SIMPLE RECURRENT NEURAL NETWORK-BASED ADAPTIVE PREDICTIVE CONTROL FOR NONLINEAR SYSTEMS

Xiang Li, Zengqiang Chen and Zhuzhi Yuan

ABSTRACT

Making use of the neural network universal approximation ability, a nonlinear predictive control scheme is studied in this paper. On the basis of a uniform structure of simple recurrent neural networks, a one-step neural predictive controller (OSNPC) is designed. The whole closed-loop system’s asymptotic stability and passivity are discussed, and stable conditions for the learning rate are determined based on the Lyapunov stability theory for the whole neural system. The effectiveness of OSNPC is verified via exhaustive simulations.

KeyWords: Neural adaptive predictive control, simple recurrent neural networks, stability passivity.

I. INTRODUCTION

Adaptive control theories have flourished and gained great success in various research fields as well as in industrial applications, which, in turn, has helped the adaptive control theory become mature and systematic [23]. One important branch of this field is predictive control [10], which is now popular owing to its special characteristics, such as the capabilities of model-based prediction, rolling optimization and feedback tuning. However, due to the assumption of linearity of the unknown system parameters, predictive control encounters great difficulties when confronted with systems that have high nonlinearity and complexity.

As a mathematical model for the human brain, the neural network has been commonly applied in most branches of natural science, not only the control systems field. Research has shown that multi-layer feedforward neural networks (MFNN) are capable of approximating a nonlinear system within an arbitrarily small error, both in cases where there are sufficiently many neurons and where a finite number of neurons are in the hidden layer [3,8,9]. Another hot area of neural network research is recurrent neural networks (RNN), on which the journal of IEEE Transaction of Neural Networks published a special issue in 1994. As combinations of fully connected recurrent neural networks and MFNN, several kinds of simplified recurrent neural networks have emerged and been applied to control systems. Because it is a dynamical mapping of a real system with fewer parameters (i.e. weight values) to tune, simplified recurrent neural networks have many advantages over MFNN and the fully connected RNN [17, 21,22,27,28,30]. A review of locally recurrent globally feedforward network architectures can be found in [29].

The neural network universal approximation ability provides an alternative means of controlling nonlinear systems [1,13,25], and MFNN has become a basic tool for control. Various kinds of adaptive control schemes and fundamental properties of neural control have been studied; interested readers can refer to such works as [4,5,6,14,18,19,20, 24]. However, most of these elegant works only considered the control cost function in the form of $J = (y_s(k) - y_k)^2$, where $y_s(k)$ is the set value for the controlled plant and $y_k$ is the response of the real system plant. Nevertheless, most of these works aimed at feedback linearizable or affine nonlinearities, thus unavoidably limiting their real-time application.

A nonlinear GMV-based uniform simple recurrent neural network control scheme was presented in [22], aimed at an NARMAX model with non-minimum phase nonlinearity. To bring a predictive mechanism into this neural control scheme, and to employ an adaptive predictive control algorithm for flexible control of general nonlinear systems, this paper develops a simple recurrent neural network with two neural networks functioning as the model and the controller, respectively. A one-step neural predictive control (OSNPC) mechanism is
introduced, and the stability and passivity of the whole closed-loop neural system are demonstrated. In addition, some simulation examples are given to verify the effectiveness of the proposed OSNPC.

The paper is organized as follows: Some system background and related knowledge are given in Section II. OSNPC is then fully developed in Section III. The closed-loop system stability as well as passivity is demonstrated in Section IV. Based on some simulations, in Section V, the effectiveness of OSNPC is demonstrated. In Section VI, conclusions are drawn.

II. SYSTEM BACKGROUND AND RELATED KNOWLEDGE

2.1 Simple recurrent neural networks structure

Various kinds of simplified recurrent neural network structures have been constructed [12,17,27,30]. Notably, they all can be unified into a uniform structure as a simple recurrent neural network (SRNN) [22]. This is illustrated as Fig. 1.

![SRNN uniform structure](image)

Fig. 1. SRNN uniform structure.

Indeed, from Fig. 1, one can write the network descriptive equation as

\[
\begin{align*}
O(k) &= W^oS(k), \\
S(k) &= h(W^S^S(k) + W^XX(k - 1)), \\
S^c(k) &= S(k - 1),
\end{align*}
\]

(1)

where \(h(\cdot)\) is a sigmoidal activation function. The context layer \(S^c(k) = S(k - 1)\) functions as the memory of the hidden layer state, but it also brings more abundant dynamics into SRNN than exist in the multi-layer feedforward neural networks [22].

2.2 Lyapunov stability theory

In the literature on nonlinear control, stability in the sense of Lyapunov is an important property for control schemes, and many works have been devoted to this topic [7]. Here, only the following theorem of the second method of Lyapunov stability in [7] is reviewed.

**Theorem 2.1. (Second method of Lyapunov for discrete-time nonautonomous systems)** Let \(x^0 = 0\) be an equilibrium of the “nonautonomous” system

\[
x_{k+1} = f_k(x_k),
\]

(2)

where \(f_k : \Omega \to \mathbb{R}^n\) is continuously differentiable in a neighborhood of the origin, \(\Omega \subseteq \mathbb{R}^n\). Then, the system (2) is globally (over the entire domain \(\Omega\)) and asymptotically stable about its zero equilibrium if there exists a scalar-valued function, \(V(x_k, k)\), defined on \(\Omega\) and continuous in \(x_k\), such that

(i) \(V(0, k) = 0\) for all \(k \geq k_0\);

(ii) \(V(x_k, k) > 0\) for all \(x_k \neq 0\) in \(\Omega\) and for all \(k \geq k_0\);

(iii) \(\Delta V(x_k, k) := V(x_{k+1}, k+1) - V(x_k, k) < 0\) for all \(x_k \neq 0\) in \(\Omega\) and all \(k \geq k_0\);

(iv) \(0 < \Psi(\tau) < V(x_k, k)\) for all \(k \geq k_0\), where \(\Psi(\tau)\) is a positive continuous function defined on \(\Omega\), satisfying \(\Psi(\tau) < \Psi(\tau^+\to\infty)\) and \(\lim_{\tau\to\infty}\Psi(\tau) = \infty\) monotonically.

2.3 Passive and dissipative system

There are detailed descriptions of continuous and discrete-time systems in [14,15,16,21]. For convenience, the related definitions of discrete-time systems given in [14] are adopted here.

**Definition 1.** A function \(E : l^m_\omega(Z_\omega) \times l^p_\omega(Z_\omega) \to \mathbb{R}\) is called an energy function if it has the quadratic form

\[
E(u, y, T_m) = \left\langle y, Su \right\rangle_{T_m} + \left\langle u, Ru \right\rangle_{T_m},
\]

(3)

**Definition 2.** A system with input \(u(k)\) and output \(y(k)\) is said to be passive if it verifies the energy function form

\[
\Delta L = y^T(k)Su(k) + u^T(k)Ru(k) - g(k),
\]

(4)

where \(L(k)\) is lower bounded and \(g(k) \geq 0\).

**Definition 3.** A system is defined as dissipative if it is passive and, in addition,

\[
E(u, y, T) \neq 0 \Rightarrow \sum_{k=0}^{T} g(k) > 0, \quad \forall T > 0.
\]

(5)

III. NEURAL ADAPTIVE PREDICTIVE CONTROL SCHEME

3.1 One-step neural predictive control

Consider a general nonlinear discrete-time system
(not necessarily the affine system), which is in the NARMAX form

$$y(k) = f(y(k - 1), \ldots, y(k - n), u(k - 1), \ldots, u(k - m)).$$  

(6)

In the following, two simple recurrent neural networks are employed, which constitute the whole OSNPC for the nonlinear system (6). One NNM models the nonlinear system in order to approximate its dynamics and to give a one-step prediction of the plant, while the other functions as the neural predictive controller (NPC). The whole OSNPC structure is shown in Fig. 2, where the solid line is the process of NNM and NPC, and the dashed line is the NNM as a one-step predictor. Two cost functions ($J_f(k)$ for NNM and $J_s(k)$ for NPC) as shown in Fig. 2 are as follows:

$$J_f(k) = \frac{1}{2}[y(k) - \hat{y}(k)]^2,$$

(7)

$$J_s(k) = \frac{1}{2}[y_s(k + 1) - \bar{y}(k + 1)]^2 + \frac{1}{2}\Delta u^2(k),$$

(8)

where $\hat{y}(k)$, resulting from NNM, is the estimate of $y(k)$, $\bar{y}(k + 1)$, is the one-step prediction of the system response obtained by the NNM after it gets its weight value modifications in the $k$th time step, and $y_s(k + 1)$ is the set value at the next time step.

NNM and NPC both have the SRNN structure shown in Fig. 1. More precisely, both NNM and NPC have input/output mappings as follows:

**NNM:** $\hat{y}(k) = O_f(k) = \sum_{j=1}^{n_s} W_{y_i}(k)S_j(k)$,

(9)

$$S_j(k) = h\left(\sum_{i=1}^{n} W_{i_j}(k)S_i(k) + \sum_{i=1}^{n} W_{i_j}(k)X_i(k)\right).$$

(10)

**NPC:** $u(k) = O_s(k) = \sum_{j=1}^{n_s} V_{o_j}(k)T_j(k)$,

(11)

where $W^e$, $W^c$, $W^f$ and $V^e$, $V^c$, $V^f$ are weight values for NNM and NPC between the hidden layer and the output layer, the context layer and the hidden layer, and the input layer and the hidden layer, respectively; $S'(k) = S(k - 1) and T'(k) = T(k - 1)$ are the NNM and NPC context layers’ state values, respectively; $X^e$, $X^c$ are NNM and NPC input sequences, respectively; $h(\cdot), g(\cdot)$ are NNM and NPC activation functions in the hidden layer, respectively; and $h(x) = g(x) = \frac{1}{1 + e^{-x}}$. To simplify the notation, we replace $\Delta u(k)$ with $\tilde{u}(k)$, thereby obtaining

$$u(k) = u(k - 1) + \Delta u(k) = u(k - 1) + \tilde{u}(k).$$

(13)

Therefore, (8) can be rewritten as

$$J_s(k) = \frac{1}{2}[y_s(k + 1) - \bar{y}(k + 1)]^2 + \frac{1}{2}\tilde{u}^2(k).$$

(14)

To summarize, a one-step neural predictive controller has been outlined as follows:

1) NNM feedforward propagates at time step $k$, approximates the system plant to get $\hat{y}(k)$, and updates $W(k)$ under the cost function (7).
2) NPC feedforward propagates to obtain $u(k)$.
3) NNM is used as the plant predictive model to obtain the one-step prediction $\bar{y}(k + 1)$.
4) The cost function (14) is used to update the NPC weight values $V(k)$.
5) $k := k + 1$, returns to 1.

### 3.2. DBP training algorithm for SRNN

With the uniform SRNN structure shown in Fig. 1, the weight values of NNM and NPC, $W^e$, $W^c$, $W^f$ and $V^e$, $V^c$, $V^f$, are updated by the dynamic backpropagation algorithm (DBP) detailed as below, where the weight value learning rates $\eta_f$, $\eta_s$ are both positive:

$$W(k + 1) = W(k) + \eta_f (-\frac{\partial J_f(k)}{\partial W(k)}),$$

(15)

$$V(k + 1) = V(k) + \eta_s (-\frac{\partial J_s(k)}{\partial V(k)}).$$

(16)

For NNM weight value updating, it follows from (7) and (15) that

$$\frac{\partial J_f(k)}{\partial W(k)} = e_f(k)\frac{\partial e_f(k)}{\partial W(k)} = -e_f(k)\frac{\partial \hat{y}(k)}{\partial W(k)} = -e_f(k)\frac{\partial O_f(k)}{\partial W(k)}.$$  

(17)
where $e_i(k) = y(k) - \hat{y}(k)$. Therefore, the weight tuning for NNM is

$$\Delta W(k) = W(k + 1) - W(k) = \eta e_i(k) \frac{\partial O_i(k)}{\partial W(k)}$$  \hspace{1cm} (18)

Let

$$h_i' = \frac{dh_i(x)}{dx} = \sum_{i=1}^{n_f} w_i' \beta x_i'$$
and

$$g_i' = \frac{dg_i(x)}{dx} = \sum_{i=1}^{n_f} v_i' \beta x_i'$$

For all parts of the weight values in NNM, $W^\alpha(k), W'(k)$, it follows from (9) and (10) that $\frac{\partial O_i(k)}{\partial W(k)}$ can be obtained as follows:

$$\frac{\partial O_i(k)}{\partial W^\alpha(k)} = S_i(k)$$

$$\frac{\partial O_i(k)}{\partial W'(k)} = \frac{\partial O_i(k)}{\partial S_i(k)} \frac{\partial S_i(k)}{\partial W^\alpha_i(k)} = W^\alpha_i(k) h_i'(S_i(k - 1) + \sum_{i=1}^{n_f} W_i' S_i(k - 1) \frac{\partial S_i(k - 1)}{\partial W^\alpha_i(k)}.$$ \hspace{1cm} (19)

In general, the modification of $W(k)$ at every time step is very small [27]. Therefore, it is presumed that $\frac{\partial S_i(k - 1)}{\partial W^\alpha_i(k)} = \frac{\partial S_i(k - 1)}{\partial W^\alpha_i(k - 1)}$, and (20) can be further written as

$$\frac{\partial O_i(k)}{\partial W^\alpha_i(k)} = \frac{\partial O_i(k)}{\partial S_i(k)} \frac{\partial S_i(k)}{\partial W^\alpha_i(k - 1)} = W^\alpha_i(k) h_i'(S_i(k - 1) + \sum_{i=1}^{n_f} W_i' S_i(k - 1) \frac{\partial S_i(k - 1)}{\partial W^\alpha_i(k - 1)}.$$ \hspace{1cm} (21)

As for weight value learning for NPC, we first get the system plant one-step prediction $\bar{y}(k + 1)$ with NNM. Let $e_i(k) = y_i(k - 1) - \bar{y}(k + 1)$. From (16), we obtain

$$\frac{\partial u(k)}{\partial V(k)} = e_i(k) \frac{\partial e_i(k)}{\partial V(k)} + \lambda \bar{u}(k) \frac{\partial \bar{u}(k)}{\partial V(k)}$$

$$\Delta V(k) = V(k + 1) - V(k) = \eta \| e_i(k) \| \frac{\partial O_i(k)}{\partial V(k)}$$

Let $u(k)$ be the first element in the sequence $X_i(k)$, and let $\overline{\gamma} = \frac{\partial \gamma(k + 1)}{\partial u(k)}$. Then, we further obtain $\overline{\gamma} = \sum_{i=1}^{n_f} \gamma_i(k + 1) h_i'(S_i(k + 1)), so that

$$\frac{\partial \gamma(k + 1)}{\partial V(k)} = \frac{\partial \gamma(k + 1)}{\partial u(k)} \frac{\partial u(k)}{\partial V(k)} = \frac{\partial \gamma(k + 1)}{\partial O_i(k)} \frac{\partial O_i(k)}{\partial V(k)} = \overline{\gamma} \frac{\partial O_i(k)}{\partial V(k)}.$$ (23)

For NPC’s $V^\alpha(k) V'(k)$ $k$ we can get $\frac{\partial O_i(k)}{\partial V(k)}$, similar to that of NNM, as

$$\frac{\partial O_i(k)}{\partial V'(k)} = T_i(k)$$

$$\frac{\partial O_i(k)}{\partial V'(k)} = \frac{\partial O_i(k)}{\partial T_i(k)} \frac{\partial T_i(k)}{\partial V'(k)} = V^\alpha_i(k) g_i'(X^\alpha_i(k)).$$ (24)

$$\frac{\partial O_i(k)}{\partial V'(k)} = \frac{\partial O_i(k)}{\partial T_i(k)} \frac{\partial T_i(k)}{\partial V'(k)} = V^\alpha_i(k) g_i'(T_i(k) - 1) + \sum_{i=1}^{n_f} V_i'(k) \frac{\partial T_i(k) - 1}{\partial V'(k)}.$$ (25)

Finally, in two networks, NNM and NPC, the weight values $W(k), V(k)$ are updated through (18), (19), (21), and (23)-(25), so the whole OSNPC runs and controls the general nonlinear system (6) to achieve the desired performance.

IV. CLOSED-LOOP SYSTEM STABILITY AND PASSIVITY

4.1. Closed-loop system stability

Since the dynamical backpropagation algorithms (DBP) described in Section II are employed to tune the network weight values, the learning rates $\eta, \eta_i$ will influence the system performance. Although neural networks have been proven to possess the universal approximation ability, the learning rate in weight tuning has to be carefully selected so as to guarantee that the whole neural system win be stable. If the learning rate is much higher,
it will cause the stability of the neural controller to deteriorate, and if it is much lower, then the optimal weight values will not be obtained. Theorem 4.1 and Theorem 4.2 prove that given \( \eta_a, \eta \) satisfying (26) and (27) respectively, NNM and NPC will be convergent at exponential speed, so the whole closed-loop system will be asymptotically stable.

Define \( \| \cdot \| \) in Theorems 4.1-4.4 as the usual Euclidean norm in \( R^n \), and denote by \( \frac{\partial O(k)}{\partial W(k)} \) the corresponding network gradient information.

**Theorem 4.1.** If the weight values \( W(k) \), \( W'(k) \), \( W''(k) \) for NN (9) and (10) are updated along with (18), (19), and (21), then NN (9) and (10) will be convergent at exponential speed, provided that its learning rate \( \eta_a \) satisfies the following condition:

\[
0 < \eta_a < \frac{2}{\left( \frac{\partial O(k)}{\partial W(k)} \right)^2 (\overline{V}^2 + \lambda)}.
\]

**Proof.** Define the Lyapunov function \( L(k) = \frac{1}{2} e^\top(k) = \frac{1}{2} (y(k) - \overline{y}(k))^2 \). The proof is similar to that of [17].

**Theorem 4.2.** If the weight values \( V(k) \), \( V'(k) \), \( V''(k) \) for NPC (11) and (12) are updated along with (23)-(25), then NPC (11) and (12) will be convergent at exponential speed, provided that its learning rate \( \eta \) satisfies the following condition:

\[
0 < \eta < \frac{2}{\left( \frac{\partial O(k)}{\partial V(k)} \right)^2 (\overline{V}^2 + \lambda)}.
\]

**Proof.** Define the Lyapunov function \( L(k) = J_x(k) = \frac{1}{2} e^\top(k) + \frac{1}{2} \Delta u(k) \), \( \Delta u(k) \) and furthermore, let \( e_x(k+1) = e_x(k) + \Delta e_x(k), \ u_x(k+1) = u_x(k) + \Delta u_x(k) \). Then,

\[
\Delta L(k) = L(k+1) - L(k) = [e_x(k) + \frac{1}{2} \Delta e_x(k)] \Delta e_x(k)
\]

\[
+ \lambda (u_x(k+1) - \Delta u_x(k)) \Delta u_x(k)
\]

\[
= \Delta L_x(k) + \Delta L_x(k).
\]

It follows from (24) that

\[
\Delta e_x(k) = \frac{\partial e_x(k)}{\partial V(k)} \Delta V(k) = - \left( \frac{\partial \overline{V}(k+1)}{\partial u_x(k)} \right)^T \Delta V(k)
\]

\[
= - \overline{V} \left( \frac{\partial O(k)}{\partial V(k)} \right)^T \Delta V(k)
\]

\[
= - \overline{V} \left( \frac{\partial O(k)}{\partial V(k)} \right)^T \Delta V(k).
\]

Therefore, \( \Delta L_x(k) = - \overline{V} e_x(k) \left( \frac{\partial O(k)}{\partial V(k)} \right)^T \Delta V(k) \). Similarly, \( \Delta u(k) = \]

\[
\frac{1}{2} \overline{V}^2 \left( \frac{\partial O(k)}{\partial V(k)} \right)^T \Delta V(k) = \eta_x \left( \frac{\partial O(k)}{\partial V(k)} \right)^T \Delta V(k).
\]

Therefore,

\[
\Delta L(k) = \Delta L_x(k) + \Delta L_x(k).
\]

As a result, to ensure that the NPC converges at exponential speed, it should satisfy \( \Delta L(k) < 0 \), that is,

\[
\eta_x \left( 2 - (\overline{V}^2 + \lambda) \eta_x \left( \frac{\partial O(k)}{\partial V(k)} \right) \right) > 0.
\]

Therefore, \( \eta \) must satisfy condition (27). Finally, it can be verified that the positive function \( \Psi \left( \left( \frac{e_x}{u_x} \right) \right) =
\]

\[
\eta \left( \frac{\partial O(k)}{\partial V(k)} \right) \left( \left( \frac{e_x}{u_x} \right) \right) \text{satisfies the requirement}
\]
Remark 1. We have developed a guideline (Theorems 4.1, 4.2) for selecting an adaptive learning rate for weight value tuning in NNM and NPC which guarantee network convergence. Here, we should note that the stable conditions (26) and (27) are both based on neural networks’ universal approximation ability, and that selecting a suitable network size is not the main concern here (interested readers can refer to [8,9] and [3]). Another work [17] has presented, according to the network size, more detailed adaptive rules for selecting a weight value learning rate, which work also as a frame of reference for selecting a network-size-oriented weight value tuning learning rate.

Remark 2. Our previous work [22] illustrated the effect of the value of $\lambda$, and it indeed is a main feature making our work different from the previous control schemes (for instance, [17]). In this paper, a predictive mechanism has been introduced into the recurrent neural control scheme. If NNM can further predict more than one step, it will theoretically be upgraded to Generalized Predictive Control in the neural control sense. This is still an open topic for future investigation.

Remark 3. Generally, controller stability is discussed based on the assumption that the set value does not change or undergoes only tiny changes (changes slowly). For the sake of simplicity, the set value can also be selected as zero, which translates the controller stability into tuning (for instance, see [11]). Through various simulation studies, it can be found that our OSNPC works smoothly when set values do not change or only change slowly. If the set value changes steeply and transiently to a new stable state, the OSNPC will adapt to it and decrease the error between the system plant response and the set value.

4.2. Neural passive systems

Assume that there exist ideal weights, i.e., that $W$, $V$ are the ideal weight values for NNM and NPC, respectively, and that the weight error values are.

$$\tilde{W}(k) = W - W(k), \quad \tilde{V}(k) = V - V(k).$$

Theorem 4.3. For the nonlinear system (6), the tuning algorithm in (18), (19), and (21) for NPC makes the map from $\Delta W(k)$ to $\tilde{W}(k)$ a dissipative map, provided that the NNM weight learning rate $\eta_t$ satisfies (26).

Proof. Select the Lyapunov function as

$$L(k) = \tilde{W}^T(k)\tilde{W}(k) + \frac{1}{2}\tilde{y}(k) - \tilde{y}(k))^2 \quad (30)$$

$$\Delta L(k) = L(k + 1) - L(k) = \tilde{W}^T(k)\tilde{W}(k + 1) - \tilde{W}^T(k)\tilde{W}(k)$$

$$+ \frac{1}{2}[[\tilde{e}^2(k + 1) - \tilde{e}^2(k)]]$$

$$\equiv \Delta L_1(k) + \Delta L_2(k). \quad (31)$$

The first part of the difference is

$$\Delta L_1(k) = \tilde{W}^T(k + 1)\tilde{W}(k + 1) - \tilde{W}^T(k)\tilde{W}(k)$$

$$= \Delta W^T(k)\Delta W(k) + 2[W(k) - W]^T\Delta W(k). \quad (32)$$

The second part of the difference is $\Delta L_2(k) = \frac{1}{2}[[\tilde{e}^2(k + 1) - \tilde{e}^2(k)]$, similar to [17], and here it results in the following:

$$\Delta L_2(k) = - \frac{1}{2}e^2(k)\eta_t\Delta 2 - \eta_t\left\| \frac{\partial O_t(k)}{\partial W(k)} \right\| \left\| \frac{\partial O_t(k)}{\partial W(k)} \right\|^2. \quad (33)$$

Given that the NNM weight learning rate $\eta_t$ satisfies (26), $\Delta L_t(k) < 0$, we have

$$\Delta L(k) = 2\tilde{W}^T(k)\Delta W(k) + \Delta W^T(k)\Delta W(k)$$

$$- \frac{1}{2}e^2(k)\eta_t\Delta 2 - \eta_t\left\| \frac{\partial O_t(k)}{\partial W(k)} \right\| \left\| \frac{\partial O_t(k)}{\partial W(k)} \right\|^2.$$}

Hence, the total difference is in the form of the energy function (4), as long as (26) is satisfied, and the proof of Theorem 4.3 is thus completed.

Theorem 4.4. For the nonlinear system (6), the tuning algorithm in (23)-(25) for NPC makes the map from $\Delta V(k)$ to $\tilde{V}(k)$ a dissipative map, provided that the NPC weight learning rate $\eta_t$ satisfies condition (27).

Proof. Select the Lyapunov function as

$$L(k) = \tilde{V}^T(k)\tilde{V}(k) + \frac{1}{2}\tilde{y}^2(k) + \frac{1}{2}\tilde{a}^2(k), \quad (34)$$

$$\Delta L(k) = \tilde{V}^T(k + 1)\tilde{V}(k + 1) - \tilde{V}^T(k)\tilde{V}(k)$$

$$+ \frac{1}{2}[[\tilde{e}^2(k + 1) - \tilde{e}^2(k)]]$$

$$\equiv \Delta L_1(k) + \Delta L_2(k). \quad (35)$$

Similar to the proof of Theorem 4.3 and using the proof result in Theorem 4.2, we can directly write the first difference of the Lyapunov function as (35):

$$\Delta L(k) = 2\tilde{V}^T(k)\Delta V(k) + \Delta V^T(k)\Delta V(k)$$
\[- \frac{\partial Q_c(k)}{\partial V(k)} \left[ \mathcal{O}_c - \lambda \hat{\mathcal{O}}(k) \right] \left\{ \eta_i - \frac{1}{2} \left( \mathcal{V}^2 + \lambda \right) \eta_c \left( \frac{\partial Q_c(k)}{\partial V(k)} \right) \right\}, \tag{35}\]

Here, we can see that this is the energy function in the form of (4), provided that the NPC weight learning rate \( \eta_c \) satisfies condition (27).

Passivity is important in a closed-loop system because it guarantees boundness of the system signals and ensures satisfactory performance in the presence of bounded irregular disturbances. Through Theorems 4.3-4.4, we have demonstrated that the weight tuning algorithms make the two networks (NNM and NPC) weight map dissipative, so that additional unknown but bounded disturbances will not destroy the OSNPC’s asymptotic stability.

V. SIMULATION STUDY

All the examples presented in this section were run on the simulation platform MATLAB V5.0.

5.1 Example 1 (NARMAX model)

This example was taken from [2]. The model is identified from a laboratory-scale liquid level system. The system consists of a d.c. water pump feeding a conical flask, which in turn feeds a square tank, giving the system second-order dynamics. The controllable input is the voltage to the pump motor, and the plant output is the height of the water in the conical flask. The aim is for the water height to follow some demand signals, and the identified model is given as

\[
y(k) = 0.9722y(k-1) + 0.3578u(k-1) - 0.1295u(k-2) - 0.3103y(k-1)u(k-1) - 0.04228y^2(k-2) + 0.1663y(k-2)u(k-2) - 0.03259y^2(k-1)y(k-2) - 0.3103y^2(k-2)u(k-2) + 0.3084y(k-1)y(k-2)u(k-2) + 0.1087y(k-2)u(k-1)u(k-2) + 0.2573y(k-2)e(k-1) + 0.2939y^2(k-2)e(k-1) + 0.1087y(k-2)u(k-1)e(k-1). \tag{36}\]

The set values \( y_s(k) \) were switched between 0.25 and -0.25 every 100 iterations in the presence of white noise \( e(k) \) bounded in \([-0.25, 0.25]\), which is in the same order of magnitude as the set values but much larger than those considered by [2].

Both NNM and NPC were 3-6-1. We selected the following initial learning rates: \( \eta_i, \eta_c \) were 0.1, and \( \lambda \) was 2.4. After 1000 iterations for training network weight values, NNM and NPC started closed-loop control, and the system performance along with the control input are shown in Fig. 3. It can be seen that the OSNPC achieved satisfactory performance in the presence of large bounded noise.

5.2 Example 2 (Rigid non-minimum phase model)

This is a rigid nonlinear system with strong non-minimum phase:

\[
y(k) = \frac{y(k-1)}{1 + y^2(k-1)} + u(k-1) + 5u(k-2). \tag{37}\]

The NNM and NPC structures were both selected as 3-7-1, and the initial learning rates were as follows: \( \eta_i, \eta_c \) were 0.1, and \( \lambda \) was 5.5. After 1000 iterations for training network weight values, NNM and NPC started closed-loop control, and the system performance along with the control input are shown in Fig. 4. As a comparison with the OSNPC, Fig. 5 shows the system performance when \( \lambda \) was set to be 0. We can think of the case \( \lambda = 0 \) as similar to the neural control schemes in [17,26], and we can see that, without \( \lambda \), the system performance is very poor.

VI. CONCLUSIONS

As stated at the beginning of this paper, predictive

Fig. 3. (a) System output performance. (b) System control input.
control, which depends on the assumption of linear system parameters, has unavoidable constraints when facing nonlinear dynamics. Therefore, construction of a novel predictive control scheme is difficult but important. Differing from other neural network-based adaptive control algorithms, our NARMAX nonlinear system-oriented one-step neural predictive control (OSNPC) scheme employs two networks (NNM and NPC). Using the DBP algorithm to train the SRNN weights, NNM approximates the system plant and makes a one-step prediction, while NPC generates control input under the cost function of (8). Theorems 4.1-4.4 prove, for over all closed-loop system stability and passivity, the system stable conditions in (26) and (27), which are very practical, at least in simulations. The effectiveness of the developed OSNPC has been verified via exhaustive simulation examples. Despite all, research on neural adaptive predictive control for nonlinear complex systems still calls for more enthusiastic studies.

ACKNOWLEDGEMENT

We wish to thank Prof. Guanrong Chen of Houston University for his valuable advice, and also want to express deep gratitude to all the reviewers for their helpful comments and suggestions.

REFERENCES

11. Clarke, D.W. and C. Mohtadi, “Properties of General-