i+1}+\sum_{l=1}^{i} v_{k,l}x_l\right).$$

The learning process can be done online by adjusting the premise and/or consequent parameters to minimize the estimation error.

2. *Applicable Plant*

Consider a system in the strict-feedback form

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 + h_1(x_1)\Delta_1(x_1), \\
\dot{x}_2 &= f_2(\bar{x}_2) + g_2(\bar{x}_2)x_3 + h_2(\bar{x}_2)\Delta_2(\bar{x}_2), \\
&\vdots \\
\dot{x}_m &= f_m(\bar{x}_m) + g_m(\bar{x}_m)u + h_m(\bar{x}_m)\Delta_m(\bar{x}_m),
\end{aligned} \tag{1}$$

where $x_i \in \mathbb{R}^n, i = 1,2,\ldots,m$ denote state vectors of the system, $u \in \mathbb{R}^n$ is vector of control inputs, $f_i \in \mathbb{R}^n$, $g_i \in \mathbb{R}^{n\times n}, i = 1,2,\ldots,m$ are vectors and matrices of smooth nonlinear functions, $h_i \in \mathbb{R}^{n\times n}$ are matrices of known smooth nonlinear functions, $\Delta_i \in \mathbb{R}^{q_i}$ are vectors of unknown smooth uncertainties, which are uniformly bounded, and $\bar{x}_i = \{x_1, x_2, \ldots, x_i\}$ denote sets of state vectors.

Typically, for the systems in which we are interested, which include robot manipulators and pendulum, the f_i are much more complicated than g_i. Therefore, it is assumed that g_i are known exactly whereas f_i are unknown. The bounds of Δ_i are not required in the controller algorithm. Uncertainties are handled by nonlinear damping.

To estimate the unknown functions f_i, intelligent systems such as neural networks and fuzzy logic, can be used due to their universal approximator property. An advantage of using intelligent systems as estimators is that the structures of the unknown functions being estimated are not required in particular. We chose an adaptive network-based fuzzy inference system (ANFIS) due to the availability of a Matlab toolbox implementation. The estimation errors of f_i are treated as uncertainties and are handled by nonlinear damping terms in the controller.

Estimation of f_1 can be done as follows. Consider the equation of the first subsystem

$$\dot{x}_1 = f_1(x_1) + g_1(x_1)x_2 + h_1(x_1)\Delta_1(x_1).$$

The term Δ_1 is unknown but is assumed to be bounded. We then have

$$\dot{x}_1 - g_1(x_1)x_2 = f_1(x_1) + h_1(x_1)\Delta_1(x_1) \triangleq F_1(x_1).$$

$F_1(x_1)$ is to be estimated by ANFIS. Note that if $F_1(x_1)$ is a vector of more than one element, we need to use one ANFIS per element of $F_1(x_1)$. The training data for ANFIS consists of inputs and output of the mapping to be estimated. To estimate $F_1(x_1)$, the training input is x_1 and the training output is $\dot{x}_1 - g_1(x_1)x_2$. We then obtain $\hat{F}_1(x_1)$, which is the approximation of $F_1(x_1)$, by offline learning. Since ANFIS can approximate any smooth functions with arbitrary accuracy, we have $\left\|F_1(x_1) - \hat{F}_1(x_1)\right\| \le \varepsilon_{F_1}$, where ε_{F_1} is a constant. Similarly we can obtain $\hat{F}_i, \forall i = 2,\ldots,m.$ For example, to approximate $F_2(x_1, x_2)$, the training inputs are x_1, x_2, and the training output is $\dot{x}_2 - g_2(x_1, x_2)x_3$.

3. *Backstepping and Nonlinear Damping Control*

The virtual control law of the first subsystem is given by

$$x_{2d} = g_1^{-1}[-\hat{F}_1 + \dot{x}_{1d} - c_1 z_1 - h_1 k_1 h_1^T z_1],$$

where $\hat{F}_1$ is the estimate of F_1 and $z_1 = x_1 - x_{1d}$ is the tracking error. $-h_1 k_1 h_1^T z_1$ is the nonlinear damping term. $c_i \in \mathbb{R}^{n\times n}$, $c_i = c_i^T > 0$, $i = 1,\ldots,m$, and $k_i \in \mathbb{R}^{n\times n}$, $k_i = k_i^T > 0$, $i = 1,\ldots,m.$ Introducing $z_2 = x_2 - x_{2d}$, the error dynamics of the first subsystem become

$$\begin{aligned}
\dot{z}_1 &= \dot{x}_1 - \dot{x}_{1d} \\
&= f_1 + g_1 x_2 + h_1\Delta_1 - \dot{x}_{1d} + g_1 x_{2d} - g_1 x_{2d} \\
&= F_1 + g_1 x_2 - \dot{x}_{1d} + g_1 g_1^{-1}[-\hat{F}_1 + \dot{x}_{1d} - c_1 z_1 - h_1 k_1 h_1^T z_1] - g_1 x_{2d} \\
&= F_1 - \hat{F}_1 + g_1 z_2 - c_1 z_1 - h_1 k_1 h_1^T z_1.
\end{aligned}$$

Let the virtual controls of the remaining subsystems be

$$x_{(i+1)d} = g_i^{-1}\left(-\hat{F}_i + \dot{x}_{id} - c_i z_i - h_i k_i h_i^T z_i - g_{i-1}^T z_{i-1}\right),$$

for $2 \le i \le m-1$, and, for the last subsystem

$$u = g_m^{-1}\left(-\hat{F}_m + \dot{x}_{md} - c_m z_m - h_m k_m h_m^T z_m - g_{m-1}^T z_{m-1}\right).$$

The error dynamics of the remaining subsystems can be found similarly to those of the first subsystem. We can express the error dynamics matrix form

$$\dot{z} = \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ M \\ \dot{z}_m \end{bmatrix} = \begin{bmatrix} -c_1 & g_1 & 0 & K & 0 \\ -g_1^T & -c_2 & g_2 & K & 0 \\ 0 & -g_2^T & -c_3 & K & 0 \\ M & M & M & O & g_{m-1} \\ 0 & 0 & 0 & -g_{m-1}^T & -c_m \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ M \\ z_m \end{bmatrix} + \begin{bmatrix} F_1 - \hat{F}_1 - h_1 k_1 h_1^T z_1 \\ F_2 - \hat{F}_2 - h_2 k_2 h_2^T z_2 \\ F_3 - \hat{F}_3 - h_3 k_3 h_3^T z_3 \\ M \\ F_m - \hat{F}_m - h_m k_m h_m^T z_m \end{bmatrix}.$$

Let the Lyapunov candidate be $V = 0.5 z^T z = 0.5 \sum_{i=1}^{m} \|z_i\|^2 > 0$. The derivative is given by

$$\dot{V} = \sum_{i=1}^{m} \left[-z_i^T c_i z_i + z_i^T \left(F_i - \hat{F}_i\right) - z_i^T h_i k_i h_i^T z_i \right].$$

Consider each component in the summation, we have

$$\begin{aligned}
&-z_i^T c_i z_i + z_i^T \left(F_i - \hat{F}_i\right) - z_i^T h_i k_i h_i^T z_i \\
&= -z_i^T c_i z_i + z_i^T h_i h_i^{-1} \left(F_i - \hat{F}_i\right) - z_i^T h_i k_i h_i^T z_i \\
&= -z_i^T c_i z_i - \left\{ h_i^T z_i - \frac{1}{2} k_i^{-1} \left[h_i^{-1} \left(F_i - \hat{F}_i\right) \right] \right\}^T k_i \left\{ h_i^T z_i - \frac{1}{2} k_i^{-1} \left[h_i^{-1} \left(F_i - \hat{F}_i\right) \right] \right\} \\
&+ \frac{1}{4} \left[h_i^{-1} \left(F_i - \hat{F}_i\right) \right]^T k_i^{-1} \left[h_i^{-1} \left(F_i - \hat{F}_i\right) \right] \\
&\le -z_i^T c_i z_i + \frac{1}{4} \left[h_i^{-1} \left(F_i - \hat{F}_i\right) \right]^T k_i^{-1} \left[h_i^{-1} \left(F_i - \hat{F}_i\right) \right] \\
&\le -\lambda_{\min}\{c_i\} \|z_i\|^2 + \frac{1}{4} \left[h_i^{-1} \left(F_i - \hat{F}_i\right) \right]^T k_i^{-1} \left[h_i^{-1} \left(F_i - \hat{F}_i\right) \right],
\end{aligned}$$

where $\lambda_{\min}\{c_i\}$ is minimum eigenvalue of c_i. Therefore, we have

$$\begin{aligned}
\dot{V} &= \sum_{i=1}^{m}\left[-z_i^T c_i z_i + z_i^T\left(F_i - \hat{F}_i\right) - z_i^T h_i k_i h_i^T z_i\right] \\
&\le -\sum_{i=1}^{m}\left(\lambda_{\min}\{c_i\}\|z_i\|^2\right) + \sum_{i=1}^{m}\left\{\frac{1}{4}\left[h_i^{-1}\left(F_i - \hat{F}_i\right)\right]^T k_i^{-1}\left[h_i^{-1}\left(F_i - \hat{F}_i\right)\right]\right\} \\
&\le -\Lambda_{\min}\{c_i\}\sum_{i=1}^{m}\left(\|z_i\|^2\right) + \sum_{i=1}^{m}\left(\frac{1}{4}\lambda_{\max}\left\{k_i^{-1}\right\}\left\|h_i^{-1}\left(F_i - \hat{F}_i\right)\right\|^2\right) \\
&\le -\Lambda_{\min}\{c_i\}\sum_{i=1}^{m}\left(\|z_i\|^2\right) + \sum_{i=1}^{m}\left(\frac{1}{4}\lambda_{\max}\left\{k_i^{-1}\right\}\left(\bar{h}_i^{-1}\varepsilon_{F_i}\right)^2\right) \\
&= -K_1 V + K_2,
\end{aligned}$$

where $\Lambda_{\min}\{c_i\} = \min_{1\le i\le m}\lambda_{\min}\{c_i\}$, $\left\|h_i^{-1}\right\| \le \bar{h}_i^{-1}$, $\forall i = 1,\ldots,m$, and K_1, K_2 are positive constants.

From Lemma 4.3 in [8], there exist class K_∞ functions α_1 and α_2 such that $\alpha_1\left(\|z\|\right) \le V \le \alpha_2\left(\|z\|\right)$. Choose some θ such that $0 < \theta < 1$. Then

$$\begin{aligned}
\dot{V} &\le -\theta K_1 V - (1-\theta) K_1 V + K_2 \\
&\le -\theta K_1 \alpha_1 - (1-\theta) K_1 V + K_2.
\end{aligned}$$

Choosing $W = \theta K_1 \alpha_1$ we see that

$$\dot{V} \le -W - (1-\theta) K_1 V + K_2.$$

Therefore if $\|z\| \ge \mu$ where $\mu = \alpha_1^{-1}\left(K_2/(1-\theta)K_1\right)$ then

$$(1-\theta) K_1 V \ge (1-\theta) K_1 \alpha_1(\mu) = K_2$$

Thus $\dot{V} \le -W$ for all $\|z\| \ge \mu$. Therefore from Theorem 4.18 in [8], the error trajectory z is globally uniformly ultimately bounded. The ultimate bound, all-time exponential-decay upper bound, and the time at which the trajectory enters the ultimate bound can be computed, but the details are omitted here.

Application to a Two-Link Flexible-Joint Robot Manipulator

We applied the proposed controller in Section 2 to a two-link flexible-joint robot manipulator shown in Figure 2. The manipulator operates in the horizontal plane and functions as follows. Input torque T_1 is applied to the first motor, which drives the first sprocket through a chain. The sprocket is attached to the first link via the first torsional spring that provides joint flexibility. The second motor is situated on the first link. Input torque T_2 is applied to the second motor, which drives the second sprocket. The second sprocket is attached to the second link via the second torsional spring.

1. Robot Model

This section derives the robot model and its transformation to the strict-feedback form (1). The description of parameters in the model is as follows. For the first link, θ_1 is the absolute angular position, m_1 is the lumped mass, J_1 is the moment of inertia, a_1 is the distance between center of gravity of the first link and the first joint, b_1 is the distance between the first joint and the second motor, and l_1 is the distance between the first joint and the second joint. For the second link, θ_2 is the relative angular position to θ_1, m_2 is the lumped mass, J_2 is the moment of inertia, a_2 is the distance between center of gravity of the second link and the second joint, and l_2 is the distance between the second joint and the end effector. For the first motor, θ_3 is the absolute angular position, m_3 is the lumped mass, J_3 is the moment of inertia, and c_3 is the friction coefficient. For the second motor, θ_4 is the relative angular position, m_4 is the lumped mass, J_4 is the moment of inertia, and c_4 is the friction coefficient. For the first sprocket, r is the gear ratio, $\theta_5 = \theta_3 / r$ is the absolute angular position, m_5 is the lumped mass, J_5 is the moment of inertia, and c_1 is the friction coefficient. For the first spring, k_5 is the spring coefficient, and c_5 is the spring damping. For the second sprocket, $\theta_6 = \theta_4 / r$ is the relative angular position, m_6 is the lumped mass, J_6 is the moment of inertia, and c_2 is the friction coefficient. For the second spring, k_6 is the spring coefficient, and c_6 is the spring damping. For payload at the end effector, m_p is the lumped mass, and J_p is the moment of inertia.

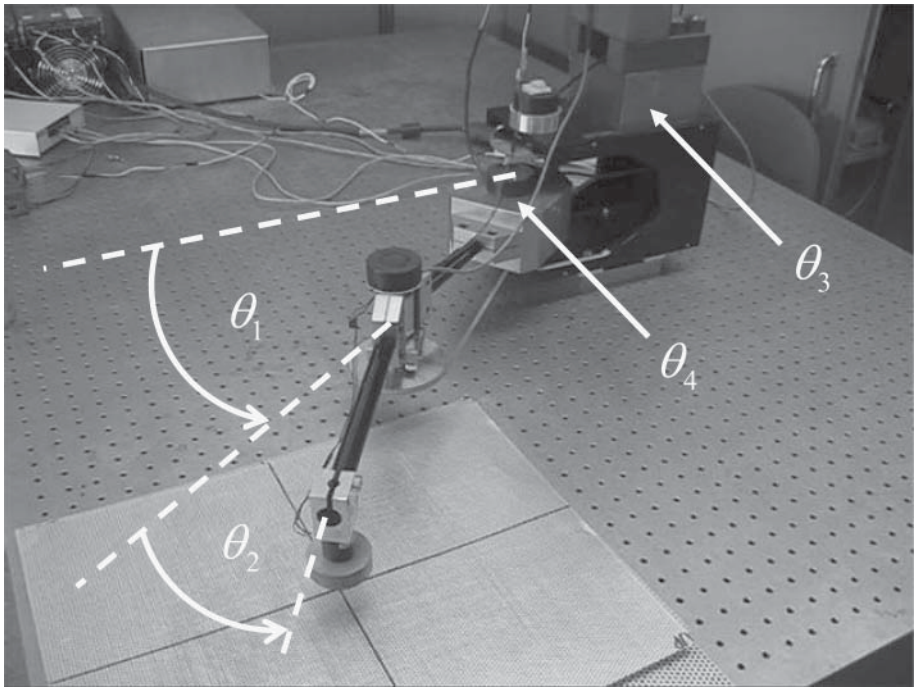

Figure 2: Photograph of a two-link flexible-joint robot manipulator under consideration.

Using the Euler-Lagrange method, the four equations of motions are

$$0=\Big[m_1a_1^2+m_2\left(l_1^2+a_2^2\right)+m_4b_1^2+m_6l_1^2+J_1+J_2+J_4+J_6+m_p\left(l_1^2+l_2^2\right)+J_p$$
$$+2l_1\left(m_2a_2+m_pl_2\right)\cos\left(\theta_2\right)\Big]\ddot{\theta}_1+\Big[m_pl_2^2+J_p+l_1\left(m_2a_2+m_pl_2\right)\cos\left(\theta_2\right)$$
$$m_2a_2^2+J_2\Big]\ddot{\theta}_2+\left(J_4+\frac{J_6}{r}\right)\ddot{\theta}_4-l_1\left(m_2a_2+m_pl_2\right)\dot{\theta}_2\left(2\dot{\theta}_1+\dot{\theta}_2\right)\sin\left(\theta_2\right)$$
$$+\left(c_1+c_5\right)\dot{\theta}_1-\frac{c_5}{r}\dot{\theta}_3+k_5\theta_1-\frac{k_5}{r}\theta_3,$$
$$0=\Big[m_2a_2^2+J_2+m_pl_2^2+J_p+l_1\left(m_2a_2+m_pl_2\right)\cos\left(\theta_2\right)\Big]\ddot{\theta}_1+\left(c_2+c_6\right)\dot{\theta}_2-\frac{c_6}{r}\dot{\theta}_4$$
$$+\left(m_2a_2^2+J_2+m_pl_2^2+J_p\right)\ddot{\theta}_2+k_6\theta_2-\frac{k_6}{r}\theta_4+l_1\left(m_2a_2+m_pl_2\right)\dot{\theta}_1^2\sin\left(\theta_2\right),$$
$$T_1=\left(J_3+\frac{J_5}{r^2}\right)\ddot{\theta}_3-\frac{c_5}{r}\dot{\theta}_1+\left(c_3+\frac{c_5}{r^2}\right)\dot{\theta}_3-\frac{k_5}{r}\theta_1+\frac{k_5}{r^2}\theta_3,$$
$$T_2=\left(J_4+\frac{J_6}{r}\right)\ddot{\theta}_1+\left(J_4+\frac{J_6}{r^2}\right)\ddot{\theta}_4-\frac{c_6}{r}\dot{\theta}_2+\left(c_4+\frac{c_6}{r^2}\right)\dot{\theta}_4-\frac{k_6}{r}\theta_2+\frac{k_6}{r^2}\theta_4.$$

Since the second motor and the second sprocket are located on the first link, their rotational kinetic energy comprises that of their own rotation and that of the first link rotation. To transform the dynamic equations above into strict-feedback form (1), we need the following assumptions.

Assumption 1: The rotational kinetic energy of the second motor and the second sprocket is mainly due to their own rotations, that is, we neglect the kinetic energy from the rotation of the first link. In the kinetic energy, therefore, the term $0.5J_4\left(\dot{\theta}_1+\dot{\theta}_4\right)^2$ becomes $0.5J_4\dot{\theta}_4^2$, and the term $0.5J_6\left(\dot{\theta}_1+\dot{\theta}_4/r\right)^2$ becomes $0.5J_6\left(\dot{\theta}_4/r\right)^2$.

Assumption 2: The internal damping of the torsional springs is small and the motor angular velocities are large compared to link angular velocities. Therefore the terms $c_5\left(\dot{\theta}_3/r-\dot{\theta}_1\right)^2/2$ and $c_6\left(\dot{\theta}_4/r-\dot{\theta}_2\right)^2/2$ in the dissipative power become $c_5\left(\dot{\theta}_3/r\right)^2/2$ and $c_6\left(\dot{\theta}_4/r\right)^2/2$, respectively.

Let $q_1=\left[\theta_1,\theta_2\right]^T$, $q_2=\left[\theta_5,\theta_6\right]^T=\left[\theta_3/r,\theta_4/r\right]^T$, and $T=\left[T_1,T_2\right]^T$ be the vectors of link angular positions, sprocket angular positions and input torque, respectively. The Lagrange equations can be put in the following form

$$M\left(q_1\right)\ddot{q}_1+V\left(q_1,\dot{q}_1\right)\dot{q}_1+F_1\left(\dot{q}_1\right)+K_1\left(q_1-q_2\right)=0,$$
$$J\ddot{q}_2+F_2\left(\dot{q}_2\right)+B\dot{q}_2-K_2\left(q_1-q_2\right)=T,$$

where $M(q_1)$ is the inertia matrix, $V(q_1,\dot{q}_1)\dot{q}_1$ represents coriolis and centrifugal terms, K_1 and K_2 are joint flexibility matrices, J represents the inertia of motors and sprockets, B contains internal damping of the torsional springs, and $F_1\left(\dot{q}_1\right)$ and $F_2\left(\dot{q}_2\right)$ are viscous friction vectors. Letting $x_1=q_1, x_2=\dot{q}_1, x_3=q_2, x_4=\dot{q}_2$, be state vectors, we obtain a model in the strict-feedback form

$$\begin{aligned}
\dot{x}_1 &= x_2,\\
\dot{x}_2 &= f_2\left(\bar{x}_2\right)+g_2\left(\bar{x}_2\right)x_3,\\
\dot{x}_3 &= x_4,\\
\dot{x}_4 &= f_4\left(\bar{x}_4\right)+g_4\left(\bar{x}_4\right)T,\\
y &= x_1,
\end{aligned}$$

where

$$f_2 = -M\left(x_1\right)^{-1}\left[V\left(x_1, x_2\right)x_2 + F_1\left(x_2\right) + K_1\left(x_1\right)\right],\ g_2 = M\left(x_1\right)^{-1}K_1,$$
$$f_4 = -J^{-1}\left[F_2\left(x_4\right) + Bx_4 - K_2\left(x_1 - x_3\right)\right],\ g_4 = J^{-1}.$$

System identification was performed using input and output signals from the actual robot. The robot model can be rearranged as a linear regression equation whose right-hand side contains known terms and whose left-hand side contains the product of known terms and unknown plant parameters. The unknown plant parameters were estimated by the linear least-squares method. The reader is referred to [9] for details of the system identification. The result is given by

$$M(q_1) = \begin{bmatrix} 0.201 + 0.06\cos\theta_2 & 0.0266 + 0.03\cos\theta_2 \\ 0.0266 + 0.03\cos\theta_2 & 0.0266 \end{bmatrix},$$
$$J = \begin{bmatrix} 0.017 & 0 \\ 0 & 0.014 \end{bmatrix},\ B = \begin{bmatrix} 5.66\times10^{-4} & 0 \\ 0 & 5.66\times10^{-4} \end{bmatrix},$$
$$V(q_1, \dot{q}_1) = \begin{bmatrix} 0 & -0.03(2\dot{\theta}_1 + \dot{\theta}_2)\sin\theta_2 \\ 0.03\dot{\theta}_1\sin\theta_2 & 0 \end{bmatrix},$$
$$K_1 = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.4 \end{bmatrix},\ K_2 = \begin{bmatrix} 0.075 & 0 \\ 0 & 0.075 \end{bmatrix}.$$

To simulate the additive uncertainties, we add $h_2\Delta_2 = \left[30\sin\left(x_{11}^2\right), 30\sin\left(x_{12}^2\right)\right]^T$ and $h_4\Delta_4 = \left[30\cos(3x_{11}x_{31}), 30\cos(3x_{12}x_{32})\right]^T$ to the system. Letting $u = T$, we obtain the state-space model to represent the actual plant during the simulation as

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = f_2\left(x_1, x_2\right) + g_2\left(x_1, x_2\right)x_3 + h_2\Delta_2,$$
$$\dot{x}_3 = x_4,$$
$$\dot{x}_4 = f_4\left(x_1, x_2, x_3, x_4\right) + g_4\left(x_1, x_2, x_3, x_4\right)u + h_4\Delta_4.$$

2. *Training of ANFIS*

It is assumed that all g_i are known and $f_2 \in \mathbb{R}^2$ and $f_4 \in \mathbb{R}^2$ are approximated offline using ANFIS. We supply dummy control input u to the system and collect 200 data patterns of input u, states x_1, x_2, x_3, x_4, and their derivatives $\dot{x}_1, \dot{x}_2, \dot{x}_3, \dot{x}_4$ to be used in training. Figure 3 and Figure 4 depict the training data.

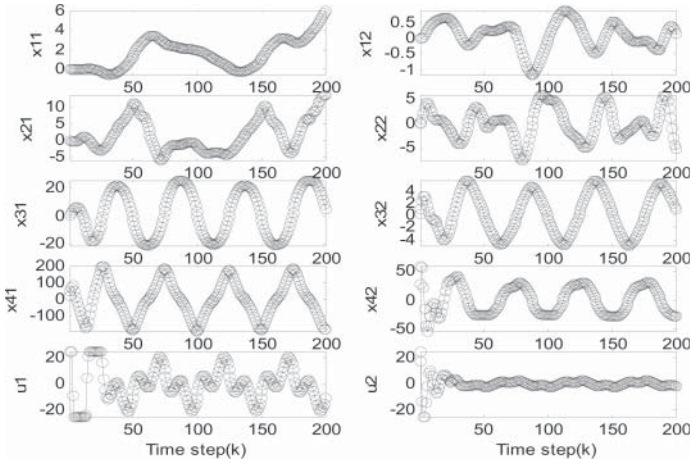

Figure 3: Training data for f_2, f_4 ANFIS approximations: x_1, x_2, x_3, x_4, u.

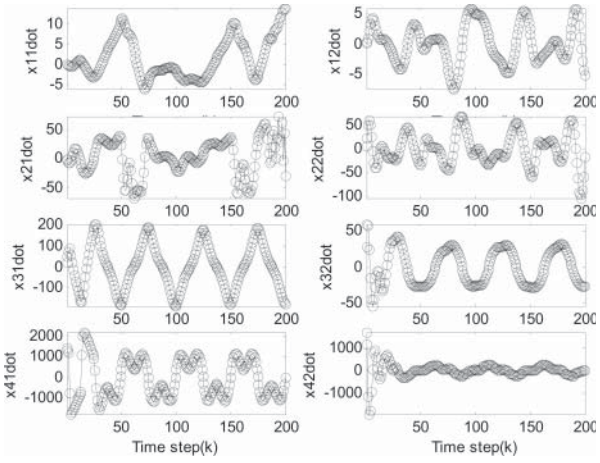

Figure 4: Training data for f_2, f_4 ANFIS approximations: $\dot{x}_1, \dot{x}_2, \dot{x}_3, \dot{x}_4$.

To approximate $f_2 = [f_{21}, f_{22}]^T \in \mathbb{R}^2$, we consider the second subsystem

$$\dot{x}_2 = f_2(x_1, x_2) + g_2(x_1, x_2)x_3 + h_2\Delta_2 = F_2(x_1, x_2) + g_2(x_1, x_2)x_3.$$

Since $h_2\Delta_2$ is unknown, we have to approximate $F_2 = [F_{21}, F_{22}]^T$ instead. One ANFIS is used for each element of F_2. For F_{21}, the training inputs are $x_{11}, x_{12}, x_{21}, x_{22}$ and the training output is $\dot{x}_{21} - (g_{2,11}x_{31} + g_{2,12}x_{32})$. For F_{22}, the training inputs are $x_{11}, x_{12}, x_{21}, x_{22}$ and the training output is $\dot{x}_{22} - (g_{2,21}x_{31} + g_{2,22}x_{32})$. We use two membership functions for each input, 200 training data patterns per epoch, and let the number of epochs be 100. The Matlab ANFIS toolbox is used to obtain the results. The approximation of F_{21} is given in Figure 5. Figure 5(a) shows root mean square training errors, which are the differences between the ANFIS output and the training data output at each epoch. Figure 5(b) shows the optimization step size. The step size is reduced as the error gets closer to the optimum value. Figure 5(c) and (d) show validation results after the training is finished. The approximation of F_{22} is given in Figure 6.

To approximate $F_4 = [F_{41}, F_{42}]^T \in \mathbb{R}^2$, we consider the fourth subsystem

$$\begin{aligned}\dot{x}_4 &= f_4(x_1, x_2, x_3, x_4) + g_4(x_1, x_2, x_3, x_4)u + h_4\Delta_4 \\ &= F_4(x_1, x_2, x_3, x_4) + g_4(x_1, x_2, x_3, x_4)u.\end{aligned}$$

The F_4 approximation is done similarly. The results are given in Figure 7 and Figure 8.

3. Tracking Results

The robust backstepping control in this section is implemented with the following design parameters

$$c_i = k_i = 17, \ \forall i = 1, \ldots, 4.$$

Figure 9 shows good tracking performance of both shoulder angular position θ_1 and elbow angular position θ_2, even when there are rather large additive uncertainties in the system.

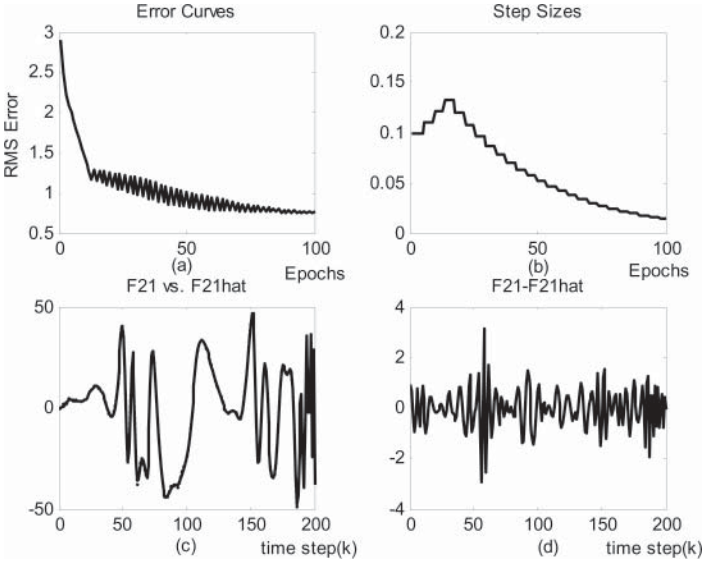

Figure 5: F_{21} approximation using ANFIS. (a) Training error. (b) Optimization step size. (c) F_{21} (dotted line) and $\hat{F}_{21}$ (solid line). (d) $F_{21} - \hat{F}_{21}$.

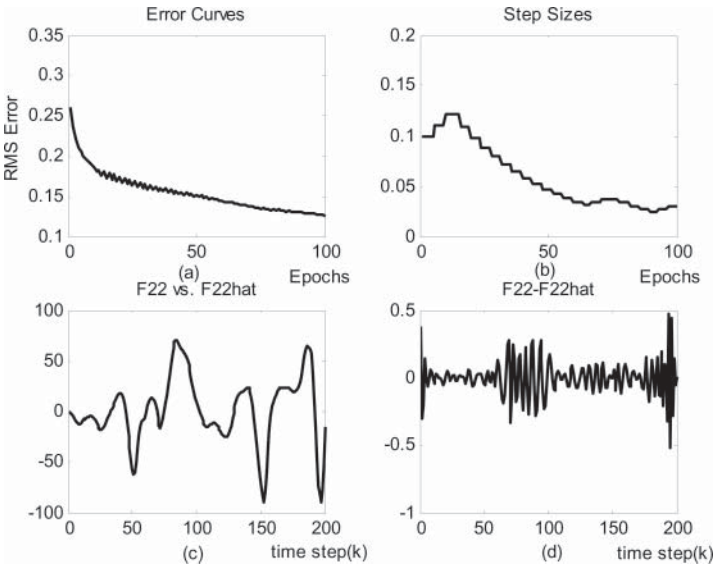

Figure 6: F_{22} approximation using ANFIS. (a) Training error. (b) Optimization step size. (c) F_{22} (dotted line) and $\hat{F}_{22}$ (solid line). (d) $F_{22} - \hat{F}_{22}$.

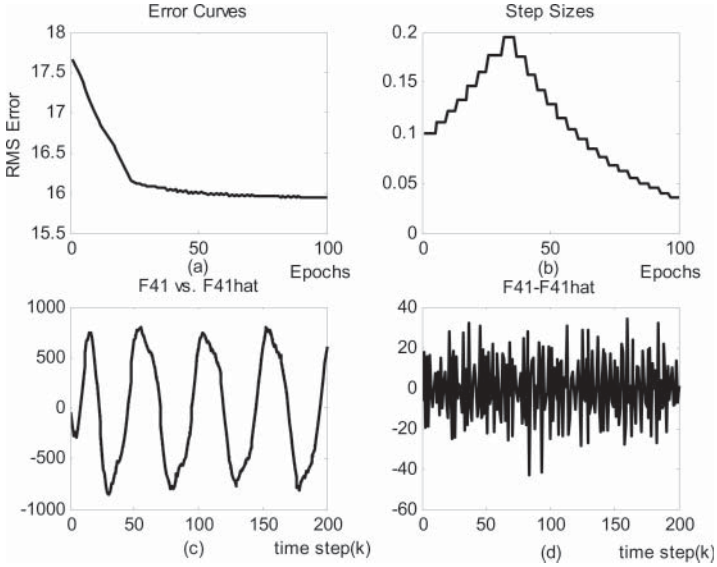

Figure 7: F_{41} approximation using ANFIS. (a) Training error. (b) Optimization step size. (c) F_{41} (dotted line) and $\hat{F}_{41}$ (solid line). (d) $F_{41} - \hat{F}_{41}$.

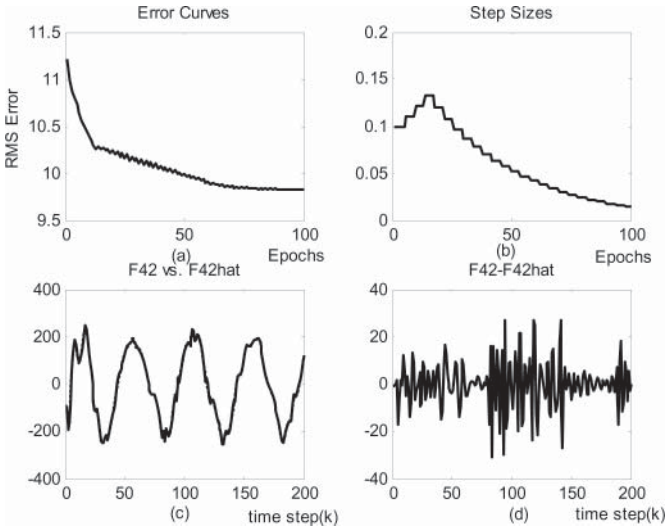

Figure 8: F_{42} approximation using ANFIS. (a) Training error. (b) Optimization step size. (c) F_{42} (dotted line) and $\hat{F}_{42}$ (solid line). (d) $F_{42} - \hat{F}_{42}$.

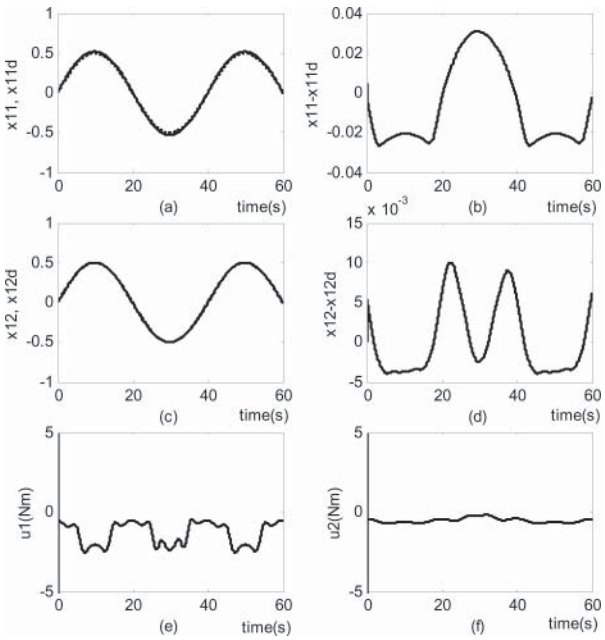

Figure 9: Tracking performance and actual control inputs in 60 seconds. (a) θ_1 and θ_{1d}. The solid line is θ_1 and the dotted line is θ_{1d}. (b) $\theta_1 - \theta_{1d}$. (c) θ_2 and θ_{2d}. The solid line is θ_2 and the dotted line is θ_{2d}. (d) $\theta_2 - \theta_{2d}$. (e) Input Torque $u_1 = T_1$. (f) Input Torgue $u_2 = T_2$.

Conclusions

A neuro-fuzzy system is presented to estimate the unknown part of the plant model, and a controller is designed from the estimated plant model. Nonlinear damping is incorporated to handle uncertainties in each subsystem using a backstepping structure. The proposed controller exhibits good tracking performance in our experiments.

The method presented here can be extended to incorporate online estimation of the plant model rather than the offline estimation used in the paper. The applicable plant in this paper is square, that is, the dimension of the input vector is the same as the dimension of the state vector. It should require only minor extension from a square system to a non-square system so that the proposed controller can be applied to a broader class of systems.

References

[1] W. Chatlatanagulchai and P. H. Meckl, "Motion control of two-link flexible-joint robot, using backstepping, neural networks, and indirect method," *Proc. of 2005 IEEE Conf. on Control Applications*, Toronto, pp. 601-605.

[2] W. Chatlatanagulchai and P. H. Meckl, "Model-free trajectory control of a two-link flexible-joint robot: theory and experiment," *Proc. of 2005 ASME Int. Mechanical Engineering Congress and Exposition*, Florida.

[3] W. Chatlatanagulchai and P. H. Meckl, "Robust observer backstepping neural network control of nonlinear systems in strict feedback form," *Proc. of 2004 American Control Conf.*, Boston, pp. 3035-3040.

[4] M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design*, Wiley Interscience, New York, 1995.

[5] L. M. Sweet and M. C. Good, "Re-definition of the Robot Motion Control Problem: Effects of Plant Dynamics Drive System Constraints, and User Requirements," *Proceedings of the 23rd IEEE Conference on Decision and Control*, Las Vegas, NV, 1984, pp. 724-731.

[6] J. S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 23, No. 3, 1993, pp. 665-685.

[7] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice Hall, New Jersey, 1997.

[8] H. K. Khalil, *Nonlinear Systems*, 3rd edition, Prentice Hall, New Jersey, 2002.

[9] W. Chatlatanagulchai, *Backstepping Intelligent Control Applied to a Flexible-Joint Robot Manipulator*, Ph.D. Dissertation, Department of Mechanical Engineering, Purdue University, West Lafayette, Indiana, 2006.