Adaptive Learning Control for Nonminimum Phase Systems

X. Z. Wang and D. J. Chen
Iowa State University
Ames, IA, 50011

Abstract

In this paper, a new adaptive learning algorithm is presented for the repetitive tracking control of a class of unstable nonminimum phase systems. After each repetitive trial, Least-Squares method is used to estimate the system parameters. The output tracking error and the identified system model are used through stable inversion to find the feed forward input, together with the desired state trajectories, for the next trial. An adaptive backstepping based tracking controller is used in each trial to ensure the regulation of the desired state trajectories. Simulation results demonstrate that the proposed learning control scheme is very effective in reproducing the desired trajectories.

1 Introduction

Iterative learning control (ILC) is a feed forward control approach aimed at achieving high performance output tracking control by "learning" from past experience so as to eliminate the repetitive errors from future execution [9]. This approach was motivated by the observation that human beings are able to improve performance through repeated practice. Since learning controller is able to eliminate the repetitive errors that exist when using a servo controller alone, it has great potential in future robotic systems.

The concept of iterative learning control was first introduced by Arimoto et al. [1]. It is based on the use of repeated trials to track a desired trajectory. At each trial, the system input and output signals are stored. The learning control algorithm then evaluates the performance error. Based on the error signal, the learning controller computes a new input signal, which is stored for use during the next trial. The new input is chosen such that the performance error will be reduced in the next trial. One of the important features of iterative learning control is that it requires little *a priori* knowledge about the controlled system during the controller design phase.

Arimoto's original learning controller is called a D-type algorithm. Since then, many researchers have proposed various learning control schemes. Moore extended Arimoto's method to systems with relative degree higher than one [9]. Hauser presented a nonlinear version of Arimoto's method for a class of nonlinear systems [5]. Application of this type of learning controllers to robotics was reported in many studies such as [2] and [7]. The basic and succinct exposition of ILC is surveyed in [10].

Although existing learning algorithms have been theoretically proven to provide output error convergence and have had successful applications, many such algorithms have practical difficulties with nonminimum phase systems [3]. An adaptive learning algorithm that works for nonminimum phase systems was recently developed by Gao and Chen [3]. In this paper, an adaptive learning algorithm is further developed to work for unstable nonminimum phase systems. Simulation results are presented to show the effectiveness of the proposed adaptive learning algorithm.

The remainder of the paper is organized as follows. In the next section we define a class of desired trajectories under consideration and state the problem of ILC. Section 3 presents the new adaptive learning control law. Section 4 contains simulation results. Finally, some conclusion remarks are given in Section 5.

2 Problem Statement

Consider a system dynamics in the k^{th} trial:

$$\dot{x}_k = f(x_k, \theta, u_k) \quad (1)$$

$$y_k = h(x_k, \theta, u_k) \quad (2)$$

where x_k is defined on a neighborhood X of the origin of $\Re^n$, θ is a parameter vector, with input $u_k \in \Re^m$ and output $y_k \in \Re^p$. The mappings f and g are smooth in x_k and u_k, with $f(0, \theta, 0) \equiv 0$ and $h(0, \theta, 0) \equiv 0$.

We make the following assumptions:

(A1) The system has a well-defined relative degree $r = (r_1, \cdots, r_m)^T$ which is known. The linearization of the system about an equilibrium point, which is assumed to be the origin WLOG, is completely controllable.

(A2) The order of the system, n, is known.

(A3) The system parameter vector θ is unknown or known incompletely.

0-7803-6583-6/00/$10.00 © 2000 IEEE

(A4) A desired output trajectory is given and is a sufficiently smooth function of t satisfying $y_d(t) = 0$ for any $t \in (-\infty, 0] \cup [T, \infty)$ and finite for any $t \in (0, T)$, where $T > 0$.

Note: In (A4), sufficiently smooth means that the signal has continuous derivatives of any order up to the relative degree. (A4) also requires $y_d(t)$ having a compact support $[0, T]$.

Iterative Learning Control Problems :

Given a desired output trajectory $y_d(t)$ and a tolerance error bound ϵ for a class of system (1) and (2), starting from an arbitrary continuous initial control input $u_0^d(\cdot)$ and initial state $x_0^d(\cdot)$, iterative learning control is to find a sequence of desired state trajectory $x_k^d(\cdot)$ and desired control inputs $u_k^d(\cdot)$, which when applied to the system, produces an output sequence $y_k(\cdot)$ such that

(1) $\|y_d(\cdot) - y_k(\cdot)\|_\infty \leq \epsilon$, as $k \to \infty$, where k is the trial number and $\|f\|_\infty = \sup_{t\in[0,T]}\|f(t)\|$.

(2) $\|u_k^d(t)\| \leq \epsilon, \|x_k^d(t)\| \leq \epsilon, \forall t \in (-\infty, 0] \cup [T, +\infty)$.

(3) $u_k^d(t)$, $x_k^d(t)$, $u_k(t)$, and $x_k(t)$ are uniformly bounded.

The system can be represented in terms of desired control input $u_k^d(\cdot)$ and output $y_k(\cdot)$ in the k^{th} trial by means of a nonlinear time-varying operator Π as follows:

$$y_k(\cdot) = \Pi(\cdot)u_k^d(\cdot) \tag{3}$$

In this dynamic process, the functions have two arguments: continuous time t and the trial number k. In the sequel it is assumed that the variation of the operator over two consecutive trials are slow and can be neglected. Then the operator obtained by identification performed in the k^{th} trial can be used to determine the input for the $(k+1)^{th}$ trial. This general description of the problem allows a simultaneous description of linear or nonlinear dynamics, continuous or discrete plant, and time-invariant or time-varying systems.

For applying linear ILC, however, the plant must fulfill the following conditions: (1) The desired trajectory $y_d(t)$ is identical for every trial and satisfies Assumption (A4). (2) Each trial has the fixed period T. (3) The system parameters are fixed or very slowly time-varying.

3 Adaptive Learning Control

In section 2, we have given the general setup of learning control. In this section, an adaptive learning controller will be presented. The block diagram of the adaptive learning system is shown in Figure 1.

The proposed adaptive learning control strategy has three components: a parameter estimator, a stable inverse system, and an adaptive backstepping feedback controller. The parameter estimator is in charge of "learning" the parameterized model of the system. During each trial, the input and output trajectories are recorded. Then off-line Least-Squares method is applied to obtain the optimal estimate of parameters. Also obtained during each trial is the output tracking error signal. This error signal and the estimated model are used by the stable inverse system to learn the optimal input signal for the next trial. Although the estimated model may be nonminimum phase which normally leads to unbounded inverse solutions, stable inversion guarantees a unique and bounded inverse solution. This "learning" action is done "off-line" between two consecutive trials. Afterwards, the new feed forward input is used by the adaptive backstepping feedback controller to stabilize the system and to ensure regulation of the tracking error. The controller is designed following a recursive backstepping procedure and it takes advantage of the parametric strict-feedback structure of the system. The controller parameters are also continuously updated in real-time using an adaptive control law. The same feedback control algorithm is used during every trial. In the following, we give the implementation of the adaptive learning algorithm for continuous-time systems.

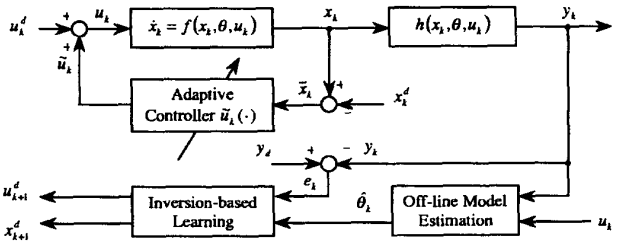

Figure 1: Block Diagram of Adaptive Learning Control System

3.1 Solution to Stable Inversion of nonminimum Phase Systems

Consider a LTI system in the form:

$$\sum_1 : \begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

Suppose an estimated model $\hat{G}$ is obtained. If it is minimum phase, one can obtain the desired feed forward input by

$$u^d = \hat{G}^{-1} y_d \tag{4}$$

However, if $\hat{G}$ is nonminimum phase, this will lead to unbounded solutions. The stable inversion theory [4] provides an avenue to overcome this difficulty. It was shown that under certain conditions, there exists a unique stable inverse system $\hat{H}$ of $\hat{G}$ such that the inverse solution $\hat{H}y_d$ is bounded and it reproduces y_d exactly when applied as an input to $\hat{G}$, that is,

$$\hat{G}(\hat{H}y_d) = y_d \tag{5}$$

Here, the procedure to obtain this unique stable inverse solution $u^d = \hat{H} y_d$ is illustrated. There are four steps:

(1) Find the time-domain state space model of $\hat{G}$:

Since $\hat{G}u^d = \hat{G}\hat{H}y_d = y_d$, a state-space representation of $\hat{G}$ yields

$$\dot{x}^d(t) = \hat{A}x^d(t) + \hat{B}u^d(t) \quad (6)$$
$$y_d(t) = \hat{C}x^d(t) + \hat{D}u^d(t) \quad (7)$$

where x^d is the state and $\hat{A}, \hat{B}, \hat{C}$, and $\hat{D}$ are matrices with suitable sizes.

(2) Find its inverse in state space:

Differentiate $y_d(t)$ until u^d appears explicitly in the right hand side. Solve for u^d and substitute into (6) and (7) to obtain

$$\dot{x}^d(t) = \bar{A}x^d(t) + \bar{B}y_d^{(r)}(t) \quad (8)$$
$$u^d(t) = \bar{C}x^d(t) + \bar{D}y_d^{(r)}(t) \quad (9)$$

where $\bar{A}, \bar{B}, \bar{C}$, and $\bar{D}$ are defined according to the substitution.

(3) Decompose the inverse system into center, stable, and unstable subsystems:

Perform a change of variables so that

$$x^d = Pz = P[z^c, z^s, z^u]^T \quad (10)$$

which leads to

$$\dot{z}^c = A^c z^c + B^c y_d^{(r)} \quad (11)$$
$$\dot{z}^s = A^s z^s + B^s y_d^{(r)} \quad (12)$$
$$\dot{z}^u = A^u z^u + B^u y_d^{(r)} \quad (13)$$
$$u^d = [C^c \ C^s \ C^u][z^c \ z^s \ z^u]^T + \bar{D}y_d^{(r)} \quad (14)$$

where A^c, A^s, and A^u are real Jordan matrices of suitable dimensions; A^c has r eigenvalues at zero; A^s has all eigenvalues in the open left-half plane; A^u has all eigenvalues in the open right-half plane.

(4) Obtain the stable inverse system:

Pick the transformation matrix P so that the center subsystem is a simple chain of r integrators. Solve that and impose two boundary conditions on the stable and unstable subsystems to yield

$$z^c = [y_d, \dot{y}_d, \cdots, y_d^{(r-1)}]^T \quad (15)$$
$$\dot{z}^s = A^s z^s + B^s y_d^{(r)}, t \geq 0; z^s(t) = 0, \ \forall\, t \leq 0 \quad (16)$$
$$\dot{z}^u = A^u z^u + B^u y_d^{(r)}, t \leq T; z^u(t) = 0, \ \forall\, t \geq T \quad (17)$$

These together with (10) and (14) define the desired stable inverse system.

In classical inversion, (8) and (9) are treated as a dynamic system. Since it is unstable for a nonminimum phase system, it leads to unbounded solutions for u^d. In contrast, the stable inverse system always yields bounded solutions for bounded and smooth y_d. This can be clearly seen from (15)-(17) since the center solution z^c is clearly bounded, the stable subsystem is in the forward time, and the unstable subsystem is in the reverse time, all leading to bounded solutions.

When the system is minimum phase, there will be no unstable subsystem. And the dimension of z^u and those of the associated matrices will be zero. Therefore, the proposed approach also applies to minimum phase systems.

In this paper, we only consider LTI systems in this form:

$$\sum_2 : \begin{cases} \dot{x}_{i-1,k} = x_{i,k}; \ i = 2, 3, \cdots, n \\ \dot{x}_{n,k} = x_k^T a + u_k \\ y_k = C x_k \end{cases}$$

Based on the stable solutions outline presented above, to facilitate iterative learning, we modify the inversion process slightly as follows. Referring to Figure 1, let G denote the system operator from u_k to y_k. Let u_{k+1} be the new input for the next trial. Then the new tracking error signal will be

$$\begin{aligned} e_{k+1} &= y_d - Gu_{k+1} \\ &= y_d - G(u_{k+1}^d + \tilde{u}_{k+1}) \\ &= e_k - G(u_{k+1}^d - u_k^d) - G(\tilde{u}_{k+1} - \tilde{u}_k) \end{aligned}$$

Then we get

$$e_{k+1} + G(\tilde{u}_{k+1} - \tilde{u}_k) = e_k - Gu_{k+1}^e$$

where $u_{k+1}^e = u_{k+1}^d - u_k^d$.

The goal of designing learning control law is to make $e_{k+1}(\cdot)$ gradually decrease as k increases. For any remaining errors, the feedback control action will try to reduce them. Therefore, we simply set $e_{k+1} = 0$ and $\tilde{u}_{k+1} - \tilde{u}_k = 0$ to design u_{k+1}^e. That is, we want

$$e_k = Gu_{k+1}^e \quad (18)$$

Since G is unknown and $\hat{G}_k$ is the best estimate model after the k^{th} trial, we will use

$$u_{k+1}^e = \hat{H}_k e_k \quad (19)$$
$$x_{k+1}^e = P_k z = P_k[z^c, z^s, z^u]^T \quad (20)$$

so that $e_k = \hat{G}_k u_{k+1}^e$. Then our learning algorithm would be

$$u_{k+1}^d = u_k^d + u_{k+1}^e \quad (21)$$
$$x_{k+1}^d = x_k^d + x_{k+1}^e \quad (22)$$

As a further modification, one may introduce a forgetting factor α $(0 < \alpha < 1)$ and use:

$$u_{k+1}^d = u_k^d + \alpha u_{k+1}^e \quad (23)$$
$$x_{k+1}^d = x_k^d + \alpha x_{k+1}^e \quad (24)$$

where u^e_{k+1} and x^e_{k+1} are given by (19) and (20), which are the stable inverse solutions from e_k and $\hat{G}_k$. (Note that: in the rest of the paper, u^d, x^d, and u^e represent u^d_{k+1}, x^d_{k+1}, and u^e_{k+1} respectively for notational convenience).

However, the controller design in the next subsection will still assume $\alpha = 1$ so that u^d, x^d, and y_d satisfy the dynamics of $\hat{G}_k$, that is :

$$\sum_3 : \begin{cases} \dot{x}^d_{i-1} = x^d_i; \ i = 2, \cdots, n \\ \dot{x}^d_n = \hat{a}_k^T x^d + u^d \\ y_d = \hat{C}_k x^d \end{cases}$$

3.2 Adaptive Backstepping Controller Design

There are a lot of methods to design a controller. Since the parameters of the systems are unknown, we need to design an adaptive controller. Here we follow a popular approach of adaptive backstepping design [8].

Define $\tilde{x}_k = x_k - x^d$ and $\tilde{u}_k = u_k - u^d$. For clarity, we will drop the subscript k in sections 3.2 and 3.3 if it does not cause confusion. One can easily verify:

$$\sum_4 : \begin{cases} \dot{\tilde{x}}_{i-1} = \tilde{x}_i; \ i = 2, \cdots, n \\ \dot{\tilde{x}}_n = \tilde{x}^T a + [a - \hat{a}]^T x^d + \tilde{u} \end{cases}$$

The goal is to design $\tilde{u}$ to guarantee the regulation of $\tilde{x}$. Since a is unknown, let $\vartheta(t)$ be the on-line estimate of a and rewrite the last equation as follows:

$$\begin{aligned} \dot{\tilde{x}}_n &= \tilde{x}^T \vartheta + [\vartheta - \hat{a}]^T x^d + \tilde{u} + [a - \vartheta]^T (\tilde{x} + x^d) \\ &= \psi^T(\tilde{x})\tilde{a} + \tilde{u}_2 \end{aligned}$$

where $\tilde{a} = a - \vartheta$, $\psi^T(\tilde{x}) = x^T$. Then $\sum_4$ is in a standard form with matching condition, then the goal becomes to design $\tilde{u}_2$ to guarantee the regulation of $\tilde{x}$. Details of the derivations are skipped here, but the final controller is given by:

$$u = u^d + \tilde{u}_2 - \tilde{x}^T \vartheta - [\vartheta - \hat{a}]^T x^d \quad (25)$$

$$\tilde{u}_2 = \alpha_n(\tilde{x}, \vartheta) \quad (26)$$

$$\dot{\vartheta} = \Gamma \psi z_n \quad (27)$$

where Γ is an adaptation gain matrix. The variables z_i and the stabilizing functions $\alpha_i, i = 1, \ldots, n$, are defined by the following recursive expressions:

$$z_i = \tilde{x}_i - \alpha_{i-1}(\tilde{x}_1, \ldots, \tilde{x}_{i-1}) \quad (28)$$

$$\alpha_i = c_i z_i - z_{i-1} + \sum_{j=1}^{i-1} \frac{\partial \alpha_{i-1}}{\partial \tilde{x}_j} \tilde{x}_{j+1}, \ i = 1, \cdots, n-1 \quad (29)$$

$$\alpha_n = c_n z_n - z_{n-1} - \psi^T \vartheta + \sum_{j=1}^{n-1} \frac{\partial \alpha_{n-1}}{\partial \tilde{x}_j} \tilde{x}_{j+1} \quad (30)$$

This adaptive controller guarantees global boundedness of $\tilde{x}(t), \vartheta(t)$, and regulation of $\tilde{x}_i(t), i = 1, \ldots, n$, i.e., $\tilde{x}_i(t) \to 0$, as $t \to \infty$.

3.3 Parameter Estimator

Off-line Least-Squares method is used to estimate the parameters. To get $\hat{a}_{k+1}$, we use the method as follows:

$$\hat{a}_{k+1} = d_k \hat{a}_k + (1 - d_k)\bar{a}_k \quad (31)$$

where $d_k \in [0, 1)$ is a memory factor and $\bar{a}_k$ is determined by using off-line Least-Squares method [6]using data from the k^{th} trial. First using filter for the last equation of $\sum_3$, we get

$$\frac{1}{s+\lambda}\dot{x}_n = \frac{1}{s+\lambda} x^T a + \frac{1}{s+\lambda} u \quad (32)$$

$$x_n - \frac{1}{s+\lambda} u = \frac{1}{s+\lambda} x^T \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} + \lambda \end{pmatrix} \quad (33)$$

where $\lambda > 0$. Then we have $Z = W^T \hat{a}_\lambda$, where $Z = (x_n - \frac{1}{s+\lambda} u)$, and $W^T = \frac{1}{s+\lambda} x^T$. By solving the ordinary differential equations (ODE), we get Z and W. Now collect all the data of Z and W. Suppose there are totally M samples for the k^{th} trial. Let $\Phi = [W_1^T, \cdots, W_M^T]^T$ and the regressor vector be $\Psi = [Z_1, \cdots, Z_M]^T$. The Lease-Squares solution is

$$\hat{a}_\lambda = (\Phi^T \Phi)^{-1} \Phi^T \Psi \quad (34)$$

From this we get $\bar{a}_k$. Then $\hat{a}_{k+1}$ is obtained by (31). Similarly, for the linear model,

$$y = Cx \quad (35)$$

We can use the same argument to get the estimates of C, except that no filtering is needed.

3.4 Adaptive Learning Algorithm

The process of the algorithm is as follows:

Step 0 : Given ε, the initial conditions $\hat{a}_0$, $\hat{C}_0$, the initial input $u^d_0(t) = 0$, and initial state trajectory $x^d_0(t) = 0$ on $t \in [0, T]$. Set k=0.

Step 1: Let $e_k(t) = y_d(t) - y_k(t)$. Get $\hat{G}_k$ from $\hat{a}_k$ and $\hat{C}_k$. Use stable inversion to get $u^e_{k+1} = \hat{H}_k e_k$ and x^e_{k+1}. Use equation (23) and (24) to get u^d_{k+1} and x^d_{k+1}.

Step 2: $u^d_{k+1}(t)$ is used as feed forward by the adaptive backstepping feedback controller to stabilize the system and to ensure regulation of $x^d_{k+1}(t)$, i.e., $x_k(t) \to x^d_{k+1}(t)$, as $t \to \infty$. The input and output trajectories are recorded respectively.

Step 3 : Then the tracking error signal $e_{k+1}(t)$ is calculated. If $\|e_{k+1}(\cdot)\|_\infty \leq \epsilon$, stops. Otherwise, set $k = k+1$, and go to Step 4 .

Step 4: Use off-line Least-Squares method to obtain the parameter estimates $\hat{a}_k$ and $\hat{C}_k$, and go back to Step 1.

Table 1: Parameter estimates and output tracking error in each trial for the 2^{nd} order nonminimum phase system

k	$\hat{a}_{0,k}(3)$	$\hat{a}_{1,k}(7)$	$\hat{C}_{0,k}(10)$	$\hat{C}_{1,k}(-1)$	$\|e_k\|_\infty$
1	2.8999	6.9235	9.9999	-1.0000	0.1021
2	2.9339	6.9636	10.0000	-0.9999	0.0003
3	2.9510	6.9837	10.0000	-1.0000	0.0002
4	2.9609	6.9980	10.0000	-1.0000	0.0001

4 Simulation Illustrations

Simulation results are provided for SISO linear nonminimum phase systems with unknown parameters. Two examples of second order and third order unstable nonminimum phase systems are included to verify the effectiveness of the proposed adaptive learning algorithm.

Example1 Consider a nonminimum phase plant:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = [3 \quad 7]x + u \\ y = [10 \quad -1]x \end{cases}$$

The parameters $a = [3 \quad 7]^T$, $C = [10 \quad -1]$. This system has two poles at 7.4051, -0.4051, and one zero at 10. Take the initial conditions $\hat{a}_0 = [2.8 \quad 6.8]^T$, $\hat{C}_0 = [9 \quad -0.8]$. Let the desired trajectories $y_d(t) = 4.6685 - 0.4244cos(1.5708t) - 4.2441cos(0.1571t)$ as shown by the solid curve in Figure 2. Take an initial input $u_0^d(t) = 0$ and initial state trajectory $x_0^d(t) = 0$. Simulation results for the trial $k = 1$ and $k = 2$ are shown in Figure 2. At the 2^{nd} trial, the output $y_2(t)$ converges to the desired $y_d(t)$ exactly by the dotted curve. Table 1 shows the parameter estimates and the infinity norm of the output tracking error at each trial. We can see the estimated parameters are very close to the true values at the 3^{rd} trial.

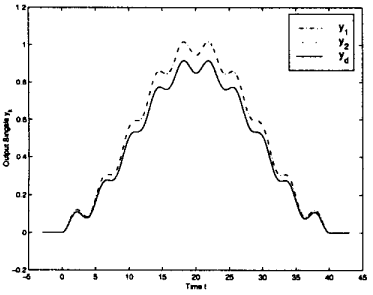

Figure 2: Tracking of 2^{nd} order linear nonminimum phase systems

Example2 Consider a nonminimum phase plant:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \dot{x}_3 = [1 \quad 2 \quad 3]x + u \\ y = [-21 \quad -4 \quad 1]x \end{cases}$$

The parameters $a = [1 \quad 2 \quad 3]^T, C = [-21 \quad -4 \quad 1]$. This system has poles at $-0.3137 \pm 0.4211i$, and 3.6274, and zeros at 7 and -3. Take initial conditions $\hat{a}_0 = [0.8 \quad 1.8 \quad 2.8]^T, \hat{C}_0 = [-25 \quad -3 \quad 2]$. Let the desired output $y_d(t) = -0.2759 - 0.0424cos(1.5708t) - 0.2122cos(0.3142t) - 0.1061cos(0.6283t)$ as shown by the solid curve in Figure 3. Take an initial input $u_0^d(t) = 0$ and initial state trajectory $x_0^d(t) = 0$. Simulation results for the trial $k = 1$ and $k = 2$ are shown in Figure 3. At the 2^{nd} trial, the output $y_2(t)$ converges to the desired $y_d(t)$ exactly as shown the dotted curve. Table 2 shows the parameter estimates and the infinity norm of the output tracking error at each trial. We can see the estimated parameters are very close to the true values at the 4^{th} trial.

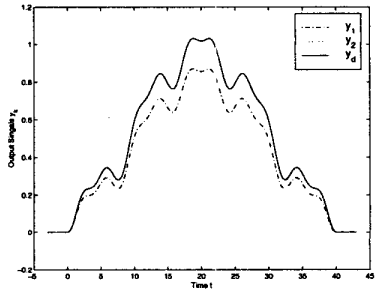

Figure 3: Tracking of 3^{rd} order linear nonminimum phase systems

The above results demonstrate that the proposed learning control is very effective in reproducing the desired trajectories.

5 Summary and Conclusions

A new adaptive learning algorithm has been developed for unstable nonminimum systems. The adaptive backstepping feedback control law is employed to guarantee regulation of tracking error and a stable inverse system is used to update the feed forward input for the next trial. Given a desired trajectory, the learning controller is able to learn and eventually drive the closed loop dynamics to track the desired trajectory. Simulation results demonstrate the effectiveness of the proposed method.

Table 2: Parameter estimates and output tracking error in each trial for 3^{rd} order nonminimum phase systems

k	$\hat{a}_{0,k}(1)$	$\hat{a}_{1,k}(2)$	$\hat{a}_{2,k}(3)$	$\hat{C}_{0,k}(-21)$	$\hat{C}_{1,k}(-4)$	$\hat{C}_{2,k}(1)$	$\|e_k(\cdot)\|_\infty$
k=1	0.8998	1.8982	2.9059	-21.0000	-4.0000	0.9999	0.1678
k=2	0.9332	1.9360	2.9391	-21.0000	-4.0000	1.0000	0.0007
k=3	0.9500	1.9541	2.9581	-21.0000	-3.9999	1.0000	0.0004
k=4	0.9600	1.9648	2.9671	-20.9999	-3.9999	1.0000	0.0002

References

[1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering Operation of Robots by Learning," *Journal of Robotic Systems*, Vol. 1, pp. 123-140, 1984.

[2] P. Bondi, G. Casalino, and L. Gambardellal, "On the Iterative Learning Control Theory for Robotic Manipulators," *IEEE Transactions on Robotics and Automation*, Vol. 4, pp. 14-22, 1989.

[3] J. Gao and D. J. Chen, "Iterative Learning Control for Nonminimum Phase Systems," *American Control Conference*, 1998.

[4] D. Chen and B. Paden, "Stable Inversion of Nonminimum Phase Nonlinear Systems," *International Journal of Control*, Vol. 64, pp. 81-96, 1996.

[5] J. Hauser, "Learning Control for a Class Nonlinear Systems," *Proceedings of the 26th Conference on Decision and Control*, pp. 859-860, 1987.

[6] R. Johansson, *System Modelling Identification*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[7] S. Kawamura, F. Miyazaki, and S. Arimoto, "Application of Learning Method for Dynamic Control of Robot Manipulations," *Proceedings of the 24th Conference on Decision and Control*, 1381-1386, 1985.

[8] M. Krstić, I. Kanellakopoulos, and P. V. Kokotovic, *Nonlinear and Adaptive Control Design*, Wiley, New York, 1995.

[9] K.L. Moore, *Iterative Learning Control for Deterministic Systems*, Springer-Verlag, New York, 1993.

[10] K. Moore, M. Daleh, and S. Bhattacharya, "Iterative Learning Control: A Survey and New Results," *Journal of Robotic Systems*, Vol. 9, pp. 563-594, 1992.