Fast Neural Learning and Control of Discrete-Time Nonlinear Systems

Liang Jin, Member, IEEE, Peter N. Nikiforuk, and Madan M. Gupta, Fellow, IEEE

Abstract—The problem of learning control for a general class of discrete-time nonlinear systems is addressed in this paper using multilayered neural networks (MNNs) with feedforward connections. A suitable extension of the concept of input-output linearization of discrete-time nonlinear systems is used to develop the control schemes for both output tracking and model reference control purposes. The ability of MNNs to model arbitrary nonlinear functions is incorporated to approximate the unknown nonlinear input-output relationship and its inverse using a new weight learning algorithm. In order to overcome the difficulties associated with simultaneous on-line identification and control in neural networks based adaptive control systems, the new learning control architectures are developed for both adaptive tracking and adaptive model reference control systems with on-line identification and control ability. The potentials of the proposed methods are demonstrated by simulation examples.

I. INTRODUCTION

The control of systems with complex, unknown, and nonlinear dynamics has become a topic of considerable research importance. In the design of conventional nonlinear control systems, three main approaches have been used: i) adaptive control, ii) Lyapunov-based adaptive control, and iii) variable structure control. Indeed, the feedback linearization technique of the nonlinear systems is especially appealing from the point of view of the nonlinear control system design. To achieve the objective of either stabilization or tracking, however, some strict assumptions were introduced regarding the structure of the uncertainties based on the completely known nonlinear models.

Advances in the area of artificial neural networks have provided the potential for new approaches to the control of complex nonlinear systems through learning processes. An artificial neural network consists of many interconnected identical simple processing units called neurons or nodes. An individual neuron sums its weighted inputs and passes the results through a threshold unit. The main potential of the neural networks for control applications can be summarized as: i) they could be used to approximate any continuous mapping to any desired degree of accuracy, ii) they perform this approximation through learning, and iii) parallel processing and fault tolerance are easily accomplished. One of the most popular neural network architectures used for control purpose is the multilayered neural networks (MNNs) with the error back-propagation (BP) algorithm. It is proved that a three-layered neural network using the back-propagation algorithm can approximate a wide range of nonlinear functions to any desired degree of accuracy [2]–[4]. To avoid modeling difficulties, a number of multilayered neural networks based controllers have been proposed [12], [19]. Regarding a control system as a mapping of control inputs into observable outputs, an appropriate mapping is realized by a MNN which is trained so that a desired response is obtained. For such types of adaptive learning control systems, the neural networks are treated as subsystems of the whole control systems. The weights of the network need to be updated using the learning algorithm, and the learning control law is constructed based on the output of the MNNs.

For neural networks based nonlinear control of a class of discrete-time nonlinear systems, Narendra, Parthasarathy [10], and Hunt, Sbarbaro [18] have recently proposed an identification, or system modeling stage, and a nonlinear control stage. The nonlinear plant is first identified off-line by the neural networks with the error back-propagation. The control is then initiated based on the model obtained by the identification. From this work, it has been concluded that for stable and efficient on-line control using the BP learning algorithm, the identification must be sufficiently accurate before any control action is initiated. In fact, the main limitations of both simultaneous control and learning is the extremely slow learning convergence of the BP algorithm. Therefore, an interesting topic in the field of neural networks based adaptive control systems is to develop the nonlinear adaptive control systems with on-line identification and control ability using neural networks with fast weight learning convergence features.

The problem of learning and control for a general class of discrete-time nonlinear systems is discussed in this paper using multilayered neural networks (MNNs) with the feedforward structure. In Section II, a suitable extension of the concept of the input-output linearization of discrete-time nonlinear systems is used to develop the control schemes for both output tracking and model reference control purposes. The fully-decoupled weight learning algorithm, named WDEKF, for MNNs is derived based on the extended Kalman filter equations in Section III. Combining the input-output linearization and control formulations with the new learning algorithm, the ability of MNNs to model arbitrary nonlinear functions is incorporated to approximate the unknown input-output relationship of a nonlinear plant and the corresponding inverse relationship. The learning control architectures are developed for both adaptive tracking and adaptive model reference con-
control with on-line identification and control ability in Section IV. The potentials of the proposed methods are demonstrated by some simulation examples in Section V. Finally, Section VI contains some conclusions.

II. INPUT-OUTPUT LINEARIZING CONTROL OF DISCRETE-TIME NONLINEAR SYSTEMS

A. I/O Linearization of Discrete-Time Nonlinear Systems

Consider a smooth single-input and single-output discrete-time nonlinear system of the form

\[ x(k + 1) = f(x(k), u(k)) \]
\[ y(k) = h(x(k)) \]  \hspace{1cm} (1)

where \( x \in \mathbb{R}^n \) is the state, and \( u, y \in \mathbb{R} \) are the input and output, respectively. The mapping \( f: \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n \) and the function \( h: \mathbb{R}^n \rightarrow \mathbb{R} \) are assumed to be analytic. The level set \( h^{-1}(0) = \{ x \in \mathbb{R}^n : h(x) = 0 \} \) defines the sliding manifold which is assumed to be sufficiently smooth.

Assume that the system (1) has relative degree \( r \) \cite{29}, \cite{30}; that is, i). \( \partial [h \circ f^r(x, u)] / \partial u = 0 \) for all \((x, u) \in \mathbb{R}^n \times \mathbb{R} \) and all \( k < r - 1 \); and ii). \( \partial [h \circ f^r(x, u)] / \partial u \neq 0 \), for all \((x, u) \) in an open and dense submanifold of \( \mathbb{R}^n \times \mathbb{R} \). Set

\[ \phi_j(x) = h \circ f^{j-1}(x), \quad j = 1, 2, \ldots, r \]  \hspace{1cm} (2)

The relative degree of the system is feedback invariant; that is, it can not be modified by feedback \cite{28}. Indeed, the relative degree \( r \) determines the time delay undergone by the input signals \( u \) before they influence the output \( y \) of the system. If \( r \) is strictly less than \( n \), it is always possible to find other \( n - r \) functions \( \phi_{r+1}(x), \ldots, \phi_n(x) \) such that the mapping

\[ \Phi(x) = [\phi_1(x), \ldots, \phi_n(x)]^T \]  \hspace{1cm} (3)

has a Jacobian matrix which is nonsingular in an open and dense subset \( M \subset \mathbb{R}^n \). The functions \( \phi_{r+1}(x), \ldots, \phi_n(x) \) are chosen in such a way that \( \phi_i(f(x, u)) = \langle \phi_i \circ f \rangle(x) \); that is, \( \phi_i(f(x, u)) \) is independent of \( u \). Therefore, \( \Phi \) is a diffeomorphism on \( M \). Setting \( z(k) = \Phi(x(k)) \), the system (1) can be transformed into a normal form in the new coordinate \( z(k) \) as follows

\[ z_1(k + 1) = z_2(k) \]
\[ z_2(k + 1) = z_3(k) \]
\[ \vdots \]
\[ z_r(k + 1) = [h \circ f^r]([\Phi^{-1}(z(k)), u(k)]) \]
\[ z_{r+1}(k + 1) = q_1(z(k), u(k)) \]
\[ \vdots \]
\[ z_n(k + 1) = q_{n-r}(z(k), u(k)) \]
\[ y(k) = z_1(k) \]  \hspace{1cm} (4)

where \( q_i(z) = \phi_i([\Phi^{-1}(z)]) \) for all \( r + 1 \leq i \leq n \), and \( z = [z_1, z_2, \ldots, z_n]^T \) \in \( \mathbb{R}^n \).

Let \( \xi = [z_1, \ldots, z_r]^T \), \( \eta = [z_{r+1}, \ldots, z_n]^T \), and the system (1) with initial condition \( x(0) \in h^{-1}(0) \) be controlled by feedback control \( u(k) = \alpha(z(k)) \) such that

\[ z_r(k + 1) = [h \circ f^r]([\Phi^{-1}(z(k)), \alpha(z(k))] = 0 \]  \hspace{1cm} (5)

for all \( k \). The subsystem

\[ \eta(k + 1) = q(0, \eta(k)) \]  \hspace{1cm} (6)

is then addressed as having "zero dynamics" \cite{28}, \cite{29}. If this subsystem is asymptotically stable, system (1) is said to be minimum phase.

The relationship between the future outputs and current state may be derived as

\[ y(k) = h(x(k)) \]
\[ y(k + 1) = h \circ f(x(k)) \]
\[ \vdots \]
\[ y(k + r - 1) = h \circ f^{r-1}(x(k)) \]  \hspace{1cm} (7)

On the other hand, the equation between the output and input is obtained as

\[ y(k + r) = h \circ f^r(x(k)), u(k)) \equiv g(x(k), u(k)) \]  \hspace{1cm} (8)

Next, a new input variable is introduced as

\[ v(k) = g(x(k), u(k)) \]  \hspace{1cm} (9)

Then

\[ y(k + r) = v(k) \]  \hspace{1cm} (10)

which obviously shows that the modified system is input-output linearized.

Since \( \partial g(x, u) / \partial u \neq 0 \) for all \((x, u) \) in an open and dense subspace of \( \mathbb{R}^n \times \mathbb{R} \). Hence, by the Implicit Function Theorem, there exists a unique local solution of the nonlinear algebraic equation (9) as follows

\[ u(k) = g^{-1}(x(k), v(k)) \]  \hspace{1cm} (11)

where \( \partial g / \partial u \neq 0 \) for all \((x, v) \) in an open and dense subspace of \( \mathbb{R}^n \times \mathbb{R} \).

B. Output Tracking and Regulation Problems

The problem of producing an output, irrespective of the initial state of the system, that converges asymptotically to a given reference function \( y_d(k) \) will now be investigated. Consider the output tracking control problem, the new input is designed as

\[ v(k) = y_d(k + r) + \sum_{j=0}^{r-1} \beta_j(y_d(k + j) - h \circ f^j(x)) \]  \hspace{1cm} (12)

where \( y_d(k) \) is a desired output. Substituting (12) into (10), the output tracking error equation is derived as

\[ e(k + r) + \beta_{r-1}e(k + r - 1) + \cdots + \beta_0e(k) = 0 \]  \hspace{1cm} (13)

where \( e(k) = y(k) - y_d(k) \) is the output tracking error. If the coefficients \( \beta_0, \beta_1, \ldots, \beta_{r-1} \) are chosen such that the \( z \)-polynomial

\[ z^r + \beta_{r-1}z^{r-1} + \cdots + \beta_0 = 0 \]  \hspace{1cm} (14)
has all its zeros inside the unit circle in the complex $z$-plane, then the output $y(k)$ of the system will track asymptotically the desired output $y_d(k)$

$$\lim_{k \to \infty} (y(k) - y_d(k)) = 0 \quad (15)$$

For a simpler exposition, the reference input is designed as

$$v(k) = y_d(k + r) \quad (16)$$

The nonlinear control $u(k) = \alpha(x(k), y_d(k + r))$ will then force the output of the system (1) to track exactly the desired output $y_d(k)$, that is,

$$y(k + r) = y_d(k + r), \quad k = 1, 2, \cdots \quad (17)$$

C. Model Reference Control Approach

For the model reference control problem, the reference output is not just a fixed function of time, but the output of a reference model, which in turn is subject to some input $r(k)$ described by equations of the form

$$\begin{cases} x_m(k + 1) = Ax_m(k) + br(k) \\ y_m(k) = cx_m(k) \end{cases} \quad (18)$$

where $x_m \in \mathbb{R}^{n_m}, r \in \mathbb{R}$, and $y_m \in \mathbb{R}$ are the state, input, and output of the model, respectively, $A$ is a $n_m \times n_m$ Hurwitz matrix, and $b$ and $c^T$ are $n_m \times 1$ vectors. The purpose of designing an equivalent control is to find a new feedback control $v(k)$ such that the output $y(k)$ of the system will asymptotically converge to the corresponding output $y_m(k)$ produced by the model under the effect of $r(k)$. Indeed, the actual feedback control law is solved by means of the nonlinear algebraic equation (9).

Note that

$$y_m(k + i) = cA^i x_m(k) + cA^{-1} br(k) + \cdots + cA br(k + i - 2) + cbr(k + i - 1) \quad (19)$$

Suppose that

$$cb = cAb = \cdots = cA^{-2}b = 0 \quad (20)$$

that is, the reference model (18) has a relative degree $\bar{r}$, and $\bar{r}$ is equal to or possibly larger than the relative degree $r$ of the system (1). The relationship between the input and output of the reference model (18) will be represented then as

$$\begin{cases} y_m(k + i) = cA^i x_m(k) \\ y_m(k + r) = cA^r x_m(k) + cA^{-1} br(k) \end{cases} \quad (21)$$

Let the new input in (10) be designed as

$$v(k) = y_m(k + r) + \sum_{j=0}^{r-1} \beta_j(y_m(k + j) - h \circ f^j(x))$$

$$= cA^{r-1} br(k) + cA^r x_m(k) + \sum_{j=0}^{r-1} \beta_j(cA^j x_m(k) - h \circ f^j(x)) \quad (22)$$

The output tracking error equation is then derived by substituting (22) into (10) as follows

$$e(k + r) + \beta_{r-1} e(k + r - 1) + \cdots + \beta_0 e(k) = 0 \quad (23)$$

where $e(k) = y(k) - y_m(k)$ is the tracking error of the output. If the coefficients $\beta_0, \beta_1, \cdots, \beta_{r-1}$ are chosen such that the $z$-polynomial

$$z^r + \beta_{r-1} z^{r-1} + \cdots + \beta_0 = 0 \quad (24)$$

has all its zeros inside the unit circle in the complex $z$-plane, the output $y(k)$ of the system will track asymptotically the desired output $y_m(k)$

$$\lim_{k \to \infty} (y(k) - y_m(k)) = 0 \quad (25)$$

III. FAST WEIGHT LEARNING OF NEURAL NETWORKS

A. Operational Equations of MNNs

In a multilayered feedforward network the neurons are organized into layers with no feedback or lateral connections. A basic structure of multilayered neural networks (MNNs) with feedforward connections is shown in Figs. 3 and 4. Let $M$ be total number of layers of the MNN including the input and output layers, the $i$th neuron in the $s$th layer be denoted by $\text{neuron}(s, i)$, $n_s$ be total number of neurons in the $s$th layer, $x_i$ be the input of the neuron $(s, i)$, $x_i$ be the output of the neuron $(s, i)$, $w_{i,k}^s$ be the linkweight coefficient from the neuron $(s, k)$ to the neuron $(s + 1, i)$, and $\theta_i$ be the threshold of the neuron $(s, i)$. Mathematically, the operation of the neuron $(s, i)$ is defined as

$$z_i^s = \begin{cases} x_{i,s}^{n_{s-i+1}} = \sum_{k=1}^{n_{s-i+1}} w_{i,k}^{s-1} z_{k,s-1}^{r}, & \text{if } 2 \leq s \leq M \end{cases}$$

and

$$x_i^s = \begin{cases} z_i^s, & \text{if } s = 1 \quad \text{or} \quad M \\ h(z_i^s), & \text{if } 2 \leq s \leq M - 1 \end{cases}$$

$\text{neuron}(s, i)$
neuron\((i, s)\)

\[
x_{1}^{s-1} \xrightarrow{u_{1}^{s-1}} x_{2}^{s-1} \xrightarrow{u_{2}^{s-1}} \ldots \xrightarrow{u_{n_{s}}^{s-1}} z_{1}^{(s)} \xrightarrow{w_{1}^{s}} \ldots \xrightarrow{w_{1}^{s}} x_{1}^{s}
\]

**Fig. 3.** Schematic representation of a multilayered feedforward neural network with three hidden layers.

**Fig. 4.** Block diagram representation for neuron \((i, s)\) in the sth layer

The function \(h(\cdot)\) may be chosen as the hyperbolic tangent sigmoidal function

\[
h(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}
\]

**B. Decomposed Kalman Filtering Structure**

It is convenient to lump all of the weights of the network into a vector that is denoted by \(w\). The output equation of the MNN at time \(k\) may then be represented by the input \(x\) and the weight vector \(w\) as follows

\[
y(k) = x^{M}(w, x(k)) + v(k)
\]

where \(y(k)\) is a \(n_{M} \times 1\) vector of the output of the network at time \(k\), \(v(k)\) is a \(n_{M} \times 1\) smooth noise vector, \(v(k)\) is assumed to be a white noise vector with the covariance matrix \(R(k)\) due to the modelling error, and the diagonal components of \(R(k)\) are equal to or slightly less than 1.

Suppose that \(x(k)\) is an input pattern, \(y_{d}(k)\) is a \(n_{M} \times 1\) vector of the desired output pattern of the MNN, and \(w(k)\) is an estimation of \(w\) at time \(k\). The purpose of the weight learning for the MNN is to estimate the weight vector \(w\) such that the output \(y(k)\) of the MNN tracks the desired output \(y_{d}(k)\), with an error which is to converge to zero, as \(k \to \infty\). Hence, if the weights of a MNN are taken into account as the unknown parameters of a nonlinear input-output system, the weight learning problem of the MNN can be phrased as the parameter identification procedure of the nonlinear system.

The nonlinear function \(x^{M}(w(k), x(k))\) can be linearized about the estimated parameter vector \(w_{i}^{s}(k-1)\) as follows

\[
x^{M}(w(k), x(k)) \approx x^{M}(w(k-1), x(k)) + \sum_{j=1}^{M-1} \sum_{i=1}^{n_{j}+1} H_{i}^{j}(k)[w_{i}^{j}(k) - w_{i}^{j}(k-1)]
\]

where the \(n_{M} \times (n_{j} + 1)\) Jacobian matrix \(H_{i}^{j}(k)\) is given by

\[
H_{i}^{j}(k) = \frac{\partial x^{M}(w(k-1), x(k))}{\partial w_{i}^{j}(k-1)}
\]

Let the \(n_{M} \times 1\) vector of the output error between the desired output and the current output of the network using the estimated parameter vector \(w_{i}^{s}(k-1)\) be defined as

\[
e(k) = y_{d}(k) - x^{M}(w(k-1), x(k))
\]

It is known that although the covariance and information implementations of the Kalman filter are algebraically equivalent, the covariance filter is the most popular, due to its relative computational simplicity when the dimension of the output is considerably smaller than the dimension of the estimated parameters. Therefore, for the weight learning problem, the decomposed extended Kalman filtering (DEKF) equations may be easily obtained from the standard extended Kalman filtering (EKF) formulation [31], [32] as follows

\[
w_{i}^{s}(k) = w_{i}^{s}(k-1) + G_{i}^{s}(k)e(k)
\]

where

\[
A(k) = \left[ R(k) + \sum_{n_{\gamma}=1}^{M-1} \sum_{\beta=1}^{n_{\gamma}+1} H_{\beta}^{\gamma}(k)G_{\beta}^{\gamma}(k)-1 \right]
\]

\[
G_{i}^{s}(k) = \sum_{\alpha=1}^{M-1} \sum_{\beta=1}^{n_{\gamma}+1} P_{i,\gamma}^{\alpha}(k-1)(H_{\beta}^{\gamma}(k)A(k))^{-1} P_{i,\gamma}^{\alpha}(k-1)
\]

\[
P_{i,m}^{k}(k) = P_{i,m}^{k}(k-1) - \sum_{\alpha=1}^{M-1} \sum_{\beta=1}^{n_{\gamma}+1} G_{i}^{s}(k)
\]

\[
1 \leq j \leq M - 1; \quad 1 \leq i \leq n_{j+1}
\]

\[
1 \leq l \leq M - 1; \quad 1 \leq m \leq n_{j+1}
\]

where \(A(k)\) is a \(n_{M} \times n_{M}\) matrix, \(G_{i}^{s}(k)\) are \((n_{j} + 1) \times n_{M}\) matrices of the filtering gain, and \(P_{i,m}^{k}(k) = (P_{i,m}^{k}(k))^{T}\) are
TABLE I

<table>
<thead>
<tr>
<th>BP Algorithm</th>
<th>WDEKF Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BP Algorithm</strong></td>
<td><strong>WDEKF Algorithm</strong></td>
</tr>
<tr>
<td>Discrete-Time Case</td>
<td></td>
</tr>
<tr>
<td>$\Delta w_{i,j}^e(k) = \sum_{v=1}^{n_M} \mu_{i,j,v}^e h_{i,j,v}^e(k) e_v(k)$</td>
<td>$\Delta w_{i,j}^e(k) = \sum_{v=1}^{n_M} \mu_{i,j,v}^e h_{i,j,v}^e(k) e_v(k)$</td>
</tr>
<tr>
<td>$\mu_{i,j,v}^e = \text{constant}$</td>
<td>$\mu_{i,j,v}^e(k) = p_{i,j,v}^e(k-1) a_v(k)$</td>
</tr>
<tr>
<td>$\Delta p_{i,j}^e(k) = -\sum_{v=1}^{n_M} \mu_{i,j,v}^e(k)(a_{i,j,V}(k))^2 p_{i,j}^e(k-1)$</td>
<td>$a_v(k) = \frac{1}{\mu_{i,j,v}^e(k-1) + \sum_{s=1}^{M} \sum_{i=1}^{n_s} (a_{i,j,v}(k))^2 p_{i,j}^e(k-1)}$</td>
</tr>
<tr>
<td>Continuous-Time Case</td>
<td></td>
</tr>
<tr>
<td>$\frac{dw_{i,j}^e(t)}{dt} = \sum_{v=1}^{n_M} \mu_{i,j,v}^e(t) h_{i,j,v}(t) e_v(t)$</td>
<td>$\frac{dw_{i,j}^e(t)}{dt} = \sum_{v=1}^{n_M} \mu_{i,j,v}^e(t) h_{i,j,v}(t) e_v(t)$</td>
</tr>
<tr>
<td>$\mu_{i,j,v}^e = \text{constant}$</td>
<td>$\mu_{i,j,v}^e(k) = p_{i,j,v}^e(k)$</td>
</tr>
<tr>
<td>$\frac{dp_{i,j}^e(t)}{dt} = -\sum_{v=1}^{n_M} \mu_{i,j,v}^e(t)(h_{i,j,v}(t))^2 p_{i,j}^e(t)$</td>
<td></td>
</tr>
</tbody>
</table>

$(n_x + 1) \times (n_t + 1)$ matrices of the error covariance between the estimations $w(k)$ and $w^e(k)$. A comparison of the EKF and DEKF shows that even if the storage requirements for the error covariance matrices $P_{i,j}^e(k)$ are same, each step learning algorithm requires $O((n_x + 1))^2$ storage for the $P_{i,j}^e$ matrices, whereas the DEKF algorithm avoids the multiplications of the matrices with very high dimensions in every iteration.

C. Fast Weight-Decoupled Kalman Filtering Algorithm

Based on the decomposed version of the extended Kalman filtering (DEKF) equations obtained in the previous subsection, a fully-decoupled recursive estimation learning algorithm, named the weight-decoupled extended Kalman filter (WDEKF) learning algorithm, for MNNs may be proposed [11], [15] as follows

$$w_{i,j}^e(k) = w_{i,j}^e(k-1) + \sum_{v=1}^{n_M} \mu_{i,j,v}^e(k) h_{i,j,v}^e(k) e_v(k) \quad \text{(35)}$$

where

$$\mu_{i,j,v}^e(k) = p_{i,j,v}^e(k-1) a_v(k)$$

$$p_{i,j}^e(k) = \left(1 - \sum_{v=1}^{n_M} \mu_{i,j,v}^e(k)(h_{i,j,v}^e(k))^2\right) p_{i,j}^e(k-1)$$

$$a_v(k) = \frac{1}{\mu_{i,j,v}^e(k-1) + \sum_{s=1}^{M} \sum_{j=1}^{n_s} (h_{i,j,v}(k))^2 p_{i,j}^e(k-1)}$$

For $e_v(k) = y_d(k) - x_M^e(w(k-1), x(k))$ and $h_{i,j,v}^e(k) = \frac{\partial e_v(k)}{\partial w_{i,j}^e(k-1)}$ with $1 \leq v \leq n_M; \quad 1 \leq s \leq M; \quad 1 \leq i \leq n_S; \quad 1 \leq j \leq n_{s-1}$

The WDEKF algorithm is, of course, computationally more complex than the gradient descent based back propagation algorithm, however, the convergence rate of the former is much faster than the that of the latter. This characteristic of the WDEKF algorithm is very useful to neural networks based control systems. Eq. (35) is similar to the weight update equation of the conventional BP learning algorithm, which is a learning algorithm with a constant learning rate. In the course of numerical simulations with the conventional BP algorithm, it becomes clear that the learning rate $\mu$ is critical. If $\mu$ is too large, the algorithm will not converge, while if $\mu$ is too small, the convergence will be too slow to be practical. The WDEKF learning algorithm overcomes this difficulty using a varying learning rate which is adjusted adaptively to reach the
optimal value at each instant. In other words, the WDEKF may be treated as a type of the BP algorithm with an optimal learning rate.

IV. LEARNING CONTROL USING NEURAL NETWORKS

A. Nonlinear Control Using Neural Networks

Recently, several independently studies have found that a three-layered neural network using the back-propagation algorithm can approximate a wide range of nonlinear functions to any desired degree of accuracy. In this section, the output-input linearized control technique combining the WDEKF learning algorithm with a fast convergence feature is applied to develop the nonlinear adaptive control systems with on-line identification and control ability. Let the multilayered neural networks be used to construct nonlinear learning control input linearized control technique combining the WDEKF learning algorithm with a fast convergence feature is applied to develop the nonlinear adaptive control systems with on-line identification and control ability. Let the multilayered neural networks be used to construct nonlinear learning control schemes for the purposes of adaptively tracking the desired output $y_d(k)$ and model reference control. Suppose that both the order $n$ and the relative degree $r$ of the system (1) are known, but the nonlinear functions $f(\cdot)$ and $h(\cdot)$ are unknown. Let the multilayered neural networks $NN_2$ with the weight $\sigma$ and the relative degree $T$ be used to approximate the nonlinear control $\alpha(\cdot)$ and the nonlinear input-output relation $g(\cdot)$, respectively. Then, (10) and (7) may be governed by using the neural networks as follows

$$\tilde{u}(k) = \tilde{\alpha}(x(k), u(k), \sigma)$$
$$\tilde{g}(k + r) = \tilde{g}(x(k), \tilde{u}(k), w)$$

(37)
(38)

where $x(\cdot)$ and $u(\cdot)$ are the inputs of the neural network $NN_2$ used to approximate nonlinear function $\alpha(x, u)$, $\tilde{g}(\cdot)$ is the output of the neural network $NN_2$, $x(\cdot)$ and $\tilde{u}(\cdot)$ are the inputs of the neural network $NN_2$ used to approximate nonlinear function $g(x, u)$, and $\tilde{u}(\cdot)$ is the output of the network $NN_2$. The networks $NN_2$ and $NN_1$ are trained to represent the input-output model of the plant and its inverse on-line, respectively.

From the discussion in Section II, the external input $v(k)$ in (37) is designed as

$$v(k) = \begin{cases} y_d(k + r) + \sum_{j=0}^{r-1} \beta_j(y_d(k + j) - h \circ f^j(x)); \\
\text{for output tracking control} \\
y_m(k + r) + \sum_{j=0}^{r-1} \beta_j(y_m(k + j) - h \circ f^j(x)); \\
\text{for model reference control} 
\end{cases}$$

(39)

Hence, the neural models (37) and (38) may be represented as

$$\tilde{u}(k) = \tilde{\alpha}(x(k), y_c(k), \ldots, y_c(k + r), \sigma)$$
$$\tilde{g}(k + r) = \tilde{g}(x(k), \tilde{u}(k), w)$$

(40)
(41)

where $y_c(k)$ is a known desired output or an output of the reference model at time $k$, $x(k)$ and $y_c(k), \ldots, y_c(k + r)$ are the inputs of the network $NN_1$ at time $k$. In order to define the errors, (41) may be rewritten as

$$\tilde{g}(k) = \tilde{g}(x(k - r), \tilde{u}(k - r), w)$$

(42)

Fig. 5. On-line identification and control using neural networks.

Let the weight vectors $w(k)$ and $\sigma(k)$ be the estimations of the $w$ and $\sigma$, respectively, at time $k$, then, the estimated models of (40) and (42) at time $k$ are expressed as

$$u^*(k) = \hat{u}(x(k), y_c(k), \ldots, y_c(k + r), \sigma(k))$$
$$g^*(k) = \hat{g}(x(k - r), u^*(k - r), w(k))$$

(43)
(44)

where $u^*(k)$ and $g^*(k)$ are, respectively, the outputs of the networks $NN_2$ and $NN_1$ with the estimated weight vectors $\sigma(k)$ and $w(k)$.

The errors used to train the neural networks $NN_1$ and $NN_2$ are defined as

$$e_1(k) = y^*(k) - y_c(k)$$
$$e_2(k) = y^*(k) - y(k)$$

(45)
(46)

where $y(k)$ is an output of the plant at time $k$, $e_1(k)$ used to train the neural network $NN_1$ is a difference between the outputs of the estimated model (44) and the plant, and $e_2(k)$ used to the neural network $NN_2$ is a difference between the output of the estimated model (44) and the desired output.

Note that inequality

$$|y(k) - y_c(k)| \leq |y^*(k) - y_c(k)| + |y^*(k) - y(k)|$$

$$= |e_1(k)| + |e_2(k)|$$

(47)

holds. Hence, if the neural networks $NN_1$ and $NN_2$ are trained such that $\lim_{k \to \infty} e_1(k) = 0$ and $\lim_{k \to \infty} e_2(k) = 0$, the asymptotically output tracking condition is then satisfied; that is,

$$\lim_{k \to \infty} (y(k) - y_c(k)) = 0$$

(48)

The partial derivatives of the errors with respect to the weight vectors $\sigma(k)$ and $w(k)$ are derived as

$$\frac{\partial e_1(k)}{\partial \sigma(k)} = \frac{\partial y^*(k)}{\partial \sigma(k)} - \frac{\partial u^*(k - r)}{\partial \sigma(k)}$$
$$\frac{\partial e_2(k)}{\partial \sigma(k)} = \frac{\partial y^*(k)}{\partial \sigma(k)}$$

(49)
(50)

where

$$u^*(k - r) = \hat{\alpha}(x(k - r), y_c(k - r), \ldots, y_c(k), \sigma(k))$$

(51)
B. Affine Nonlinear Input-Output Systems

If the input-output relation (8) is an affine nonlinear system to the control input \( u(k) \); that is, the input \( u(k) \) is seen to occur linearly in the difference equation (8), then, (8) may be represented as

\[
y(k + r) = g_1(x(k)) + g_2(x(k))u(k)
\]

(52)

where \( g_2(x(k)) \) is bounded away from zero.

Once both \( g_1(x(k)) \) and \( g_2(x(k)) \) are known, the inverse of the function \( g_1(x(k)) \) is well defined. The most convenient control structure for exactly output tracking is then the one in which the input variable \( u(k) \) is set equal to

\[
y(k + r) = \frac{y_d(k + r) - g_1(x(k))}{g_2(x(k))}
\]

(53)

Since the continuous functions \( g_1(\cdot) \) and \( g_2(\cdot) \) are unknown, a neural network \( NN_3 \) is introduced to approximate the nonlinear functions \( g_1(\cdot) \) and \( g_2(\cdot) \) using the on-line WDEKF learning algorithm. Hence, the neural network modelling of the nonlinear system (52) can be given by

\[
y(k + r) = \hat{g}_1(x(k), w) + \hat{g}_2(x(k), w)u(k)
\]

(54)

or

\[
y(k) = \hat{g}_1(x(k - r), w) + \hat{g}_2(x(k - r), w)u(k - r)
\]

(55)

where \( x(k) \) is the input of the \( NN_3 \), \( \hat{g}_1(x(k), w) \) and \( \hat{g}_2(x(k), w) \) are the two outputs of the \( NN_3 \), and \( w \) is the weight vector of the \( NN_3 \). Let \( y(k) \) be an estimate of the weight \( w \) at time \( k \). The estimated model with the estimated weight vector \( \hat{w}(k) \) of (52) is then given as

\[
y^*(k) = \hat{g}_1(x(k - r), \hat{w}(k)) + \hat{g}_2(x(k - r), \hat{w}(k))u(k - r)
\]

(56)

According to the control law (53), the nonlinear learning control law is designed as

\[
u(k) = \frac{y_d(k + r) - \hat{g}_1(x(k), \hat{w}(k))}{\hat{g}_2(x(k), \hat{w}(k))}
\]

(57)

and the error that will be used to train the \( NN_3 \) is defined as

\[
e(k) = y^*(k) - y(k)
= \hat{g}_1(x(k - r), \hat{w}(k)) + \hat{g}_2(x(k - r), \hat{w}(k))u(k - r) - y(k)
\]

(58)

Let

\[
u^*(k - r) = \frac{y_d(k) - \hat{g}_1(x(k - r), \hat{w}(k))}{\hat{g}_2(x(k - r), \hat{w}(k))}
\]

(59)

If the weight learning algorithm converges as \( k \to \infty \), then

\[
\lim_{k \to \infty} (\hat{w}(k) - w(k - r)) = 0
\]

(60)

Hence

\[
\lim_{k \to \infty} (v^*(k - r) - u(k - r)) = 0
\]

(61)

Based on the control law (59), one can show that the output of the estimated model is made to converge to the desired output

\[
\lim_{k \to \infty} (y^*(k) - y_d(k)) = 0
\]

(62)

Note that

\[
|y(k) - y_d(k)| \leq |y^*(k) - y_d(k)| + |y^*(k) - y(k)|
\]

(63)

Hence, if the output of the estimated model is forced to track the output of the plant

\[
\lim_{k \to \infty} e(k) = 0
\]

(64)

the desired output \( y_d(k) \) is then asymptotically tracked by the output \( y(k) \) of the plant

\[
\lim_{k \to \infty} (y(k) - y_d(k)) = 0.
\]

(65)

For the WDEKF learning algorithm discussed in preceding sections, the partial differentials of the weights of the \( NN_3 \) with respect to the learning error \( e(k) \) can be obtained as follows

\[
\frac{\partial e(k)}{\partial w(k)} = \frac{\partial \hat{g}_1(x(k - r), w(k))}{\partial w(k)} + \frac{\partial \hat{g}_2(x(k - r), w(k))}{\partial w(k)}u(k - r)
\]

(66)

The above control algorithm avoids the assumption that the sign of the function \( g_2(\cdot) \) must be known, what is necessary for both the neural networks based model reference adaptive control and the self-turning control proposed by [12] and [17], respectively.

V. SIMULATION EXAMPLES

Example 1. (Nonlinear output tracking control problem)

Assume that the plant is a single-input and single-output system with unknown dynamics described by the nonlinear difference equation

\[
y(k + 1) = g[y(k), y(k - 1), y(k - 2), u(k), u(k - 1)]
\]

(67)

where \( y(k) \) is the current output, \( u(k) \) is the current control input. It is easy to show that the order and the relative degree are 4 and 1, respectively. Denote \( x_1(k) = y(k) \), \( x_2(k) = y(k - 1) \), \( x_3(k) = y(k - 2) \), and \( x_4(k) = u(k - 1) \). Let the unknown plant \( g(\cdot) \) in (67) for numerical simulation have the form [12]

\[
g(x_1, x_2, x_3, x_4, u) = \frac{x_1 x_2 x_3 x_4 (x_3 - 1 - \Delta \alpha) + (1 + \Delta \beta) u}{1 + x_3^2 + x_4^2}
\]

(68)

where \( \Delta \alpha \) and \( \Delta \beta \) are assumed to be the varying values of parameters, and \( \Delta \beta \neq -1 \). In a normal case, \( \Delta \alpha = 0 \) and
Δβ = 0. In order to study the nonlinear output tracking control problem for the system (67) using the approach developed in Section IV, the nonlinear system (67) is represented by a following affine nonlinear form

\[ y(k+1) = g_1(y(k), y(k-1), y(k-2), u(k-1)) + g_2(y(k), y(k-1), y(k-2), u(k-1))u(k) = g_1(x(k)) + g_2(x(k))u(k) \]  

(69)

where \( x(k) = [x_1(k), x_2(k), x_3(k), x_4(k)]^T \). According to the system model (67), it is easy to understand that the unknown nonlinear functions \( g_1(\cdot) \) and \( g_2(\cdot) \) in plant (67) for numerical simulation should be of the following form

\[ g_1(x_1, x_2, x_3, x_4) = \frac{x_1 x_2 x_3 x_4 (x_3 - 1 - \Delta \alpha)}{1 + x_2^2 + x_3^2} \]  

(70)

\[ g_2(x_1, x_2, x_3, x_4) = \frac{1 + \Delta \beta}{1 + x_2^2 + x_3^2} \]  

(71)

Since \( 1 + x_2^2 + x_3^2 \neq 0 \), for any \( x_2 \in \mathbb{R} \), and \( x_3 \in \mathbb{R} \), the nonlinear tracking problem may be solved based on the results obtained in the previous section. A four-layered neural network with two hidden layers, four inputs \( x_1(k), x_2(k), x_3(k), \) and \( x_4(k) \), and two outputs were used to identify the nonlinear model \( g_1(\cdot) \) and \( g_2(\cdot) \) on-line by means of the WDEKF learning algorithm. For this purpose, the number of hidden neurons of the network was chosen to be \( n_2 = n_3 = 5 \).

Let the desired output \( y_d(k) \) be set as a square wave signal, the initial values of the weights be chosen randomly in the interval \([-1,1]\), and the initial variances of the weights be set as \( \sigma_w^2(0) = 1000 \). Fig. 7 show the histories of the outputs of the desired and controlled plant, output tracking error \( e(k) \), and the control input \( u(k) \), respectively, when \( A_\alpha = 0 \) and \( A_\beta = 0 \). Let \( A_\alpha = 3.0 \), and \( A_\beta = 3.0 \), for \( k \geq 200 \), the robustness of the system to the changes in the parameters of the system is shown in Fig. 8. These figures present the outputs of the desired and controlled plant, the output tracking error \( e(k) \), and the control input \( u(k) \), respectively. The oscillation around \( k = 200 \) is due to the varying of the plant parameters. The simulation results in Fig. 8 show that the output tracking control of the unknown system (67) is performed perfectly using the WDEKF learning algorithm. Although, the first few steps responses of the controlled plant oscillated around the desired output \( y_d(k) \), the tracking error converged rapidly to zero after the learning period, whereas the initial values \( w_{ij}^2(0) \) of the weights could be chosen arbitrarily. On the other hand, the control algorithm is very robust for the varying of the system parameters.

**Example 2. (Nonlinear model following control)**

The unknown nonlinear system in this case is described by the difference equation of the form

\[ y(k+1) = g(y(k), y(k-1), y(k-2), u(k-1)) + \Delta g(y(k), y(k-1), y(k-2), u(k-1)) + \beta u(k) \]

\[ = g(x(k)) + \Delta g(x(k)) + \beta u(k) \]  

(72)

where \( x(k) = [x_1(k), x_2(k), x_3(k), x_4(k)]^T = [y(k), y(k-1), y(k-2), u(k-1)]^T \) is the state vector of the system. The specific plant used in the simulation studies was

![Fig. 7. (a) Desired output \( y_d(k) \) and controlled plant output \( y(k) \). (b) Output tracking error \( e(k) \).](image)

\[ g(\cdot) = \frac{x_1 x_2 x_3}{1 + x_2^2 + x_3^2} + 0.5 x_4 \]  

(73)

\[ \beta = 2.0 \]  

(74)

and \( \Delta g(\cdot) \) in (72) was the structure varying term of the plant. The reference model was given by the stable linear system.
Fig. 8. (a) Desired output $y_d(k)$ and controlled $y(k)$. (b) Output tracking error $e(k)$. (c) Nonlinear learning control $u(k)$.

\[ y_m(k+1) = 0.12y_m(k) + 0.22y_m(k-1) - 0.17y_m(k-2) + 0.33r(k) \]  

where $r(k)$ is the uniformly bounded reference input. The structure of the MNN was designed to be similar to that in Example 1. Using a plant that is time-invariant; that is, $\Delta g(\cdot) = 0$, the responses of the reference model and the tracking error $e(k)$, and the control action $u(k)$ when $r(k) = \sin(2\pi k/25)$ are shown in Fig. 9.

The structure varying term was then set to

\[ \Delta g(\cdot) = \begin{cases} 
0 & \text{if } k \leq 50 \\
0.01x_2x_3/(1+x_3^2) & \text{if } k > 50 
\end{cases} \]
JIN et al.: FAST NEURAL LEARNING AND CONTROL OF DISCRETE-TIME NONLINEAR SYSTEMS

VI. CONCLUSIONS

Some multilayered neural networks based nonlinear control algorithms are proposed in this paper using the input-output linearization concept of nonlinear systems and the weight learning process. As in all adaptive control techniques, the MNNs based learning control schemes combine identification and control performed by the on-line adaptively weight learning process. The ability of the MNNs to model arbitrary nonlinear functions is applied to approximate the unknown input-output relationship of a system and the corresponding inverse relationship. Since the unknown nonlinear systems are on-line modeled and controlled by the input-output dependent neural networks, the control mechanisms are very robust for the varying of the system parameters and structures that do not change the relative degree of the initial models.

The simulation studies on SISO nonlinear dynamic systems revealed that on-line identification and control using the methods suggested can be very effective. The extension of this method to the identification and control of unknown MIMO nonlinear control systems is straightforward. Behind the adaptive learning control schemes proposed in this paper, many practical and theoretical questions can be raised. An obvious and important aspect of the learning and control algorithms in the future will be the theoretical analysis of the convergence of the weight learning algorithm, and the stability of the adaptive control schemes.

REFERENCES


