Identifying Chaotic Systems Via a Wiener-Type Cascade Model

Guanrong Chen, Ying Chen, and Haluk Ogmen

In this article we first show a theory that a Wiener-type cascade dynamical model, in which a simple linear plant is used as the dynamic subsystem and a three-layer feed-forward artificial neural network is employed as the nonlinear static subsystem, can uniformly approximate a continuous trajectory of a general nonlinear dynamical system with arbitrarily high precision on a compact time domain. We then report some successful simulation results, by training the neural network using a model-reference adaptive control method, for identification of continuous-time and discrete-time chaotic systems, including the typical Duffing, Henon, and Lozi systems. This Wiener-type cascade structure is believed to have great potential for chaotic dynamics identification, control and synchronization.

Introduction

It has been recognized that a variety of artificial neural networks are universal uniform approximators for static continuous functions on compact sets [9, 11, 13, 17]. These results have also been partly extended to dynamic systems [10, 32]. The main characteristic of these approaches has been to introduce memory to the system through feedback loops, resulting in the so-called recurrent neural networks. One difficulty in analyzing and synthesizing such networks arises from the fact that memory and nonlinearity are intermixed. In this article we propose a simpler approach, whereby the nonlinearity and memory are in a sense separated, which is the Wiener-type cascade model shown in Fig. 1, consisting of a linear dynamic subsystem cascaded with a static (zero-memory) nonlinear subsystem [27, 32].

There are several advantages in applying a Wiener-type cascade model. First, as can be seen from the figure, the complexity of dynamics is involved essentially in the linear subsystem, while the complexity of nonlinearity is contained only in the static subsystem, where both subsystems have been extensively (yet oftentimes separately) studied, with many useful results and techniques available. Second, its overall system output can be written analytically in terms of the input via kernel functions [32]. Moreover, there have been some reports of success in applying this type of model to nonlinear dynamic systems identification [19].

The approach that we present in this article combines artificial neural networks and traditional control techniques. In the first part of the article, we show that if the static subsystem is a universal uniform approximator for static continuous functions over a compact time domain, and if the dynamic subsystem is controllable, then the Wiener-type cascade model is a universal uniform approximator for general nonlinear dynamical systems, in the sense that it can uniformly approximate any continuous trajectory of a nonlinear dynamical system with arbitrarily high precision on a compact time domain. In the second part of the article, we present several computer simulation results to show how well the proposed system structure can be used to identify chaotic trajectories of some typical nonlinear dynamical systems, in both continuous-time and discrete-time cases.

In recent years, there have been increasing interest and research on controlling and synchronizing chaos (see, for example, the bibliography [2] and surveys [3, 24, 16, 29]). A chaotic system is generally a parametrized nonlinear dynamical system that displays complex behavior such as period-doubling bifurcation, strange attractors, fractals, etc., and can be characterized by several criteria including positive Lyapunov exponent, finite Kolmogorov-Sinai entropy, continuous power spectrum, etc. [6]. In particular, a positive Lyapunov exponent is equivalent to the extreme sensitivity of the nonlinear dynamics to initial conditions, and is defined by the following limit for a discrete-time system (it is similar for continuous-time systems [6]): For the (smooth) discrete-time nonlinear system

$$x_{n+1} = f(x_n), \quad x_n \in \mathbb{R}^n, \quad k = 0, 1, 2, ...$$ (1)

let $J_k(z)$ be the system Jacobian evaluated at $z$, and $\mu (\cdot)$ be the $i$th singular value of the argument matrix, $i = 1, ..., n$. Then the $i$th Lyapunov exponent for the system trajectory starting from $x_0$ is defined to be

$$\lambda_i(x_0) = \lim_{k \to \infty} \frac{1}{k} \ln \left[ \mu \left( J_k(x_0) \right) \mu \left( J_k(x_0) \right) \cdots \mu \left( J_k(x_0) \right) \right], \quad i = 1, ..., n. \quad (2)$$

The authors are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204. This research was supported in part by the Texas Advanced Technology Program under Grant No. 003652023 and the U.S. Army Research Office under Grant No. DAAGH04-94-1-0103.
Identifying, controlling, and synchronizing chaos have proved to be somewhat nontraditional and challenging, with possible new applications in certain interdisciplinary areas such as human brain and heart controls [14, 28].

At this point it is helpful to outline some recent progress (summarized for example in [3, 24, 16, 29]) and highlight some distinctive features of chaos control theory and methodology in contrast to the conventional approaches [5]:

1. The target trajectories in chaos control can be unstable equilibria or unstable periodic orbits (limit cycles), perhaps of high periods. The controller is designed to stabilize these unstable trajectories, or to drive the trajectories of the controlled system to switch from one orbit to another. The inter-orbit switching can be either chaos → order, chaos → chaos, order → chaos, or order → order, depending on the application in interest. Conventional control, on the contrary, does not normally investigate such state-transition problems of a dynamical system, especially not those problems that involve guiding the system trajectory to an unstable state by any means.

2. A chaotic system typically has a dense set of unstable orbits embedded within it, and is extremely sensitive to tiny perturbations to its initial conditions. Such a special feature is useful for chaos control, which is not available in linear or nonlinear systems that normally do not have chaotic characteristics.

3. In many conventional control problems, one usually works within a state space framework; in chaos control, one deals with phase space more often, in which the Poincaré maps, delay-coordinates embedding, parametric variation, entropy reduction, bifurcation monitoring, etc., are some typical but nonconventional tools for study.

4. In conventional control, except for some cases like model-referenced control, the target for tracking is usually a state vector in the state space, which is generally not a state of the given system (otherwise, perhaps no control is needed or can be easily achieved). Moreover, the terminal time for conventional control is usually finite (e.g., the fundamental concept of "controllability" is typically defined by using a finite and fixed terminal time, at least for linear systems [7] and affine-nonlinear systems [23]). However, in chaos control, the target for tracking is not limited to state vectors in the state space; more often than not, it is an unstable periodic orbit of the given system in the phase space. Also, in chaos control, the terminal time is infinite to be meaningful and practical, because most nonlinear dynamical behaviors such as equilibrium states, limit cycles, attractors, and chaos are asymptotic properties.

5. The performance measure in chaos control can be different from the conventional ones, where the former generally uses criteria like Lyapunov exponents, topological dimensions of fractal sets, Kolmogorov-Sinai entropy, power spectra, and ergodicity, etc., whereas the latter usually emphasizes robustness of the system stability and/or control performance, optimality of control energy or time, ability of disturbances rejection, etc.

6. The variety of problems for chaos control include formation of symmetry, self-similarity and patterns, and the birth of limit cycles, fractals and bifurcations, etc., which are not conventional control tasks.

7. In particular, chaos control includes a unique task—anticontrol, required by some special applications such as human brain and heart controls within the context of biomedical engineering, which tries to retain or even enhance chaos for good reasons. Bifurcation monitoring is another example of this kind, where a bifurcation point is expected to be delayed in case it cannot be avoided, so as to significantly extend the operating time (or system parameter range) for a time-critical process such as chemical reaction, power collapse, and compressor control for gas turbine jet engines. Conventional controls almost always try to eliminate unstable behaviors, thereby stabilizing the overall controlled system.

Chaos identification, control, and synchronization can be accomplished via different approaches, including artificial neural networks. This article continues the endeavor in pursuing identification and control of chaos by the neural network technology [12, 8, 26, 4]. The approach taken is to combine neural-network control techniques [21, 15] with an advantageous Wiener-type dynamic model. Depending on whether the chaotic behavior is seen as harmful or useful to the system operations in interest, the design methodology presented here can be used either to suppress (control) or to induce (synchronize) chaos in the underlying system dynamics, which will be a subject for further investigation.

A Theoretical Result

Consider the Wiener-type cascade model shown in Fig. 1, which consists of a linear dynamic subsystem and a static (zero-memory) nonlinear subsystem in cascade.

Suppose that the linear dynamic subsystem (plant) has the following general multi-input multi-output (MIMO) state-space model:

\[
\begin{align*}
\dot{x}(t) &= A(t)x(t) + B(t)u(t), \quad t_0 \leq t < \infty, \\
x(t_0) &= x_0,
\end{align*}
\]

where \( x \in \mathbb{R}^n \), with a given initial state \( x_0 \) and \( u \in \mathbb{R}^{m} \), \( 1 \leq m \leq n \). We assume that the system matrices \( A(t) \) and \( B(t) \) are at least piecewise continuous. Suppose also that the nonlinear subsystem is an artificial neural network (ANN), which is an MIMO, multi-layer, and feed-forward type of network, with the sigmoidal

\[
\sigma(x) = \frac{1}{1 + e^{-x}}
\]

and linear activation functions for hidden and output layer neurons, respectively. This network is shown to be a universal uniform approximator for continuous functions over compact sets [13]. As can be seen from Fig. 1, the output of the ANN is the control input to the plant.

We have the following result.

Theorem 1. Consider the Wiener-type dynamic model shown in Fig. 1. If the linear dynamic plant (1) is completely controllable, and if the multi-layer nonlinear static ANN has a sufficiently large number of layers and neurons, then for any given continu-
**Fig. 2. The structure of the system during its performance phase.**

The structure of the system during its performance phase is illustrated in Fig. 2. The system consists of a reference, controller, linear system, and output. The reference signal is fed into the controller, which then generates the control input to the linear system. The output of the linear system is then fed back to the controller. This structure allows for a feedback mechanism that can be used to control the system's performance.

**Fig. 3. The structure of the system during its learning phase.**

Fig. 3 shows the structure of the system during its learning phase. The system consists of a reference, controller, linear system, and output. The reference signal is fed into the controller, which then generates the control input to the linear system. The output of the linear system is then fed back to the controller. However, in this phase, the error signal is also used to adjust the controller parameters. This allows the system to adapt and learn from its performance over time.

**Simulations**

In this section, we present several computer simulations that show how well the proposed system can identify and synchronize some typical chaotic trajectories.

**Architecture and Training of the System**

The entire structure is illustrated by Fig. 2, where the linear plant is chosen to be completely controllable, for which it is convenient to use the first canonical form in the same dimension as the given system (i.e., the unknown chaotic system to be identified in this study; if even the dimension is unknown, one may gradually increases the dimension during the training process described below). For the static nonlinearity, we use a simple three-layer feed-forward ANN, with the sigmoidal activation function

\[
\sigma(x) = \frac{1}{1 + e^{-x}}
\]

and linear activation functions for hidden and output layer neurons, respectively. Different numbers of neurons, from 10 to 30, have been tested in the simulations to be further described below. It can be seen from Fig. 2 that the inputs of the ANN are the feedback signals from the plant outputs (with time-delays) and the outputs of the ANN are used to control the plant to synchronize the given plant (a chaotic nonlinear system in our simulations).

The controller is trained by a model-reference adaptive control method [20, 11, 30]. The overall learning structure of the system is shown in Fig. 3. The basic idea is to determine the parameters of the neural network using a model-reference adaptive control method using the available outputs of the chaotic system, whose formulation is unknown, so as to obtain a trained neural network that can drive the selected linear dynamic plant to synchronize the unknown dynamics. In training the network for different cases to be discussed below, 1000 training points were used, taken arbitrarily from the same output trajectory of each unknown chaotic system (since the trajectory is unique).

The architecture of the ANN is shown in Fig. 4, which was used for all simulations. Using the notation shown in this figure, we have

\[
u(t) = \sum_j w_j r_j(t)
\]

and

\[r_j(t) = \sigma \left( \sum_k z_{jk} y_k(t) \right),
\]

where the function \(\sigma\) was defined in (4) above. In the training process, we minimize the error

\[
J = \frac{1}{2} \sum (x(t) - y(t))^2,
\]

where \(x(t)\) and \(y(t)\) are the outputs of the plant and the chaotic system, respectively.

The gradient descent algorithm [22] is applied to derive the learning rules for the ANN off-line. After the learning procedure is completed, we keep the trained weights of the ANN as constants and then remove the given chaotic system but use the ANN
Simulation Results

In this subsection, we show several representative simulation results. We first discuss two typical discrete-time chaotic systems, which are easier to describe, and then study a well-known continuous-time chaotic system.

In our simulations, the positive Lyapunov exponent criterion (2) was used to verify that the identified model is not only close to the true model in the sense of least-squares but also has chaotic characteristics with a positive Lyapunov exponent.

Case I: The Henon System. We first simulated the chaotic Henon system with a linear dynamic plant. The discrete-time Henon system is typical in the study of chaotic dynamics and is not too simple in the sense that it is of second order with one delay and two sensitive parameters. The system is described by the following equation:

\[
y(t+1) = -P \cdot y^2(t) + Q \cdot y(t-1) + 1.0, \quad t = 1, 2, \ldots
\]

which, with \( P = 1.4 \) and \( Q = 0.3 \), produces a chaotic strange attractor as shown in Fig. 5.

After a few comparisons, we found that the simple linear plant of the form

\[
x(t+1) = a \cdot x(t) + b \cdot x(t-1), \quad t = 1, 2, \ldots
\]

is sufficient for chaos identification in the present case, where the constants \( a \) and \( b \) should satisfy some stability condition and may also be adjusted for additional consideration in dynamics. We selected \( a \) and \( b \) such that this plant is stable and has the only zero equilibrium point. To force the plant to synchronize the Henon attractor, we added a driving signal \( u(t) \) to it:

\[
\begin{align*}
x(t+1) &= a \cdot x(t) + b \cdot x(t-1) + u(t), \\
\end{align*}
\]

where \( u(t) \) is to be generated by the feed-forward ANN, which is a \( 3 \times N \times 1 \) network with inputs \( y(t), y(t-1), \) and \(-1\), as shown in Fig. 4. The last input is the bias input emulating the threshold of neurons, which enables the system to approximate the zero-to-nonzero mapping in the Henon attractor.

To derive the learning rules, we took the partial derivatives of the objective function \( J \), defined in (7), with respect to the weights \( \theta \) (representing either \( w_i \) or \( z_j \)), and obtain

\[
\frac{\partial J}{\partial \theta} = \Delta(t) \eta(t),
\]

where \( \Delta(t) = x(t) - y(t) \) and \( \eta(t) = \frac{\partial x(t)}{\partial \theta} \) is calculated by the following recursive equation (with zero-initial conditions):

\[
\eta(t) = a \cdot \eta(t-1) + b \cdot \eta(t-2) + \frac{\partial r(t-1)}{\partial \theta},
\]

with

\[
\frac{\partial r(t-1)}{\partial \theta} = \begin{cases} 
w_i(t-1) & \text{if } \theta = w_i, \\
w_j(t-1)(1 - u(t-1))v_i(t-1) & \text{if } \theta = z_j, 
\end{cases}
\]

where \( j = 0, 1, 2 \), and \( i = 0, 1, \ldots, N - 1 \) (\( N \) is the number of the hidden units).

Now, according to the gradient descent algorithm [22], we change the weights by the following learning rule:

\[
\theta \leftarrow \theta - \rho \cdot \Delta(t) \eta(t),
\]

After obtaining the weights, we simulate the chaotic Henon system with these weights and observe its behavior. The results shown in Figs. 5 and 6 were obtained with the initial conditions \( (x_0, y_0) = (0.4, 0.4) \), \( \rho = 0.6 \), and \( N = 30 \) for Case I, and \( (x_0, y_0) = (0.0, 0.0) \), \( \rho = 0.6 \), and \( N = 20 \) for Case II.

Table 1

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
<th>a</th>
<th>b</th>
<th>\rho</th>
<th>N</th>
<th>initial point</th>
<th>LE</th>
<th>M</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>case I</td>
<td>1.4</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.2</td>
<td>30</td>
<td>(0.4, 0.4)</td>
<td>0.461</td>
<td>50,000</td>
<td>1.86%</td>
</tr>
<tr>
<td>case II</td>
<td>1.8</td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>20</td>
<td>(0.0, 0.0)</td>
<td>0.501</td>
<td>300,000</td>
<td>1.81%</td>
</tr>
</tbody>
</table>

Simulation set-ups for the discrete-time systems, where \( P \) and \( Q \) are parameters of the chaotic systems, \( a \) and \( b \) parameters of the dynamic plants, \( N \) the number of hidden units, \( \rho \) the user-specified constant, \( LE \) the Lyapunov exponent, \( M \) the number of learning steps, and \( e \) the relative average root-mean-squared errors between the true and learned chaotic trajectories.
where \( p > 0 \) is the step-size.

The simulation result for this system setting is summarized in Table 1, Case I, where the identified chaotic dynamics is shown in Fig. 6. It should be remarked that in addition to the similarity between the true chaotic trajectory of the Henon system (Fig. 5) and the identified one after the neural network learning (Fig. 6), with only 1.86% relative average root-mean-squared error between the two figures, the resulting trajectory shown in Fig. 6 indeed has a positive Lyapunov exponent 0.461 (versus the true value 0.462).

However, the learning of the ANN is not always as efficient as in the Henon example discussed above, where \( M = 50,000 \) iterations were sufficient for learning (see Table 1). In some cases (even for simpler chaotic systems), the same ANN takes much longer time to arrive at near optimal weights by the same learning procedure. The reason seems to be dependent on the "smoothness" of the chaotic orbits in the phase space, as can be seen from the next example.

**Case II: The Lozi System.** In this simulation, we repeated the process similar to Case I above, but for the chaotic Lozi system

\[
\chi(t+1) = -p \cdot |\chi(t)| + Q \cdot \chi(t-1) + 1,
\]

which has a chaotic strange attractor as shown in Fig. 7 when \( P = 1.8 \) and \( Q = 0.4 \). We again used the same linear plant and the same learning rules as in Case I. For this system, however, it took much longer time (\( M = 300,000 \)) to achieve a reasonably good identification result, as summarized in Table 1, Case II, together with Fig. 8. The Lyapunov exponent of the identified model is 0.501 (versus the true value 0.500).

Next, we show that the proposed technique works as well for continuous-time chaotic systems, which are generally more difficult than discrete-time ones in both identification and control, due to their inherent complex behavior generated by differential rather than difference dynamics.

<table>
<thead>
<tr>
<th>Case</th>
<th>( p )</th>
<th>( p_1 )</th>
<th>( o )</th>
<th>( q )</th>
<th>( p_1 )</th>
<th>( N )</th>
<th>initial point</th>
<th>( LE )</th>
<th>( M )</th>
<th>( e )</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.4</td>
<td>-1.1</td>
<td>1.8</td>
<td>0.620</td>
<td>0.2</td>
<td>10</td>
<td>(0.0, 0.0)</td>
<td>0.0</td>
<td>30,000</td>
<td>0.56%</td>
</tr>
<tr>
<td>II</td>
<td>0.4</td>
<td>-1.1</td>
<td>1.8</td>
<td>1.800</td>
<td>0.2</td>
<td>10</td>
<td>(0.0, 0.0)</td>
<td>1.33</td>
<td>30,000</td>
<td>1.42%</td>
</tr>
</tbody>
</table>

Simulation set-ups for the continuous-time Duffing system, where \( p, p_1, o, \) and \( q \) are parameters of the chaotic systems, \( N \) the number of hidden units, \( p \) the user-specified constant, \( LE \) the Lyapunov exponent, \( M \) the number of learning steps, and \( e \) the relative average root-mean-squared errors between the true and learned chaotic trajectories.
Case III: The Duffing System. The continuous-time chaotic model that we chose for simulation is the typical chaotic Duffing oscillator described by

\[ \frac{\ddot{y} + p\dot{y} + py + y^3}{y} = q\cos(\omega t) \quad (15) \]

The system parameters used in the simulations are summarized in Table 2, and the dynamics of the Duffing chaotic model are displayed in Figs. 9 (for Case I) and 10 (for Case II), respectively.

To choose a linear dynamic plant, after a few comparisons we found that the simple model

\[ \dot{x} + x = 0 \quad (16) \]

is sufficient for the identification purpose. The phase plot of the Duffing system used for learning is the one displayed in Fig. 9. Clearly, it is a limit cycle. It is certainly interesting to see if this stable limit cycle can synchronize chaos under the proposed learning and modeling method.

In order to drive the plant to synchronize the chaotic state of the model, we added a driving signal \( u(t) \) to the plant:

\[ \dot{x} + x = u(t) \quad (17) \]

where \( u \) is the output of the feed-forward neural network controller.

Similarly to the discrete-time cases, we computed the partial derivatives of \( J \) with respect to all the weights \( \theta \). Using the same notation as that in the discrete-time setting, it follows from (17) that

\[ \frac{\partial J}{\partial \theta} + \frac{\partial J}{\partial \theta} = \frac{\partial r}{\partial \theta} \quad (18) \]

This time, we have a differential equation instead of a difference equation. However, except for this, the rest of the procedure is exactly the same as that in the discrete-time setting. By applying the gradient descent algorithm to derive the controller, we obtained the new (identified) dynamics as shown in Figs. 11 (Case I) and 12 (Case II), respectively. The resulting trajectories have fairly small matching errors as shown in Table 2. The Lyapunov exponent of the identified chaotic model is 1.33 (versus the true value 1.32).

It is clear from the simulation results that the ANN is learning very well and enables the controlled plants to capture different chaotic motions.

Conclusions and Discussion

In this article, we have presented a theoretical result showing that by applying a feed-forward artificial neural network in a Wiener-type cascade dynamic model results in a system that can approximate uniformly any continuous trajectory of a general nonlinear dynamic system with arbitrarily high precision on any compact time-domain. We have demonstrated the approximation capability of the proposed system by simulations, where it was used to identify (synchronize) the chaotic continuous-time Duffing oscillator and the discrete-time Henon and Lozi systems. Since our aim in this article is only to illustrate the approximation capability of the proposed system structure, we did not emphasize on the learning efficiency issue. We have noticed that the gradient descent scheme is simple, but in general it has a very low convergence rate. For instance, unless we increase the number of neurons in the network, it takes as many as \( 3 \times 10^5 \) iterations for the ANN to learn to identify the Lozi system (see Case II in Table 1). To remedy this weakness, an alternative algorithm may be used to reduce the learning time [18]. Experimenting with alternative network architectures can certainly improve the learning performance, but will complicate the design in general. In addition, adaptive algorithms that determine the number of hidden units may be used in the future to enable the ANN to better match the complexity of the plant as well as the desired dynamics. Finally, we would like to point out that the approximation discussed in this article uses a uniform measure \( L_{\infty} \)-norm, which is a natural choice based on the uniform approximation property of the ANN. However, this does not guarantee the tracking errors go to zero except in the ideal case where the finite time interval has a dense partition. What we can guarantee is that the resulting dynamics retain the chaos in the sense of positive Lyapunov exponents. A better approximation measure may produce better chaos synchronization but unfortunately there is no theoretical result currently available for the ANN approximation under such measures. This seems to be an interesting research problem that remains as a challenge at the present time.

Appendix (Proof of Theorem 1)

Given a continuous target trajectory \( y(t) \) on \([0,T]\) and an arbitrarily small \( \varepsilon > 0 \), we want to prove that the Wiener-type system

![Fig. 11. The phase plot plant after learning (Duffing: Case I).](image1)

![Fig. 12. The phase plot plant after learning (Duffing: Case II).](image2)
can produce an output \( x(t) \) on \([t_0,T]\) such that after the transient period \([t_0,t_1]\) for an arbitrarily small \( t_1 > t_0 \), we have

\[
|x - y|_{L_{[t_1,T]}} < \varepsilon .
\]  

(19)

Since the linear dynamic plant (1) is completely controllable, for any \( t^* \in (t_0,T] \) the control

\[
u^*(t) = B^*(t) \Phi^*(t^*,t) Q_x \left[ y(t^*) - \Phi^*(t^*,t_0) x(t_0) \right],
\]

(20)

\( t \in (t_1,t_2] \), will drive the output \( x(t) \) to the point \( y(t^*) \), starting from the given initial position \( x(t_0) = x_0 \) and arriving at the target point when \( t = t^* \) (see Theorem 3.3 of [7]), where

\[
\Phi(t^*) = \exp \left\{ \int_{t_0}^{t^*} A(s) ds \right\},
\]

\[
Q_x = \int_{t_0}^{t_1} \Phi(t^*) B^*(t) B(t) \Phi(t) dt > 0.
\]

Now, start with an arbitrarily small \( t^* = t_1 > t_0 \) and its corresponding control function \( u^*(t), t \in [t_0,t_1] \), given as above. We first note that this piece of control function is continuous and hence bounded on \([t_0,t_1]\), and then recall that we have \( x(t_1) = y(t_1) \) from the controllability described above. Continue this control process as the time variable \( t \) moves from \( t_1 \) towards \( T \), in the manner that for any \( t_2 \in (t_1,T) \), we use the control

\[
u(t) = B^*(t) \Phi^*(t_1,t) Q_x \left[ y(t_1) - \Phi^*(t_1,t_1) x(t_1) \right],
\]

(21)

\( t \in (t_1,t_2] \), to guide the output \( x(t) \), from the just-arrived position \( x(t_1) = y(t_1) \), to the new position \( x(t_2) = y(t_2) \), such that the uniform boundedness condition (19) is satisfied on \([t_1,t_2]\). Since both \( y(t) \) and \( x(t) \) are continuous, and hence uniformly continuous, on the compact interval \([t_1,t_2]\), this approximation is always possible by selecting a small enough \( t_2 > t_1 \). Note also that the control function \( u(t) \) is continuous and bounded on the (now larger) interval \([t_0,t_2]\) (see (20) and (21)). Thus, when this continuation of control process is completed at the terminal time \( T \), we obtain a continuous and bounded control function \( u(t) \) that drives the output \( x(t) \) to uniformly approximate \( y(t) \) on the compact interval \([t_1,T] \) within the error bound \( \varepsilon > 0 \), namely

\[
\| x - y \|_{L_{[t_1,T]}} < \varepsilon .
\]

Now, consider the following case: Suppose \( x(t) \) is a solution of (1) with an initial condition \( x(t_0) = x_0 \) and \( u(t) \) is the control function generated by the static nonlinear subsystem, i.e., the ANN, which satisfies the uniform boundedness condition (20). Then, we can prove that the corresponding output of the nonlinear system in closed loop will be uniformly bounded.

Next, let \( \Phi(t) := \exp \left\{ \int_{t_0}^{t} A(s) ds \right\} \)

\[
\Phi(t) = \int_{t_0}^{t} \Phi(s) B(s) \Phi(s) ds > 0.
\]

(22)

and let \( \tilde{x}(t) \) be the output of the linear dynamic plant corresponding to this control \( v \). From the input-output relations of the plant on \([t_1,T]\), that is (see Equation (3.2) of [7]),

\[
x(t) = \Phi(t,t_1) x(t_1) + \int_{t_1}^{t} \Phi(t,\tau) B(\tau) u(\tau) d\tau
\]

and

\[
\tilde{x}(t) = \Phi(t,t_1) x(t_1) + \int_{t_1}^{t} \Phi(t,\tau) B(\tau) v(\tau) d\tau,
\]

we have

\[
\| x - \tilde{x} \|_{L_{[t_1,T]}} \leq \| x - x_0 \|_{L_{[t_1,T]}} + \| x - y \|_{L_{[t_1,T]}}
\]

\[
\leq C \| y - y_0 \|_{L_{[t_1,T]}} + \epsilon
\]

\[
\leq (C + 1) \epsilon,
\]

where

\[
\int_{t_1}^{t} \| \Phi(t,\tau) \|_{L_{[t_1,T]}} \| B(\tau) \|_{L_{[t_1,T]}} d\tau \leq (T - t_1) \| B(\tau) \|_{L_{[t_1,T]} = C < \infty},
\]

in which \( \| \cdot \|_{L_{[t_1,T]}} \) is the spectral norm of a finite-dimensional matrix on \([t_1,T] \). Since \( \epsilon > 0 \) can be arbitrarily small, this completes the proof of the theorem.

References


