JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 19, No. 4, July–August 1996

Adaptive-Critic-Based Neural Networks for Aircraft Optimal Control

S. N. Balakrishnan* and Victor Biega†
University of Missouri–Rolla, Rolla, Missouri 65401-0249

A dual neural network architecture for the solution of aircraft control problems is presented. The neural network structure, consisting of an action network and a critic network, is used to approximately solve the dynamic programming equations associated with optimal control with a high degree of accuracy. Numerical results from applying this methodology to optimally control the longitudinal dynamics of an aircraft are presented. The novelty in this synthesis of the optimal controller network is that it needs no external training inputs; it needs no a priori knowledge of the form of control. Numerical experiments with neural-network-based control as well as other pointwise optimal control techniques are presented. These results show that this network architecture yields optimal control over the entire range of training. In other words, the neural network can function as an autopilot. A scalar problem is also used in this study for easier illustration of the solution development.

I. Introduction

OPTIMIZATION is a primary concern in most real world processes. Two-point boundary value problem (TPBVP) methods[1] provide near-exact solutions, but must be solved for each set of initial conditions. This means determining a separate solution for each possible set of initial conditions for a given system. Dynamic programming[1,2] is a method of determining optimal control for a family of initial conditions. However, the usual method of solution of dynamic programming equations becomes very difficult to solve for in higher dimensions and for nonlinear systems. These methods of solution for control do not usually yield a feedback form of control in terms of states either.

Other methods of solution also have their advantages and disadvantages. Neighboring optimal control is beneficial in that the solution of a single TPBVP allows an approximate solution over a range of initial conditions. The disadvantage is that approximation methods such as neighboring optimal control can fail at a distance from the original TPBVP solution.

Several authors[3–5] have used neural networks to solve control problems. This paper will review a few examples from the literature. Ismail et al.[6] use neural networks to optimally control a variable reluctance motor. The method they use, however, first requires that the optimal control path be determined mathematically for a variety of initial conditions. A neural network is then trained to model these trajectories. In essence, this method[6] generalizes for various initial conditions according to its original training set. A weakness of this method is that the optimal trajectories must be determined mathematically prior to training the neural networks. In neural network literature, this technique is known as cloning.

Nikolauo and Hanagandi[7] use neural networks to control nonlinear systems. In their method, the networks are used to determine a system model, and the model is subjected to linearization. The control is then determined from this linearization. This method[7] again uses traditional mathematics to solve for the optimal control laws, and uses the neural network solely to model the system.

Yamada and Yabuta[8] have developed a method of using neural networks to optimally control nonlinear systems. Their method is concerned with a single initial condition. A neural network is used to model the optimal control, and the system cost function is calculated using this control law. After the cost function is determined for the entire run, the control network is adapted using the gradient of the cost function with respect to the weights of the control network. Some limitations of this method are that it can only determine the control law for a single initial condition, it requires an explicit look-ahead, and, therefore, is limited to a fixed and finite horizon.

Similar to these, there is a multitude of neurocontrollers in the published literature.[4] Almost all of them fall within four categories: 1) supervised control, 2) direct inverse control, 3) neural adaptive control, and 4) backpropagation through time. A fifth and rarely studied class of controller has the most interesting structure. It is called an adaptive critic architecture. This neural network structure falls under the class that is known as learning control. Fu[9] gives one of the first formal discussions of the use of learning in control systems. Fu describes a learning controller in terms of a two-component combination. These components are the controller and the trainer. Together, the controller and trainer perform tasks of pattern recognition, control parameter searching, and system performance improvement over time. Sklansky's[10] general form for a learning controller contains a hierarchy of three feedback loops, a simple feedback loop, an adaptive loop, and a learning loop. Sklansky's learning controller can also be thought of as a two-component system, one for pattern recognition and control parameter selection and the other to work as a teacher, which observes system performance and adjusts category boundaries in the controller.

Nikolic and Fu[11] describe an on-line learning algorithm for a learning controller. Nikolic's learning control contains a student and an unreliable teacher. The student acts as the controller. The unreliable teacher uses a reinforcement learning strategy and observations from the environment to modify controller parameters. The Widrow et al.[12] method also uses learning with a critic. The critic in this case provides feedback to a neural control element based on a comparison between short- and long-term performance. Parameters in the control element are then modified using positive or negative reinforcement. Barto et al.[13] and Anderson[14] describe reinforcement learning control with an adaptive critic. The control architecture consists of two neuronlike elements, an associated search element (ASE) and an adaptive critic element (ACE). The ASE is responsible for associating proper control actions with regions in the state space. The ACE assists also learns to improve in its job as a teacher based on less frequent reinforcement from an external source.

A major difference between all of these existing works and this study is that the networks are synthesized offline but they work as feedback controllers. (A reviewer has pointed out that a few studies have used offline trained control for online though they are not based on the critic design used in the study.[15]) Also, an aircraft cannot be trained through crashing as in the case of an inverted pendulum or a robot studied by Barto et al.[13] and others. The reasons for

Received July 31, 1995; revision received Dec. 12, 1995; accepted for publication Dec. 15, 1995.

*Associate Professor, Department of Mechanical and Aerospace Engineering and Engineering Mechanics, 219 ME Annex. Associate Fellow AIAA.
†Graduate Student, Department of Mechanical and Aerospace Engineering and Engineering Mechanics.

choosing this structure for formulating the aircraft optimal control problems are as follows: 1) This structure obtains an optimal controller through solving dynamic programming equations. 2) This approach (see Fig. 1) has a supervisor (critic) that critiques the outputs of the controller network and a neural network controller. Therefore, this approach has a built-in fault tolerance. 3) This approach needs no external training as in other forms of neurocontrollers. 4) This is not an open-loop optimal controller but a feedback controller. 5) It preserves the same structure regardless of the problem (linear or nonlinear).

The method proposed in this study determines an optimal control law for a system by successively adapting two networks, an action and a critic network. It determines the control law for an entire range of initial conditions. When the networks converge, the dynamic programming solutions are imbedded in the networks. The general problem is formulated in the next section. It is followed by an illustrative scalar example and results from an aircraft optimal control problem. The conclusions are summarized at the end.

II. Solution Method Development

A. Neural Network Background

Neural networks, or, in the case of this study, multilayer perceptrons (MLPs), are known for their ability to model any mapping from input to output given a correctly chosen network structure. They are also able to adapt to new sets of input output pairs. This makes them ideal in adapting to an optimal control policy.

For the problems in this study, we will be using MLPs of the form shown in Fig. 2. In this case, the activation functions are

$$f_1 = \frac{2}{1+e^{(-\text{net})}} - 1 \qquad f_2 = \frac{1}{1+e^{(-\text{net})}} \qquad f_3 = \text{net} \tag{1}$$

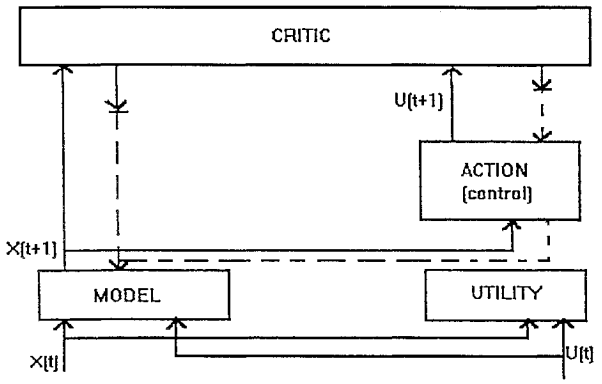

Fig. 1 Adaptive critic architecture.

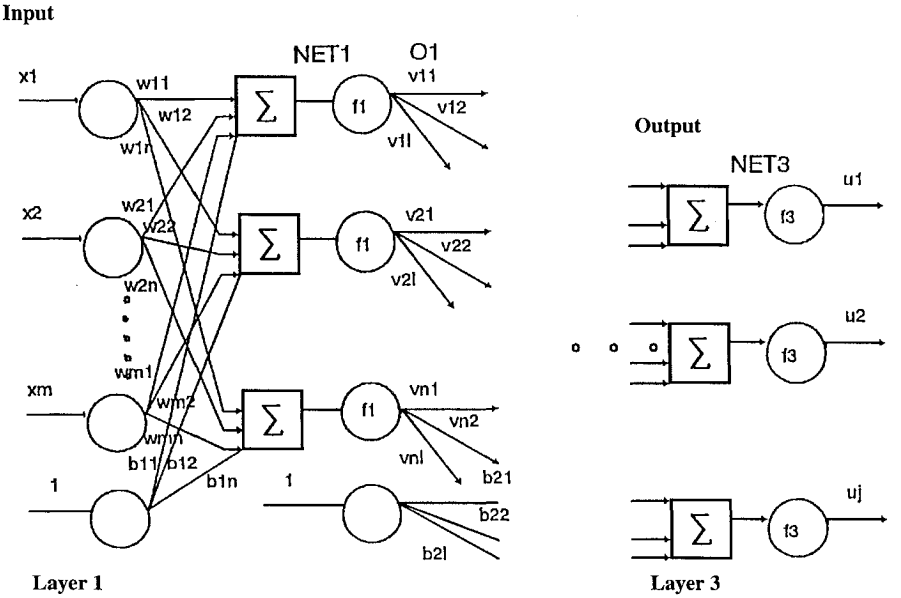

Fig. 2 Standard multilayer perceptron.

Assuming that there is some function to be minimized, it is then possible to adjust the weights of the MLP to model the appropriate mapping using a standard gradient descent algorithm.[16] If suitable, then recurrent networks such as Hopfield can be used for different applications.[4] The solution process is not a function of the neural network paradigm. It should, however, be noted that the network used should be able to provide gradients of the output with respect to input.

B. Problem Formulation

In this formulation, problems of the form (infinite dimensional)

$$J = \int_0^{t_f} \psi[x(\tau), u(\tau)]\,d\tau \tag{2}$$

$$\dot{x} = f(x, u) \tag{3}$$

$$t_f \equiv \text{given} \qquad x_0 \equiv \text{given} \tag{4}$$

can be considered. The first step taken is to discretize them into the form

$$J = \sum_{k=0}^{N-1} \psi_D[x(k), u(k)] \tag{5}$$

$$x(k+1) = f_D[x(k), u(k)] \tag{6}$$

$$N \equiv \text{large or given} \qquad x(0) \equiv \text{given} \tag{7}$$

The neural-network-based method in this study has advantages over the previous methods in that solutions are found over any user specified range of x, and these solutions are then available for the entire span of x.

C. Dynamic Programming Background

Assuming the type of cost function in Eq. (5), one can write that

$$J[x(t)] = U\{x(t), u[x(t)]\} + \langle J[x(t+1)]\rangle \tag{8}$$

Here, $J[x(t)]$ is the cost associated with going from time t to the funal time. $U\{x(t), u[x(t)]\}$ is the utility, which is the cost from going from time t to time $t+1$. Finally, $\langle J[x(t+1)]\rangle$ is assumed to be the minimum cost associated with going from time $t+1$ to the final time.

If both sides of the equations are partially differentiated with respect to x and we define

$$\lambda[x(t)] \equiv \frac{\delta J[x(t)]}{\delta x(t)} \tag{9}$$

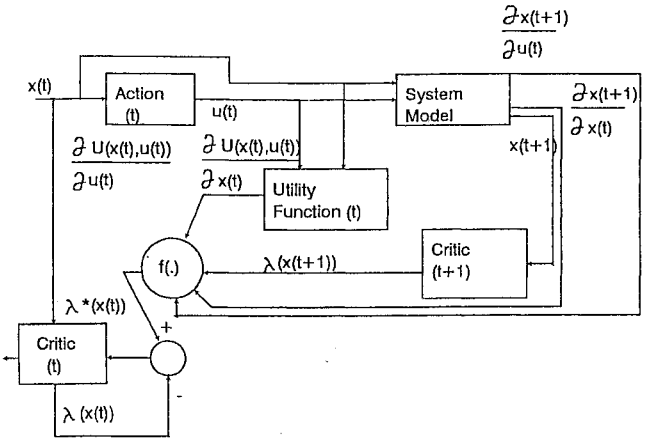

Fig. 3 Critic network training diagram.

then

$$\lambda[x(t)] = \frac{\delta U[x(t),u(t)]}{\delta x(t)} + \frac{\delta U[x(t),u(t)]}{\delta u(t)}\frac{\delta u[x(t)]}{\delta x(t)} + \left\langle \lambda[x(t+1)]\frac{\delta x(t+1)}{\delta x(t)} \right\rangle + \left\langle \lambda[x(t+1)]\frac{\delta x(t+1)}{\delta u(t)}\frac{\delta u[x(t)]}{\delta x(t)} \right\rangle \tag{10}$$

From this it can be seen that if $\langle\lambda[x(t+1)]\rangle$, $U[x(t), u(t)]$ and the system model derivatives are known, then $\lambda[x(t)]$ can be found.

Next, the optimality equation is defined as

$$\frac{\delta J[x(t)]}{\delta u(t)} = 0 \tag{11}$$

In dynamic programming we use these equations to aid in solving an infinite horizon control policy or to determine the control policy for a finite horizon problem.

D. Network Development or Training Methods (Approximation Techniques)

As mentioned earlier, this study uses Eqs. (10) and (11) to determine the optimal control policy. The basic training takes place in two stages: the training of the action network (the network modeling $u[x(t)]$ and the training of the critic network (the network modeling, or approximating $\lambda[x(t)]$). Both networks are initially assumed to be feedforward MLPs.

To train the action network for time step t, first $x(t)$ is randomized and the action network outputs $u(t)$. The system model is then used to find $x(t+1)$ and $[\delta x(t+1)]/[\delta u(t)]$. Next, the critic from $t+1$ is used to find $\lambda[x(t+1)]$. This information is used to update the action network. This process is continued until a predetermined level of convergence is reached.

To train the critic network for the time step t, the $x(t)$ is randomized and the output of the critic $\lambda[x(t)]$ is found. The action network from step t calculates $u(t)$ and $[\delta u(t)]/[\delta x(t)]$. The model is then used to find $[\delta x(t+1)]/[\delta x(t)]$, $[\delta x(t+1)]/[\delta u(t)]$, and $x(t+1)$. The critic from step $t+1$ is then used to find $\lambda[x(t+1)]$. After this, Eq. (10) is used to find $\lambda^*[x(t)]$, the target value for the critic. This process is continued until a predetermined level of convergence is reached.

A flowchart of the critic network updates and related calculations is presented in Fig. 3.

III. Applications

In this section of the study, two specific examples will be dealt with. The first of these is an infinite horizon one-dimensional linear problem. After this motivating example, a four-dimensional aircraft control problem is presented.

A. Infinite Time One-Dimensional Linear Application

The first application deals with a problem of the form

$$x(t+1) = x(t) + 2u(t) \tag{12}$$

and a cost function of the form

$$J = \sum_{t=0}^{\infty}[x^2(t) + u^2(t)] \tag{13}$$

As a first step in the solution, a stabilizing controller is defined. In the case of this problem, the initial control is defined as

$$u(t) = -0.2x(t) \tag{14}$$

Alternately, the control can be initiated by a network with random weights. Next, a neural network is designed and the initial weights of this network are randomized. For this problem, the network has three layers and each of the hidden layers possesses three neurons. This network functions as the adaptive critic.

It can be observed that for the infinite horizon problem the cost associated with state $x(t)$ at time t should be equal to the cost associated with state $x(t)$ at time $t+1$; therefore, a single critic can be used to calculate both $\lambda[x(t)]$ and $\lambda[x(t+1)]$. Thus, we define

$$U[u(t), x(t)] = x^2(t) + u^2(t) \tag{15}$$

Note that we can obtain the derivatives of the utility function from Eq. (15). This, in combination with the critic outputs and the system model derivatives, allows the use of Eq. (10) to determine the target value for the critic $\lambda^*[x(t)]$. This target value is calculated for random values of $x(t)$ until the critic network converges. For the control derivatives in Eq. (10), the functional form in Eq. (14) is used.

After the critic converges, a new neural network is initialized to act as the action network. For this problem a neural network with two hidden layers and three neurons per layer was chosen. The action network is then trained using a gradient descent algorithm with outputs from the converged critic network, which are used in solving Eq. (11) for control.

After the action network converges, the critic is again trained using the outputs of the new action network. (Note that the weights of the critic are not randomized. Instead, the weights from the previous critic are used as the initial weights.) This process is repeated until both networks converge. At this point, the outputs of the action network produce optimal control. In other words, the necessary conditions for optimal control as stated by Eqs. (10) and (11) are simultaneously imbedded in the critic and action networks and, therefore, the outputs of the controller network are optimal.

The evolution of the control law (or the action network) is presented in Fig. 4. The solid line represents the assumed control used in the design of the first critic. The corresponding critic is presented by a dotted line in Fig. 5. Other curves in Fig. 5 represent the evolution of the critic. In three iterations of the action and critic networks, both networks have shown convergence, as can be observed from Figs. 4 and 5. At this point, the action network is expected to output control, which is optimal. To check the optimality of the output, the optimal control law obtained through a Riccati solution formulation[1] is also shown in Fig. 4. It is observed that at almost all of the points considered, the neural-network-based control is nearly identical with the Riccati solution. It should be observed that the number of iterations for convergence is problem dependent.

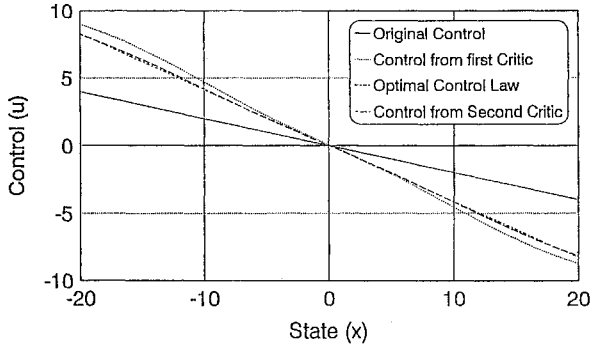

Fig. 4 Control law for infinite horizon problem.

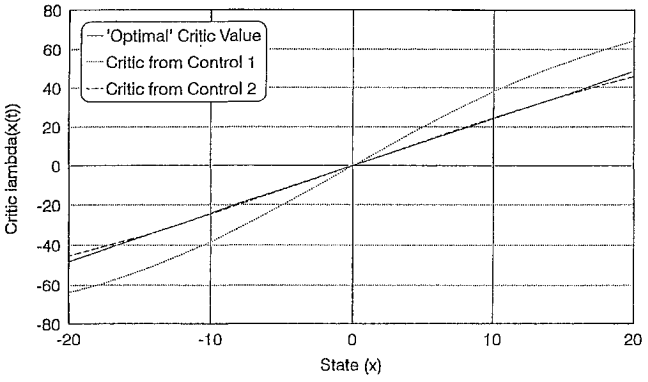

Fig. 5 Critic for infinite problem.

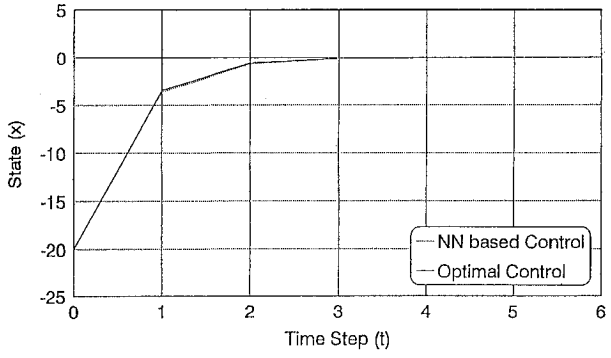

Fig. 6 Controlled system with initial condition $x(0) = -20$.

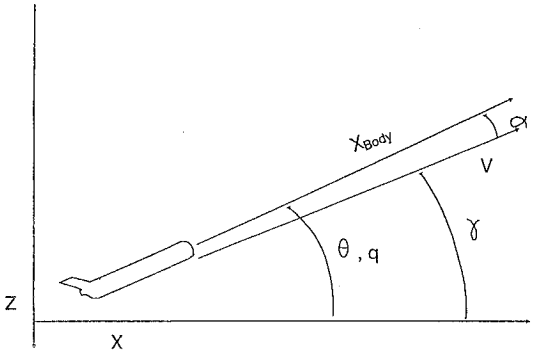

Fig. 7 Aircraft control problem: α, angle of attack; γ, flight path angle; v, velocity; θ, pitch angle; and q, pitch rate.

Figure 6 shows a comparison of the system state being controlled by both the optimal control (Riccati solution) and the control determined by this adaptive-critic-based method for $x(0) = -20$. Note that this initial condition was chosen arbitrarily. The neural network has determined the near optimal control law for each point within its training range.

B. Aircraft Control Application

We consider the synthesis of an optimal longitudinal autopilot in this section. The performance index in this application is an infinite-time quadratic cost function. The minimizing control is expected to drive the deviations of the longitudinal dynamics in pitch angle θ, pitch rate q, forward velocity u', and angle of attack α to zero.

The orientation of an aircraft involving longitudinal dynamics is shown in Fig. 7. The linearized equations of motion of an aircraft in a vertical plane are given by[17]

$$\dot{x} = Ax + Bu \tag{16}$$

where the elements of the state space x are

$$x \equiv [u', \alpha, \theta, q]^T \tag{17}$$

Elements $a_{ij}, i = 1, 2, 3, 4, j = 1, 2, 3, 4$ of the 4×4 matrix A represent the dynamic stability derivatives and are given by

$$\begin{aligned} &a_{11} = -0.0148, \quad a_{12} = -13.88, \quad a_{13} = -32.2 \\ &a_{14} = 0, \quad a_{21} = -0.00019, \quad a_{22} = -0.84 \\ &a_{23} = 0, \quad a_{24} = 1, \quad a_{31} = 0, \quad a_{32} = 0 \\ &a_{33} = 0, \quad a_{34} = 1, \quad a_{41} = 0.00005 \\ &a_{42} = -4.8, \quad a_{43} = 0, \quad a_{44} = -0.5 \end{aligned} \tag{18}$$

Elements $b_{ij}, i = 1, 2, 3, 4, j = 1$ of the 4×1 matrix B represent the control derivatives and are given by

$$b_{11} = -1.1, \quad b_{21} = -0.11, \quad b_{31} = 0, \quad b_{41} = -8.74 \tag{19}$$

The control variable u represents elevator deflection.

The performance index J is formulated so as to keep the pitch angle, pitch rate, normal acceleration, and elevator deflection low and penalize if they exceed the prespecified maximum values. That is,

$$J = \int_0^\infty \left[\left(\frac{\theta}{\theta_{\max}}\right)^2 + \left(\frac{q}{q_{\max}}\right)^2 + \left(\frac{n_z}{n_{z_{\max}}}\right)^2 + \left(\frac{u}{u_{\max}}\right)^2 \right] \mathrm{d}t \tag{20}$$

where $\theta_{\max} = 0.26$ rad, $g_{\max} = 0.31$ rad/s, $n_{z_{\max}} = 6\ g$, and $u_{\max} = 0.1$ rad. Here, n_z represents normal acceleration and g is the gravitational acceleration, which is set at 32.2 ft/s^2. Note that n_z can be obtained in terms of other states as

$$n_z = (V_o/g)\,(q - \dot{\alpha}) \tag{21}$$

where V_o is the steady-state aircraft velocity.

Note that this performance index, with the use of Eqs. (18) and (21), has the form

$$J = \int_0^\infty (x^T Q x + u^T R u + 2x^T P u)\,\mathrm{d}t \tag{22}$$

where Q, R, and P are appropriate weighting matrices in terms of $9\theta_{\max}$, $q_{\max}$, $n_{z_{\max}}$, and $u_{\max}$. These are $9R = 91.32$, $P = (0, 22 \times E{-}4, 0.97, 0)^T$, and

$$Q = \begin{bmatrix} 10.37 & 0 & 0 & 0 \\ 0 & 3.7 \times E{-}7 & 1.65 \times E{-}3 & 0 \\ 0 & 1.65 \times E{-}7 & 7.25 & 0 \\ 0 & 0 & 0 & 14.84 \end{bmatrix}$$

Note that P is present because of cross terms in x and u, which occur after n_z is rewritten in terms of state equations. Solutions to this optimization problem are obtained using the adaptive critic approach described in the last section.

The numerical results from these experiments are presented in Figs. 8–13. Histories of $u'(t)$, $\alpha(t)$, $\theta(t)$, and $q(t)$ with time are presented in Figs. 8–11 respectively. To demonstrate the versatility of the adaptive critic approach, we have presented plots of the neural-network-based states and optimal state histories for arbitrary initial conditions in Figs. 8–11. In each one of the cases, we can observe that the optimal trajectories (from exact Riccati solutions) and the neural-network-based solutions are virtually identical. All of these control outputs are generated from one converged neural network. In other words, the action network can be used as one repository of gains for various operating conditions or errors. In other words, it can be an ultimate gain scheduler. It is a feedback controller since the inputs are the current states and the outputs are the control values. Note that no external training is necessary to achieve this. The optimal control history and the neural-network-based control history are presented in Fig. 12. From the control error plot given in Fig. 13, it can be seen that at any point the errors are of the order of 10^{-3}.

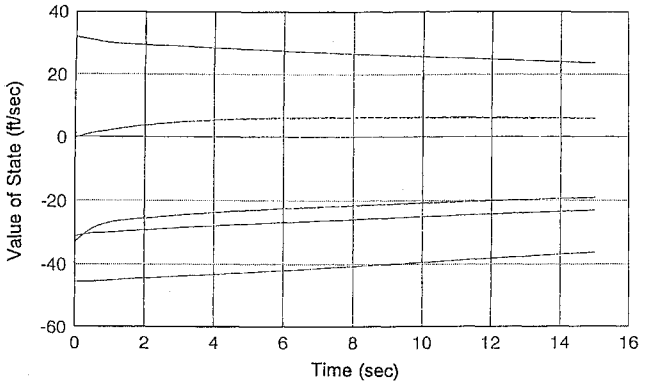

Fig. 8 Value of $u'(t)$ for various initial conditions solid-neural network determined value dashed-optimal control.

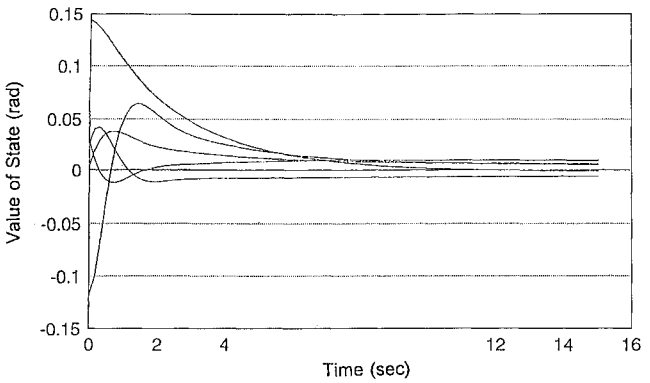

Fig. 9 Value of $\alpha(t)$ for various initial conditions solid-neural network determined value dashed-optimal control.

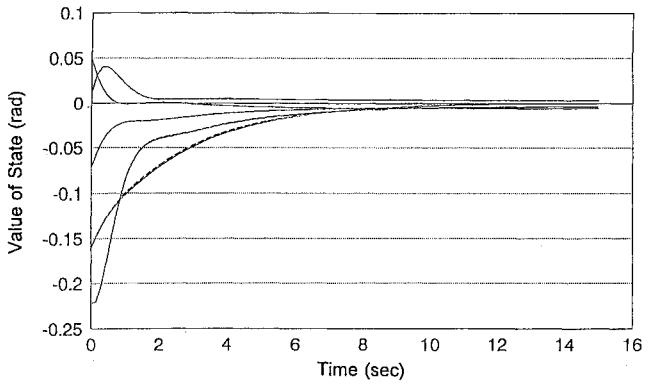

Fig. 10 Value of $\theta(t)$ for various initial conditions solid-neural network determined value dashed-optimal control.

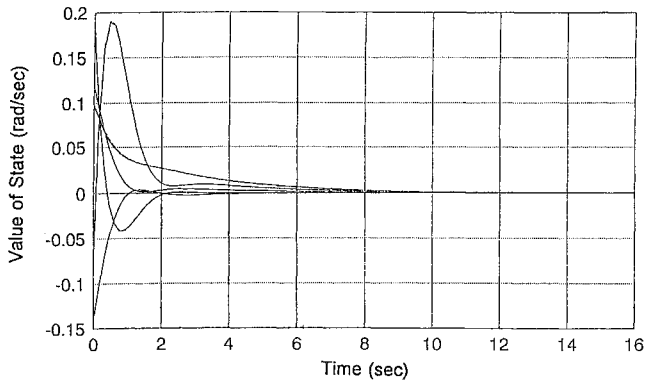

Fig. 11 Value of $q(t)$ for various initial conditions solid-neural network determined value dashed-optimal control.

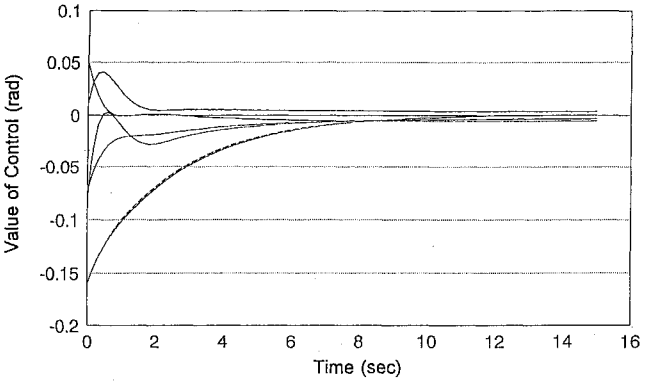

Fig. 12 Value of $u(t)$ for various initial conditions solid-neural network determined value dashed-optimal control.

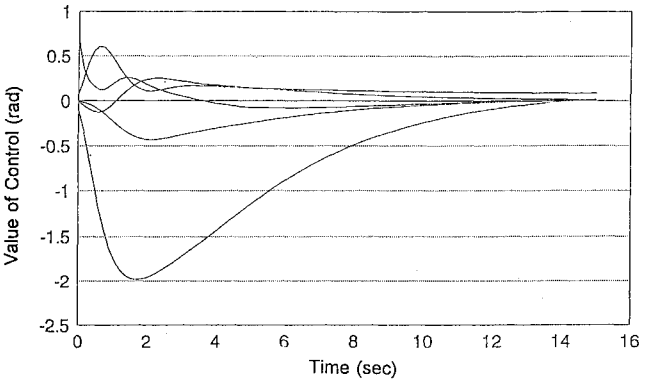

Fig. 13 Plot of error between neural network and optimal control for various initial conditions for $u(t)$.

IV. Conclusions

A new adaptive-critic-based neural architecture has been presented to solve optimal control problems. A scalar problem has been solved to illustrate the steps in the design process of the dual network structure. It has been shown that these networks can produce near optimal control policies for infinite horizon problems such as an aircraft control problem. This architecture requires no external training data and yields optimal control through the entire range of operation and can be used in closed loop. Since the controller network contains an envelope of gains, it can act as an autopilot. The added advantage of this approach is that the critic network can provide fault tolerance. Future work on this topic will investigate the robustness of such network controllers and the use of this method for finite-horizon class of problems.

Acknowledgments

This study was funded by National Science Foundation (NSF) Grant ECS-9313946 and by the Missouri Department of Economic Development Center for Advanced Technology Program. The authors thank Paul Werbos of NSF for his technical interactions during this study. Comments of some reviewers have been very informative. A patent application is in progress.

References

[1]Bryson, A. E., and Ho, Y., *Applied Optimal Control*, Hemisphere, New York, 1975, pp. 128–211.

[2]Howard, R. A., *Dynamic Programming and Markov Processes*, Technology Press of the Massachusetts Inst. of Technology and Wiley, New York, 1960, Chaps. 3–6.

[3]Werbos, P. J., "Maximizing Long-Term Gas Industry Profits in Two Minutes in Lotus Using Neural Network Methods," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 2, 1989, pp. 315–333.

[4]White, D. A., and Sofge, D., *Handbook of Intelligent Control*, Van Nostrand Reinhold, New York, 1992, Chaps. 3, 5, 8, 12, and 13.

[5]Miller, W. T., Sutton, R. S., and Werbos, P. J., *Neural Networks for Control*, MIT Press, Cambridge, MA, 1990.

[6]Ismail, F., Wahsh, S., Mohamed, A., and Elsimary, H., "Neural Network Application to an Optimal Control of a Variable Reluctance Motor," *Proceedings of the 35th Midwest Symposium on Circuits and Systems* (Washington, DC), Vol. 2, Inst. of Electrical and Electronics Engineers, 1992, pp. 1048–1051.

[7]Nikolauo, M., and Hanagandi, V., "Control of Nonlinear Dynamical Systems Modeled by Recurrent Neural Networks," *AIChE Journal*, Vol. 39, No. 11, 1993, pp. 1890–1894.

[8]Yamada, T., and Yabuta, Y., "Nonlinear Neural Network Controller for Dynamic System," *IECON '90. 16th Annual Conference of IEEE Industrial Electronics Society* (Pacific Grove, CA), Vol. 2, Inst. of Electrical and Electronics Engineers, 1990, pp. 1244–1249.

[9]Fu, K. S., "Learning Control Systems," *Computer and Information Sciences*, edited by J. T. Tou and R. H. Wilcox, Spartan, Washington, DC, 1964, pp. 318–343.

[10]Sklansky, J., "Learning Systems for Automatic Control," *IEEE Transactions on Automatic Control*, Vol. 11, No. 1, 1966, pp. 6–19.

[11]Nikolic, Z., and Fu, K. "An Algorithm for Learning without External Supervision and Its Application to Learning Control Systems," *IEEE Transactions on Automatic Control*, Vol. 11, No. 3, 1966, pp. 414–422.

[12]Widrow, B., Gupta, N., and Maitra, S., "Punish/Reward: Learning with a Critic in Adaptive Threshold Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 3, No. 5, 1973, pp. 455–465.

[13]Barto, A. G., Sutton, R. S., and Anderson, C. W., "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, Sept.–Oct. 1983, pp. 834–846.

[14]Anderson, C. W., "Strategy Learning with Multilayer Connectionist Representations," *Proceedings of the 4th International Workshop on Machine Learning* (Irvine, CA), Morgan Kaufmann, San Mateo, CA, 1987, pp. 103–114.

[15]Maren, A., Marsten, C.-T., and Pap, R.-M., *Handbook of Neural Computing Applications*, Academic, New York, 1990.

[16]Wasserman, P. D., *Neural Computing Theory and Practice*, Van Nostrand Reinhold, New York, 1989.

[17]Friedland, B., *Control System Design*, McGraw–Hill, New York, 1986, Chap. 4.