Adaptive Neural Replication and Resilient Control Despite Malicious Attacks

Salvatore Giorgi, Firdous Saleheen, Frank Ferrese, and Chang-Hee Won
Department of Electrical and Computer Engineering
Temple University
Philadelphia, Pennsylvania 19122

***Abstract*— In this paper, an Adaptive Neural Control (ANC) architecture is used for system replication and control within a Resilient Control framework. A dynamic model is chosen for our plant and a "maliciously attacked" plant. A Model Reference Adaptive Control (MRAC) architecture is used to replicate and control the plant to match an ideal reference system. At certain time, we replicate a malicious attack by changing plant parameters, injecting false data, or altering sensor data. This attacked plant is then replicated and controlled to match the reference system. Simulations were carried out to show that accurate system replication and resilient control is possible using adaptive neural networks.**

I. Introduction

The problem of maintaining control in the face of malicious attacks is of growing concern. By malicious attacks we mean changes to the plant (even the change of the entire plant model) as well as spoofing types of attacks (i.e., false data injection and sensor data alteration). We propose a control system that adaptively replicates and controls a plant through the use of neural networks and a reference system, which is independent of the plant. Specifically, the Adaptive Neural Control architecture is a Model Reference Adaptive Control (MRAC) system, which was first proposed by D. Hyland and his collaborators [1]–[4]. The architecture is both hierarchical and modular, which gives the system a high level of fault tolerance. It uses two neural networks, one to replicate an unknown plant and another to control the plant. Since this system follows the standard MRAC architecture, as shown in Fig. 1 [5], the plant is controlled to match the closed loop characteristics of an ideal reference system. In this type of control setup, the controller is resilient to the changes in the plant model.

A resilient control system is defined as a system that maintains state awareness and operational normalcy in response to disturbances, including threats of an unexpected and malicious nature [6]. In the ANC system, if the plant were to experience a malicious attack, the neural network would adapt to the newly attacked plant and forget about the previous plant. Thus, the replication alone is not enough to satisfy the resilient definition. The connection to resilient control comes from the MRAC architecture. Here, the first neural network will adapt to replicate the attacked plant, and the second neural network will adapt to match the ideal reference system. Since this reference system is independent of the plant, it remains undisturbed. Hence, even in the face of malicious attacks, the final controlled output matches that of the similarly controlled, but unattacked plant. It is through this combination of adaptive replication and ideal reference system that the ANC system satisfies the requirements of resilient control.

This material is partly supported by the National Science Foundation AIS Grant, ECCS-0969430.

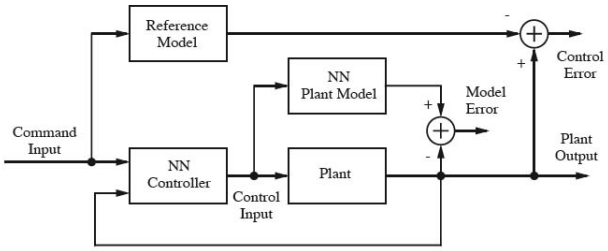


Fig. 1: Model Reference Adaptive Control Architecture

This paper is organized as follows: Section II defines the hierarchy of the ANC system and then discusses each component in detail, as well as convergence results for this type of neural network. Section III describes the model for our original plant, attacked plant, and ideal reference system and presents the simulation results for three different types of attacks: plant parameter changes, false data injection, and sensor data alteration. Section IV discusses the results and finally our conclusions and future directions are given in the last section.

II. Adaptive Neural Control Architecture

A. Hierarchy of Control System

The ANC System hierarchy is shown in Fig. 2. The lowest level contains three devices: a memory unit, an individual neuron, and a synaptic connector. The next level up in the hierarchy consists of dynamic ganglia, or groups of neurons, and Toeplitz synapses, or constrained groups of synaptic connectors. Next are replicator units, or sets of ganglia. Finally, at the top of Fig. 2, is the ANC system, which consists of several replicator units.

B. Memory Units, Neurons and Synaptic Connectors

A memory unit takes a scalar time-series input and produces an *L*-dimensional vector consisting of the current value of the input and the *L-1* delayed values. The training signal

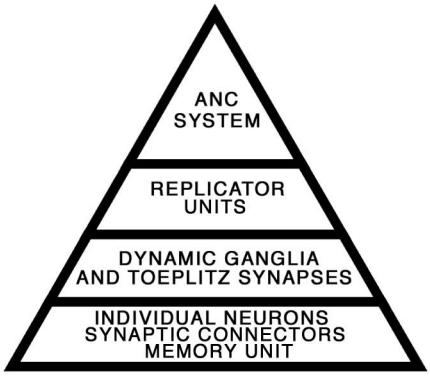


Fig. 2: Hierarchy of Adaptive Neural Control System

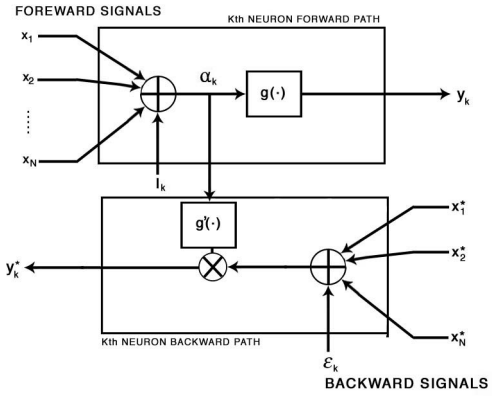


Fig. 3: Explicit Neuron Diagram

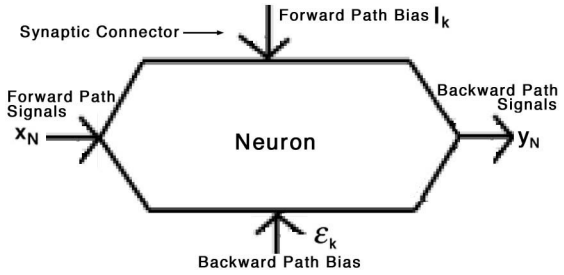


Fig. 4: Simplified Neuron Diagram

ϕ is first sent through the memory unit and results in the output, for $n = 0, 1, 2, ..., N$,

$$\bar{\phi}(n) = \begin{pmatrix} \phi(n) \\ \phi(n-1) \\ \vdots \\ \phi(n-L-1) \end{pmatrix}. \qquad (1)$$

Throughout the paper we will use a bar notation (e.g. $\bar{\lambda}$) to denote a column vector with L past signals augmented as in (1).

Each neuron is defined as in Fig. 3; a dual channel device with a forward signal flow path and a backward signal flow path. Both the forward and backward signal flow paths consist of a sum of a series of input signals and a bias signal. The input signals are usually received from other neurons via synaptic connectors. The neuron sums N forward path inputs $(x_1, x_2, ..., x_N)$ and the forward bias I_k to form α_k which is then operated on by a neural function $g(\cdot)$ (usually a linear or sigmoid function). The output of the neural function forms the output of the forward path of the neuron y_k. The signal α_k is also sent into the backward path of the neuron, where it is operated on by the derivative of the neural function $g'(\cdot)$. In the backward flow path, the N backward inputs $(x_1^*, x_2^*, ..., x_N^*)$ are added to a backward path bias ε_k. This sum is then multiplied by the derivative of the neural function evaluated at α_k, resulting in the output of the backward flow path y_k^*. Fig. 4 is a simplified version of Fig. 3 and shows the N-th input and output of the K-th neuron. The hexagon represents everything within the forward path and backward path blocks in Fig. 3. Only the forward path signals are shown explicitly (all backward path signals are inferred). As a rule, forward path bias signals are shown entering the top of the hexagon and backward path bias signals are shown entering the bottom.

A synaptic connector is a connection between the output of one neuron and the input of a separate neuron. Each synaptic connector has an associated scalar weight $w(n)$ at any instant n, which is multiplied with the output from the neuron. This product is then fed into the input of the next neuron. Defined this way, the neuron itself stays static and only the synaptic weights change. The synaptic connectors are also two-way devices, with a forward and backward path input and a forward and backward path output. Both the forward and backward path inputs are multiplied by the weight and the resulting products form the forward and backward path outputs, respectively. Fig. 5 shows a simplified synaptic

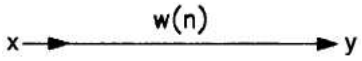


Fig. 5: Simplified Synaptic Connector

connector. Again, only the forward paths are shown, while the backward paths are inferred.

C. Ganglia

Next in the hierarchy are the ganglia and Toeplitz synapses, both seen in Fig. 6. The ganglia, which are collections of neurons, are shown within the dotted lines. In this figure there are four neurons per ganglia, but in general there is no limit to the number of neurons contained within a single ganglia. Toeplitz synapses are constrained groups of synaptic connectors and they are shown within the solid box in Fig. 6. Specifically, Toeplitz synapses constrain connections in the following way: any neuron m places from the top of the ganglia receive signals from all neurons m or more places from the top of the adjacent ganglia. It is important to note that any set of synapses constrained in such a way is considered one Toeplitz synapse. For example, the single

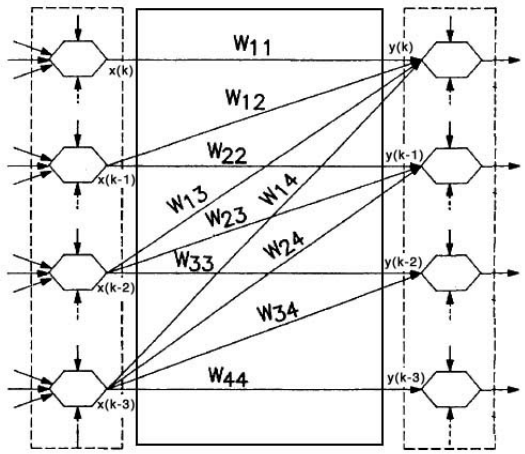


Fig. 6: Ganglia with Toeplitz Synapse

synapses originating from $x(k), x(k-1), ..., x(k-3)$ and terminating at $y(k)$ are all considered one Toeplitz synapse. Sets of Toeplitz synapses are constrained by Toeplitz matrices (2) and the numbering for each synaptic weight uses the following convention: weight w_{ij} is the weight associated with the output from the j-th neuron in the first ganglia going to the i-th neuron in the second ganglia.

These types of interconnection impose a temporal ordering and causality on the neurons. The position of the neurons determine the age of the data, where top level neurons represent current data and lower level neurons represent past data. Since top level neurons do not feed signals into lower level neurons, past data points do not depend on future inputs. Therefore, these types of constrained interconnections are particularly suited for replicating causal systems.

$$W = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1N} \\ 0 & w_{22} & \dots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_{NN} \end{pmatrix} \tag{2}$$

Fig. 7 shows a simplified version of Fig. 6. Ganglia are denoted by the double hexagon and double lined arrows denote sets of Toeplitz synapses.

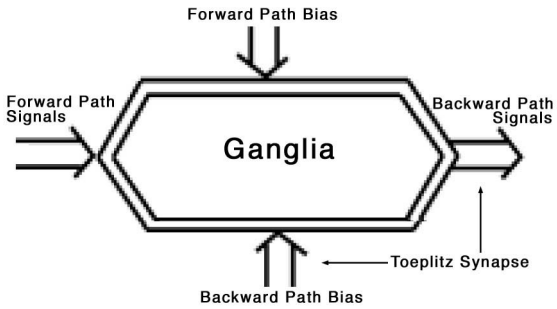


Fig. 7: Simplified Ganglia with Toeplitz Synapse

D. Replicator Unit

The ganglia and Toeplitz synapses are then combined into replicator units, which are used to replicate unknown systems and are shown as the dotted line in Fig. 8. To replicate a general unknown dynamic plant a linear input ganglia, two layers of nonlinear ganglia in the hidden layer, and one linear output ganglia are needed [1]. The number of ganglia in the hidden layer is not restricted, though three are shown in the figure for simplicity. A series of training impulses are sent into both a memory unit and the unknown system. The output from the memory unit is then fed into the forward path bias of the input ganglia. At the same time, the output from the unknown plant is also fed into a memory unit. The output from the plant's memory unit is then compared to the output from the output ganglia. The difference is then fed into the backward path bias of the output ganglia, which is then back propagated through the ganglia driving the weight update laws. The replicator unit is responsible for system replication

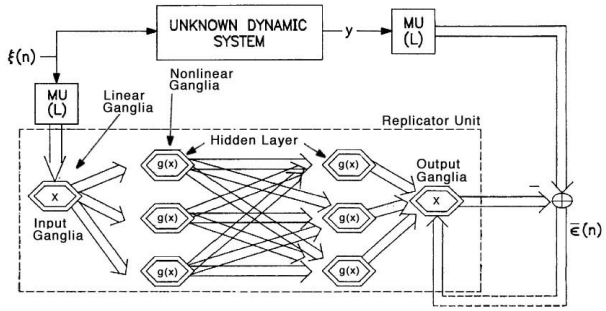


Fig. 8: General Unknown System with Replicator Unit

and is thus a fundamental part of the ANC system. It is important to note the difference between system replication and system identification. By replication we mean to match, as closely as possible, the input and output of an unknown system. System identification not only matches input and output but also requires a mathematical model of the system dynamics.

E. Adaptive Neural Control System

There are two main parts to the Model Reference ANC system in Fig. 9 the closed-loop modeler and the control adaptor. Both parts contain a minimum of two replicator units, W_M and W_C. Note that each replicator unit only shows the input and output layers, the hidden layers are not shown for simplicity. The closed loop modeler uses the training signals $\bar{\xi}$ and the plant sensor outputs to adapt the weights so that the closed-loop system is replicated. When ε_M approaches zero the modeler has replicated the plant within the closed-loop system. The control adaptor uses a training signal, its own output, and the output of an ideal reference system to adjust its weights so that the reference system is replicated. Note that in the closed-loop modeler, W_C is a copy of the current values being updated in the control adaptor and W_M is left unconstrained. Similarly, in the control adaptor, W_M is a copy of the current values adapted

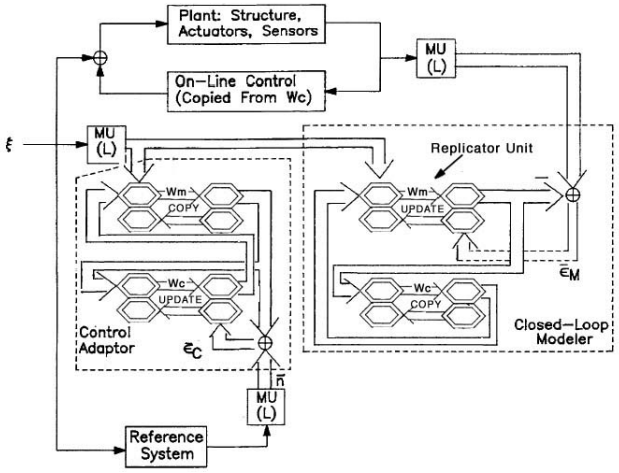


Fig. 9: Adaptive Neural Control System

by the modeler and W_C is left unconstrained. Thus, only one replicator unit is adapting in both the closed-loop modeler and control adaptor. Though both the control adaptor and closed-loop modeler use the same training signal, each has its own error signal, $\bar{\varepsilon}_C$ and $\bar{\varepsilon}_M$, respectively.

F. Adaptive Update Law

The adaptive update law at time $n+1$ depends on the weight $W_k(n)$, forward path signal $\bar{x}_k(n)$, backward path signal $\bar{x}_k^*(n)$, and the global error $\bar{\varepsilon}(n)$, all at time n. $W_k(0)$ must be initialized prior to running the system. For most purposes initializing all matrices to upper diagonal matrices with all nonzero entries equal to 0 or 1 will suffice. The law for the k-th Toeplitz synapse at time $n+1$, for $n = 0, 1, ..., N$ is as follows

$$W_k(n+1) = W_k(n) + \mu_k(n) U_0 * (\bar{x}_k^*(n) \bar{x}_k^T(n)) \tag{3}$$

where

$$\mu_k(n) = \beta_k F(n), \tag{4}$$

$$U_0 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \in \mathbb{R}^{m \times m}, \tag{5}$$

and $W_k \in \mathbb{R}^{m \times m}$ with m being the number of neurons in the ganglia connected via the Toeplitz synapses. The symbol * denotes the Hadamard product, which is term by term matrix multiplication. Additionally,

$$F(n) = \frac{P(n)}{A(n)}, \tag{6}$$

$$P(n) = L(\frac{1}{2}\|\bar{\varepsilon}(n)\|^2 - J), \tag{7}$$

$$L(\sigma) = \begin{cases} \sigma : \sigma > 0 \\ 0 : \sigma \leq 0 \end{cases}, \tag{8}$$

and

$$A(n) = \sum_{\omega} \|\bar{x}_k^*(n)\|^2 \|\bar{x}_k(n)\|^2. \tag{9}$$

In the above equations, P is referred to as the Performance Function, $J \in \mathbb{R}$ is the desired mean square error level, $A(n)$ is the neural activity level, and ω is the set of all Toeplitz synapses connecting the two ganglia in question. The matrix (5) constrains the entire equation to Toeplitz form. The global error $\bar{\varepsilon}(n)$ is the error of the system being replicated. For example, in Fig. 9, the closed-loop modeler replicated the unknown plant within a closed-loop and uses the difference between the replicator unit and the closed-loop system output. So for this part of the system the global error is considered $\bar{\varepsilon}_m$.

In (7) and (9), $\|\cdot\|$ denotes the Frobenius norm. As discussed above, the double lined arrow denotes a set of Toeplitz synapses constrained by the weight matrix W_k. Therefore the summation in (9) is over all Toeplitz synapses within the set constrained by W_k.

In (4), $\mu_k(n)$ is a time varying adaptive speed. This time varying adaptive speed, along with the Toeplitz synapses, is one of the defining features of the ANC architecture. β_k is the learning rate constant. This constant is chosen before adaptation begins and is programmed into the system.

Convergence results for a Toeplitz network with adaption described by (3) through (9) can be found in [1]. First, we note that $\frac{1}{2}\|\bar{\varepsilon}(n)\|^2$ is a function of the training histroy and of all of the weight matrices W_k for $k = 1, ..., N_a$, where N_a is the number of neural arrays, at time n. Therefore, we can write

$$\Lambda(W(n)) = \frac{1}{2}\|\bar{\varepsilon}(n)\|^2. \tag{10}$$

We say that (10) is bounded by a homogenous function of degree M if and only if

$$0 < \frac{\Lambda - \Lambda_0}{\frac{1}{M}(W - W_0)^T \frac{\partial \Lambda}{\partial W}} < 1, \tag{11}$$

for $\forall (W - W_0) \neq 0_{N_s}$ and $W_0 \in \mathbb{R}^{N_s}$, where N_s is the total number of independent, nonzero, synaptic weights, and Λ has a single global minimum Λ_0 at $W = W_0$. In [1] it is proven that if we assume that all vectors $\bar{x}_k^*(n)$ and $\bar{x}_k(n)$ in (3) are uniquely determined, $\Lambda_0 \leq J$, $\Lambda(n)$ is bounded by a homogeneous function of degree M, and $\beta_k < 2M$, that

$$\lim_{n \to \infty} \Lambda(n) \leq J. \tag{12}$$

III. Simulations

The control architecture shown in Fig. 9 was built in Matlab and Simulink. To simulate the malicious attack, we change the plant model at a specified time during the simulation. This time can be chosen arbitrarily, but we assume the controller has been running for some time, and therefore the closed-loop output already matches that of the reference system. Thus the attack time is chosen so that the plant is already being controlled. For example 1 and 2 we choose $t = 5$ as the attack time and for example 3 and 4 we choose $t = 3$.

Four different learning rate constants are used in the simulations: β_L (for synapses connecting linear neurons to nonlinear neurons), β_N (for synapses between nonlinear neurons), β_M (for synapses in the closed-loop modeler), and β_C (for synapses in the control adaptor). For any single synaptic connector we use a product of two learning rate constants, depending on the location of the neuron and the neural function being used. For example, for a synapse connecting two nonlinear neuron in the control adaptor the learning rate constant in (4) is $\beta_k = \beta_N \beta_C$.

We arbitrarily set the number of outputs from the memory unit and the number of neurons in each ganglia equal to 10. As an input function we used a unit step with a step time of 0 to train the neural network. Additionally, the sample time was set to 0.001 seconds, J to 1.0×10^{-8}, the weight matrices connecting the linear ganglia to the nonlinear ganglia were initialized to upper diagonal matrices with all nonzero entries equal to 1, and the weight matrices connecting the nonlinear ganglia to other nonlinear ganglia to upper triangular matrices with all nonzero entries equal to 1.0×10^{-6}. The neural function for the linear neurons was given as $g(x) = x$ (Fig. 8, input and output ganglia). The neural function for the nonlinear neurons was a tansig function

$$g(x) = \frac{2}{1+e^{-2x}} - 1. \tag{13}$$

The sigmoid function is used because, according to the Universal Approximation Theorem, or Cybenko Theorem [7], a neural network with a finite number of neurons, each containing a sigmoid function, can approximate any continuous function. Specifically, using a sigmoid function as the neural function, any continuous, real valued function in the space of continuous functions with support in an m-dimensional unit hypercube can be uniformly approximated within an arbitrary tolerance .

A. Example 1: Simple Plant Model Changes

In order to simulate a malicious attack, we consider two plants; one which is the original undisturbed plant and another that represents the plant after the attack. Because of the nature of the ANC system, we consider both plants "unknown". For the undisturbed plant, we consider the following system taken from [8]

$$\dot{x}(t) = -f[x(t)] + u(t) \tag{14}$$

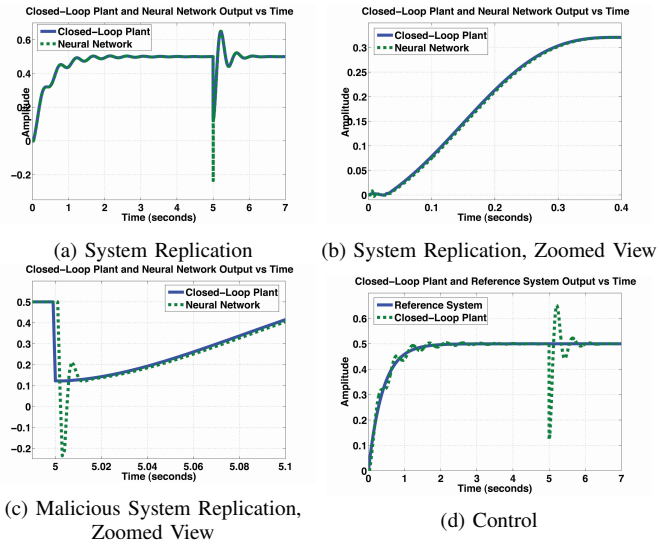
$$f[x(t)] = 2x(t) + 0.8x^3(t), \tag{15}$$

where $u(t)$ is the input to the plant, with the additional assumption

$$y(t) = x(t). \tag{16}$$

For the maliciously attacked plant, we consider the following simple linear model

$$T(s) = \frac{1}{s+9}. \tag{17}$$



(a) System Replication

(b) System Replication, Zoomed View

(c) Malicious System Replication, Zoomed View

(d) Control

Fig. 10: Simulations for Attacked Linear Model

To follow the control architecture shown in Fig. 9 we need an additional plant model for the Reference System. Again, we follow [8] and use the following system

$$\dot{x}(t) = -2.5x(t) + 2.5u(t) \tag{18}$$

and assume

$$y(t) = x(t). \tag{19}$$

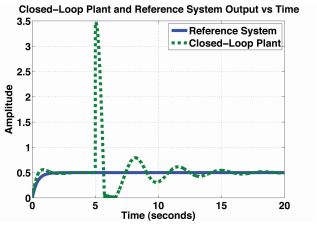
We use the following values for the learning rate constants: $\beta_M = 0.5$, $\beta_C = 0.5$, $\beta_N = 5 \times 10^{-5}$, and $\beta_L = 0.5$. Fig. 10a shows the results of the system replication simulations. Here we see the output from the closed loop plant and the output of the closed loop modeler replicator unit. The figure shows the neural network output matching that of the closed-loop plant almost identically at all times, even when the system changes at $t = 5$ seconds. Fig. 10b is a zoomed in view of Fig. 10a from $t = 0$ to $t = 0.4$. Here we see that the closed-loop replicator matches the closed-loop plant output before 0.1 seconds. Fig. 10c shows the replication of the malicious attack, from $t = 4.9$ to $t = 5.1$. From this figure we can see that the new plant is replicated after approximately 0.003 seconds.

Fig. 10d shows both the original plant and the maliciously attacked plant being controlled to match the reference system. The closed-loop output of the first plant matches the reference system output before $t = 1$, with slight oscillations. At 5 seconds we see the plant output drastically change as a result of the attack. The attacked plant's output is controlled to match the reference system's output approximately 1.5 seconds after the attack.

B. Example 2: Nonlinear Plant Model Changes

For this example our original plant model is decribed by (14), (15), and (16), and our reference model is described by (18) and (19). The attacked plant is described by the following nonlinear model [8],

$$\dot{x}(t) = -f[x(t), u(t)] + u(t) \tag{20}$$



(a) Attacked Nonlinear Model

Fig. 11: Simulations for Attacked Nonlinear Model

$$f[x(t),u(t)] = \frac{y(t)}{1+y^2(t)} + \tanh[u(t)], \tag{21}$$

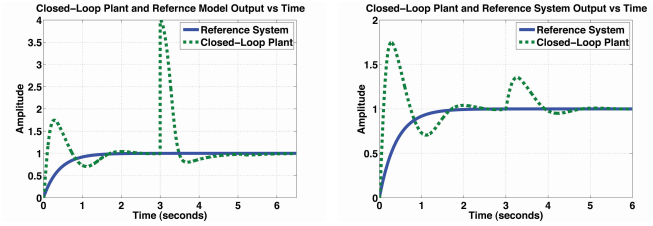
with the additional assumption

$$y(t) = x(t). \tag{22}$$

Fig. 11a shows the simulation results when our model changes to the nonlinear plant described by (20), (21), and (22). All simulation variables for this and all subsequent simulations are exactly the same as in the previous example, except we change the learning rate constant values to $\beta_M = 1$, $\beta_C = 0.1$, $\beta_N = 1 \times 10^{-6}$, and $\beta_L = 0.1$. Here we see the closed-loop plant output matching that of the reference system after approximately 11 seconds after the attack occurs.

C. Example 3: Sensor Data Alteration

To simulate sensor data alteration we use the plant described by (14), (15), and (16) and the reference system described by (18) and (19). At $t = 3$ we add additional data to the output of the plant. The additional data is another step signal, whose step time is 0 and whose final value is 3. From Fig. 12a below we see that the ANC system is able to match the reference systems output after approximately 3 seconds.



(a) Sensor Alteration (b) False Data Injection

Fig. 12: Simulations for Spoofing Attacks

D. Example 4: False Data Injection

We follow the same procedure detailed above in the sensor data alteration simulations to simulate false data injection, except that instead of adding additional data to the output of the plant, we inject false data into the input of the plant. The false data is a step signal whose step time is 0 and whose final value is 3. Fig. 12b shows the results of this simulation. As seen in the figure, the closed-loop plant output matches that of the reference system approximately 2 seconds after the false data is injected.

IV. Discussion

Our system parameters (learning rate constants) were chosen, through repeated simulations, to minimize the control and replication times as well as guarantee convergence to the reference system output. Increasing the learning rate constants speeds up replication and control. Setting the learning rate constants too high resulted in considerable oscillations in our closed-loop response and often in little to no control. Simulations show that any learning rate constant set greater than one resulted in extreme oscillations and absolutely no replication or control. Choosing small constant values resulted in extremely long replication and control times but minimized any overshoot or oscillations. These constants were also highly sensitive to our original plant, attacked plant, and referent system models. Despite the sensitivity to the choice in parameter values, simulations carried out for various linear and nonlinear plant and reference models gave similar results with the above parameter values. In particular, setting $\beta_M \leq 1$, $\beta_C \leq \frac{\beta_M}{5}$, $\beta_L \leq 0.1$, and $\beta_N < 1 \times 10^{-4}$ resulted in fast replication and control times with minimal overshoot.

V. Conclusions

In this paper we studied the possibility of accurate system replication and control with an adaptive neural network within a resilient control framework. From the simulation results we conclude that the system replicator is able to quickly and accurately replicate the original plant and the "attacked" plant. Despite changes to the plant model, sensor alteration, and false data injection, our final closed loop system output matches that of our ideal reference system after only a few seconds, with worst case time for control being approximately 12 seconds. Therefore, the Adaptive Neural Control system described above is able to maintain control when faced with multiple types of malicious attacks.

References

[1] D.C. Hyland, "Connectionist algorithms for identification and control: System structure and convergence analysis," in *American Institute of Aeronautics and Astronautics 35th Aerospace Sciences Meeting*, Jan. 1997.

[2] D.C. Hyland, "Multiprocessor system and method for identification and adaptive control of dynamic systems," U.S. Patent No. 5796920, Aug 1998.

[3] D.C. Hyland, "Neural network architecture for on-line system identification and adaptively optimized control," in *Proceedings of the 30th Conference on Decision and Control*, 1991, vol. 3, pp. 2552 – 2557.

[4] L.D. Davis and D.C. Hyland, "Adaptive neural control for the astrex testbed," in *Proceedings of the 1997 American Control Conference*, 1997, vol. 3, pp. 1794 – 1798.

[5] M.T. Hagan and H.B. Demuth, "Neural networks for control," in *American Control Conference, 1999. Proceedings of the 1999*, 1999, vol. 3, pp. 1642 –1656 vol.3.

[6] C.G. Rieger, "Notional examples and benchmark aspects of a resilient control system," in *Resilient Control Systems (ISRCS), 2010 3rd International Symposium on*, aug. 2010, pp. 64 –71.

[7] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, pp. 303–314, 1989, 10.1007/BF02551274.

[8] H.D. Patino and D. Liu, "Neural network-based model reference adaptive control system," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 30, no. 1, pp. 198 –204, feb 2000.