01 Jan 1996

A Dual Neural Network Architecture for Linear and Nonlinear Control of Inverted Pendulum on a Cart

S. N. Balakrishnan
Missouri University of Science and Technology, bala@mst.edu

Victor Biega

Recommended Citation

S. N. Balakrishnan and V. Biega, "A Dual Neural Network Architecture for Linear and Nonlinear Control of Inverted Pendulum on a Cart," *Proceedings of the 1996 IEEE International Conference on Control Applications, 1996*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1996.
The definitive version is available at https://doi.org/10.1109/CCA.1996.558932

Proceedings of the 1996 IEEE International
Conference on Control Applications
Dearborn, MI • September 15-18, 1996

A DUAL NEURAL NETWORK ARCHITECTURE FOR LINEAR AND NONLINEAR CONTROL OF INVERTED PENDULUM ON A CART

Victor Biega and S. N. Balakrishnan

Department of Mechanical and Aerospace Engineering and Engineering Mechanics
University of Missouri-Rolla
Rolla, MO 65409-0050
bala@umr.edu

Abstract

The use of a self-contained dual neural network architecture for the solution of nonlinear optimal control problems is investigated in this study. The network structure solves the dynamic programming equations in stages and at the convergence, one network provides the optimal control and the second network provides a fault tolerance to the control system. We detail the steps in design and solve a linearized and a nonlinear, unstable, four-dimensional inverted pendulum on a cart problem. Numerical results are presented and compared with linearized optimal control. Unlike the previously published neural network solutions, this methodology does not need any external training; solves the nonlinear problem directly and provides a feedback control.

I. Introduction

Optimization has been a field of interest to mathematicians, scientists and engineers for a long time. Problems of optimization of functions or functionals and optimal control of linear or nonlinear dynamical systems can be solved through direct or indirect methods [Bryson and Ho [1]]. In direct methods where, in general, the cost function is evaluated or indirect methods where, in general, values of the derivatives are used to check optimum, separate solutions are obtained for each set of parameters or initial conditions. For optimal solutions which encompass perturbations to the assumed initial conditions or a family of initial conditions, we can use neighboring optimal control [1] or dynamic programming [1]. Neighboring optimal control allows pointwise solutions of an (optimal) two-point boundary value problem (TPBVP) to be used with a linearized approximation over a range of initial conditions. However, the neighboring optimal solution can fail outside where linearization is invalid. Dynamic programming can handle a family of initial conditions for linear as well as nonlinear problems. The usual method of solution, however, is computation-intensive. Furthermore, the solution is not available in a feedback form either (usually) and for implementation, this becomes a drawback.

Outside of dynamic programming, currently there is no unified mathematical formalism under which a controller can be designed for nonlinear systems. Techniques like feedback linearization have been used for a few nonlinear problems under limited conditions, such as equal number of inputs and outputs. More rigorous and general solutions are available with linearized models; however, they are restricted by the assumption of linear models. Other available solutions for nonlinear controllers are highly problem oriented. Consequently, we propose a formulation which: 1) solves a nonlinear control problem directly without any approximation to the system model (in the absence of a good model this approach can synthesize a nonlinear model of the states), 2) yield a control law in a feedback form as a function of the current states, and 3) maintain the same structure regardless of the type or problem (handles linear problems as well). Such a formulation is afforded by the field of neural networks.

Several authors have used neural networks to "optimally" solve nonlinear systems [Hunt [2], White and Sofge [3]]. Almost all these studies fall within four categories: 1) supervised control, 2) direct inverse control, 3) neural adaptive control, and 4) backpropagation through time [19]. A fifth and rarely studied class of controller has the most interesting structure. It is called an Adaptive Critic Architecture (Figure 1) [3,4,8]. The reason for choosing this structure for formulating the optimal control problems are that this approach needs NO external training as in other forms of neurocontrollers, this is not an open loop optimal controller but a feedback controller, and it preserves the same structure regardless of the problem (linear or nonlinear). Balakrishnan and Biega [4] have shown the usefulness of this architecture for finite-time linear problems. In this study, we seek to present a general neural framework for the study of linear as well as nonlinear, infinite-time optimal control problems. It is hoped that the two other important features associated with a controller, namely robustness and identification (observers) with neural networks can be studied in a more general way with our network structure since it does not have any restrictive assumption (like linearity, etc.).

The method discussed in this study determines an optimal control law for a system by successively adapting two networks, an action network and a critic network. This method determines the control law for an entire range of initial conditions [4]. It simultaneously determines and adapts the neural networks to the optimal control policy for both linear and nonlinear systems.

II. Solution Method Development

A. Neural Network Background

Neural networks, or in the case of this paper multi-layer perceptrons (MLP's), 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 MLP's of the form shown in Figure 2. In this case, the activation functions used are

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

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 [3].

B. Problem Formulation

In this study, problems of the form

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

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

$$t_f\equiv given \qquad x_o\equiv given \tag{4}$$

are being considered. The first step taken is to discretize them into the form (for use with the neural networks)

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

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

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

In these equations, $\psi(\cdot)$ is a scalar, nonlinear time-varying function of the n-dimensional state vector, x and m-dimensional control vector, u. The system model is given by $f_D(\cdot)$. The final time, t_f, is discretized into N equal steps. The initial conditions x(o) are assumed known. N $\rightarrow$ large represents an infinite-time problem.

C. Dynamic Programming Background [Bryson [1]]

Assuming the cost function in Eqn. 5, one can rewrite J($\cdot$) as

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

(Here, J(x(t)) is the cost associated with going from time t to the final time. U(x(t),u(x(t)) is the utility, which is the cost from going from time t to time t+1. Finally, <J(x(t+1))> is assumed to be the minimum cost associated with going from time t+1 to the final time.

If both sides of the equation are differentiated and we define

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

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 <λ(x(t+1))>, U(x(t),u(t)) and the system model derivatives are known then λ(x(t)) can be found.

Next the optimality equation is defined as

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

Dynamic programming uses these equations to aid in solving an infinite horizon policy or to determine the control policy for a finite horizon problem.

D. Training Methods (Approximation Techniques)

As mentioned earlier, this study uses Eqn. 10 in order 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 λ(x(t))). Both networks are assumed to be feedforward MLP's.

In order 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 (δx(t+1))/(δu(t)). Next the critic from t+1 is used to find λ(x(t+1)). This information and Eqn. 11 are 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, x(t) is randomized and the output of the critic λ(x(t)) is found. The action network from step t calculates u(t) and (δu(t))/(δx(t)). (Note that the functions which determine u(t) from x(t) are known for the MLP action network, therefore (δu(t))/(δx(t)) can be found using ordered derivatives.) The model is then used to find (δx(t+1))/(δx(t)), (δx(t+1))/(δu(t)) and x(t+1). The critic from step t+1 is then used to find λ(x(t+1)). After this, Eqn. 10 is used to find $\lambda^*(x(t))$, the target value for the critic. The difference between λ(x(t)) and $\lambda^*(x(t))$ is then used to update the critic using a standard backpropagation algorithm. This process is continued until a predetermined level of convergence is reached. Training of the critic network is presented schematically in Figure 3. This two-step process is repeated till the control network outputs show no difference above a specified tolerance.

III. Applications

The first application is an infinite horizon four dimensional linear problem, specifically, that of the linearized system for an inverted pendulum and a cart. Second, this method is applied to finding the optimal control policy for the nonlinear equations describing the inverted pendulum problem, and the corresponding cost improvement from the linear equations is shown.

A. First Application (Infinite Time 4-D Linear) [Friedland [5]]

The classic inverted pendulum on a moving cart (Figure 4) is concerned with the infinite horizon problem, i.e. steady state solution.

The linearized equations of motion for this system are (with the mass of the pendulum as 0.1 kg, the mass of the cart as 1 kg, the length of the inverted pendulum as 1 m and gravitational acceleration as 9.8 m/s².)

$$\begin{vmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{vmatrix} = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.98 & 0 & 0 \\ 0 & 10.78 & 0 & 0 \end{vmatrix} \begin{vmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ 1 \\ -1 \end{vmatrix} u \qquad (12)$$

where x is the horizontal distance of the center of gravity of the cart from a reference point and θ is the angle of the inverted pendulum to the vertical. Dots denote differentiation with respect to time. The mass of the pendulum is assumed to be concentrated at a point and the length is assumed constant. The problem is to find the control which minimizes the cost functions, J, given by

$$J = \int_0^\infty (x(n)^2 + \theta(n)^2 + \dot{x}(n)^2 + \dot{\theta}(n)^2 + u(n)^2)dt \qquad (13)$$

The first step is to discretize the system for use with neural networks. By using Euler's approximation for a time step of 0.01 seconds, the discretized linear system becomes

$$\begin{vmatrix} x(n+1) \\ \theta(n+1) \\ \dot{x}(n+1) \\ \dot{\theta}(n+1) \end{vmatrix} = \begin{vmatrix} 1 & -4.9e-5 & 1e-2 & -1.6334e-7 \\ 0 & 1.0005 & 0 & 1.0002e-2 \\ 0 & -9.8018e-3 & 1 & -4.9004e-5 \\ 0 & 1.0782e-1 & 0 & 1.0005 \end{vmatrix} \begin{vmatrix} x(n) \\ \theta(n) \\ \dot{x}(n) \\ \dot{\theta}(n) \end{vmatrix} + \begin{vmatrix} 5e-5 \\ -5.0004e-5 \\ 1e-2 \\ -1.0002e-2 \end{vmatrix} u(n) \qquad (14)$$

with a cost function of

$$J = \sum_{n=0}^{\infty} x(n)^2 + \theta(n)^2 + \dot{x}(n)^2 + \dot{\theta}(n)^2 + u^2(n) \qquad (15)$$

Since this problem is concerned with steady states, a single utility function can be defined. For this problem, the correct utility function is

$$U(n) = (x(n) + \theta(n) + \dot{x}(n)^2 + \dot{\theta}(n)^2 + u(n)^2) \qquad (16)$$

As a first step, some control must be specified. For this problem the original controller was chosen by defining λ(n+1)=0 and training an action network using a gradient descent algorithm.

As a next step, a neural network is randomized to act as the critic network. Now Eqn. 10 can be used to train the critic. (It is important to note that during the steady state operation of a system, the cost associated with a specific state at time (n+1) is equivalent to the cost associated with that same state at time (n). Therefore the values for both λ(n) and λ(n+1) are taken from the same critic.)

After the critic has converged, a new neural network is randomized to act as the action network. The previously determined critic and the utility function have been specified, therefore it is possible to adapt the action network using a gradient descent algorithm.

After the action network has converged, the critic network is trained using the action network as the new control law. This method is repeated until the critic and action networks have completely converged (i.e. repetition of this process does not change controller network)

Figure 5 shows a plot of the trajectory of x1 (position) for the control law determined by the neural network method and the control law determined by LQR (Ricatti Equation) [1,2] methodology. As desired, the trajectories are very nearly identical. Likewise, Figures 6-8 show that the same characteristics are exhibited in the plots of angle of the inverted pendulum, cart velocity, and the angular velocity of the inverted pendulum. All neural- network controlled trajectories agree closely with the LQR determined trajectories. The closeness of the single neural network generated control for a span of initial conditions throughout the trajectory to several separate LQR generated trajectories is seen in Figure 9. It should also be noted that the cost of the neural network determined trajectories for a randomly generated set of initial conditions agree within 1.5% with the costs calculated using LQR based control. Note that the control law can be determined for **any** point in the training range and it will be optimal. This is a unique and extremely useful feature of this methodology since it represents the approximate dynamic programming solutions.

B. Second Application (Infinite Time 4-D Nonlinear)
The second application once again deals with the nonlinear motion of the inverted pendulum on a cart, except that the exact nonlinear equations are used without any linearization.

Anderson [6] and Barto, et al. [7] have investigated the control of the inverted pendulum on a cart using neural networks. Their objectives, though, are different as compared with this study. They are concerned with a stable control of the pendulum, not optimal control. It is assumed that the system model is not known and that the only feedback signal is a failure signal when the angle exceeds some given maximum. The following problem will involve the use of a neural network to directly solve for the optimal solution of the nonlinear equations for the inverted pendulum on a cart problem, assuming a given model. Introduction of an identification network in the loop is not difficult.

The problem which is being examined consists of the system described by the following equations

$$\begin{aligned} \ddot{x} = \frac{1}{1.1}&\left(u + 0.1\,\dot{\theta}^2 \sin(\theta) - \right. \\ &\frac{0.1}{1.1}\cos(\theta)\,\frac{1}{1.1}\,(0.98\sin(\theta) \\ &- (u + 0.1\,(\dot{\theta})^2\sin(\theta))\,\frac{0.1}{1.1}\cos(\theta)\,) \\ &\left. / \; 0.1 - \frac{0.1}{1.1}\cos(\theta)^2 \right. \end{aligned} \tag{17}$$

$$\begin{aligned} \ddot{\theta} = (&\frac{1}{1.1}(0.98\sin(\theta) \\ &- (u+0.1(\dot{\theta})^2\sin(\theta))\frac{0.1}{1.1}\cos(\theta)\,) \\ &/ \; 0.1 - \frac{0.1}{1.1}\cos(\theta)^2 \end{aligned} \tag{18}$$

The cost function assigned to this problem is

$$\begin{aligned} J = \int_0^\infty &(x(t)^2 + \theta(t)^2 \\ &+ \dot{x}(t)^2 + \dot{\theta}(t)^2 + u(t)^2)\,dt \end{aligned} \tag{19}$$

Once again, it is assumed that the optimal discrete control for a time period of 0.01 seconds is desired. In order to use Eqn. 10, some modifications have to be made. First, it is important to note that the nonlinear system is modeled in a discrete manner by a fourth order Runga-Kutta simulation on a 0.01 second interval. All derivatives are determined using forward estimation with the system model. The discrete utility function used is the same as that used in the linear problem. This provides the derivatives necessary for use of Eqn. 10.

The first step in this process is to define an arbitrary initial control. The initial control will be assumed to be the same as that in the linear problem. Next, a neural network must be randomized to act as the critic network. This is carried out by using Eqn. 10.

When the critic network converges, a new neural network is randomized to act as the action network. The appropriate control law is determined using a gradient descent algorithm. The training then proceeds exactly as in the linear case. (Notice that the training using the nonlinear equations is no more difficult than using the linear system equations.)

Figure 10 shows a plot of the trajectories for x1 (position) for the neural network determined nonlinear control and the LQR based optimal control. Note that unlike the linear case, the neural network based trajectory is noticeably different than that of the optimal LQR trajectory. Figures 11-13 show the same trajectories for the pendulum angle, cart velocity and pendulum angular velocity. In each case the trajectory varies slightly.

The costs are calculated by integration of the performance index in Eqn. 19. The neural network controlled trajectories shown in Figures 10-13 result in a reduction of overall cost by 5.7%. By repeating the process for a random set of initial states, the overall cost is found to be reduced from between 2.5% and 8.5% compared to the cost determined by the linearized system based control for the same conditions. This clearly shows that a nonlinear neural controller is more efficient than a gain-scheduled linear controller. The neural network structure in this study can handle either type of controller. Furthermore, it should be noted that the critic network functions as a check to see whether the controller outputs are optimal, thereby, providing a measure of fault tolerance. Unlike the LQR controller, the same network contains the nonlinear feedback gains to a **span of initial conditions** since the network represents the solution to a dynamic programming formulation.

IV. Conclusions

A dual neural network architecture, called the 'adaptive critic' has been presented in this study to determine optimal control for a linear and a nonlinear inverted pendulum. The dual network, which does not need external target data, embeds the dynamic programming solutions and offers a feedback type optimal controller for an entire range of initial conditions. The network structure is invariant to linear or nonlinear problems. Therefore, it is hoped that we have obtained a general framework for neural network on which we can base stability and robustness studies. It is also easily extendable to identification problems.

Acknowledgments

This study was funded by National Science Foundation of the United States, Grant ECS-9313946, and by the Missouri Department of Economic Development Center for Advanced Technology Program. The authors thank Dr. Paul Werbos for his useful and insightful technical interactions during this study.

References

[1] A. E. Bryson and Y. Ho, *Applied Optimal Control*, Hemisphere Publishing Co., 1975, pp. 128-211.
[2] K. J. Hunt, "Neural Networks for Controller Systems, A Survey," *Automatica*, Vol. 28, No. 6, 1992, pp. 1083-1112.
[3] D. A. White and D. Sofge, *Handbook of Intelligent Control*, Van Nostrand Reinhold, 1992, Ch. 3,5,8,12,13.
[4] S. N. Balakrishnan and V. Biega, "A New Neural Architecture for Homing Missile Guidance," *Proc. of the American Control Conference*, Seattle, WA, June 1995, pp. 2148-2152.
[5] Friedland, *"Control System Design,"* McGraw Hill, 1985, Ch. 4.
[6] C. W. Anderson, "Learning to Control an Inverted Pendulum using Neural Networks," *IEEE Control Systems Magazine*, April 1898, Vol. 9, pp. 31-37.
[7] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, 1983, Vol. SMC-13, pp. 834-846.
[8] P. Werbos, "Neurocontrol and Supervised Learning; An Overview and Evaluation," *Handbook of Intelligent Control*, Van Nostrand Reinhold, 1992.

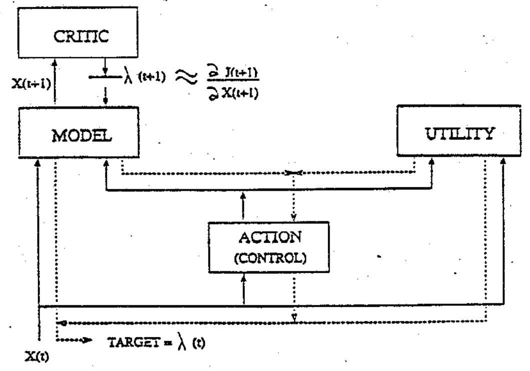

Figure 1. Adaptive Critic for Control

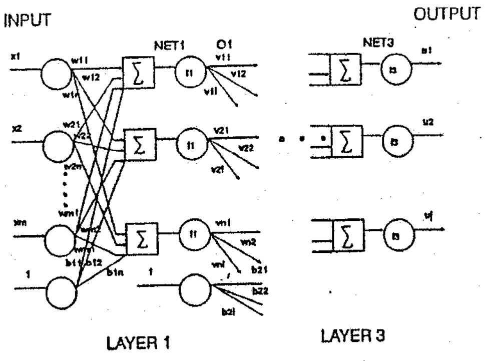

FIGURE 2. STANDARD MULTI-LAYER PERCEPTRON

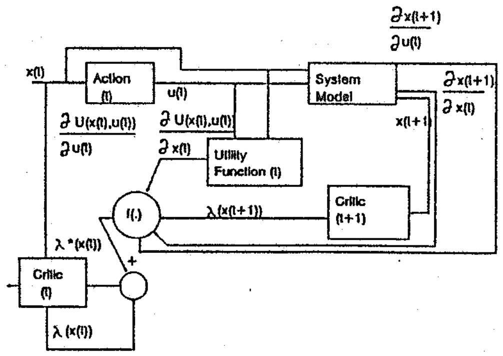

Figure 3. Critic Network Training Diagram

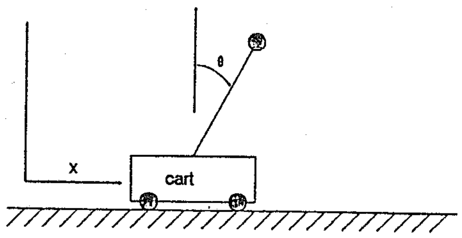

Figure 4.
INVERTED PENDULUM ON A CART

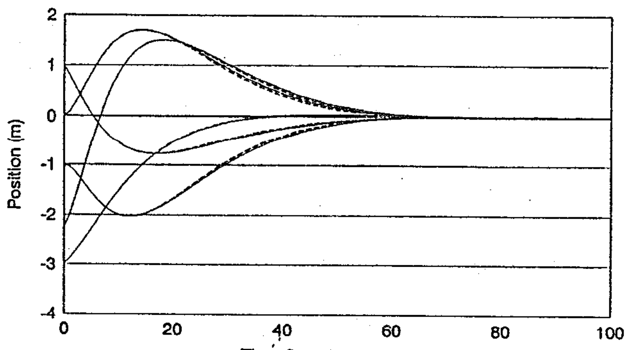

Figure 5.
Plot of Values for Position for Random Initial Conditions
solid- Neural Network Determined Value
dashed- Optimal

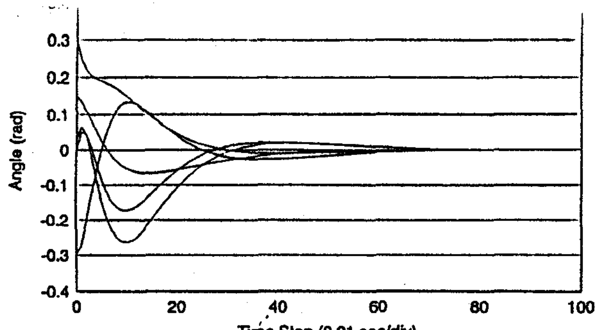

Figure 6.
Plot of Values for Angle for Random Initial Conditions
solid- Neural Network Determined Value
dashed- Optimal

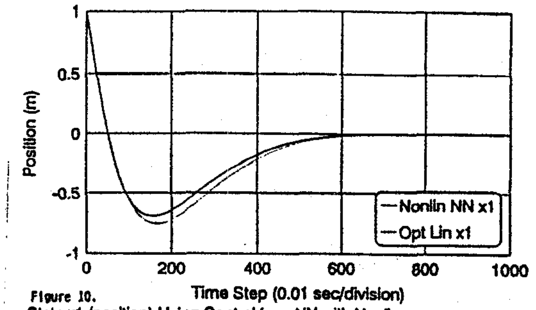

Figure 10.
State x1 (position) Using Control from NN with Nonlinear System Model and from Optimal Control of Linearized System for Pendulum

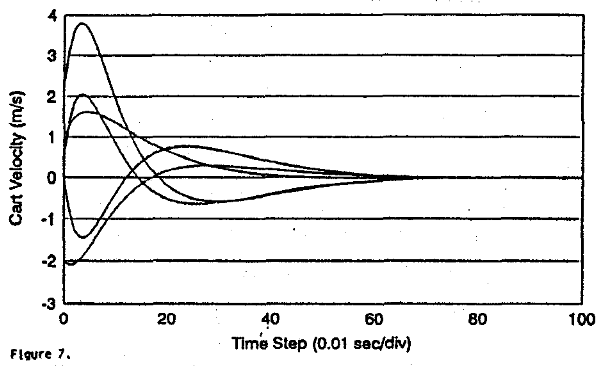

Figure 7.
Plot of Values for Cart Velocity for Random Initial Conditions
solid- Neural Network Determined Value
dashed- Optimal

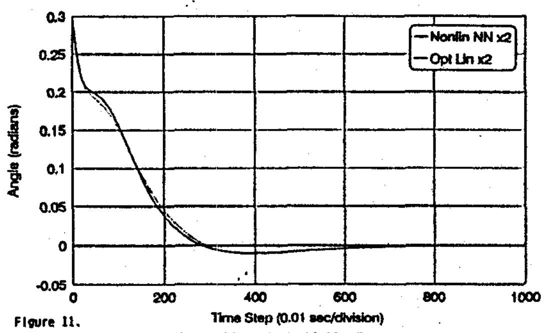

Figure 11.
State x2 (angle) Using Control from NN with Nonlinear System Model and from Optimal Control of Linearized System for Pendulum

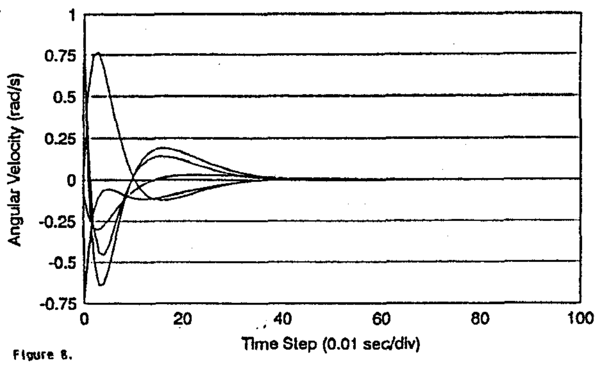

Figure 8.
Plot of Values for Angular Velocity for Random Initial Conditions
solid- Neural Network Determined Value
dashed- Optimal

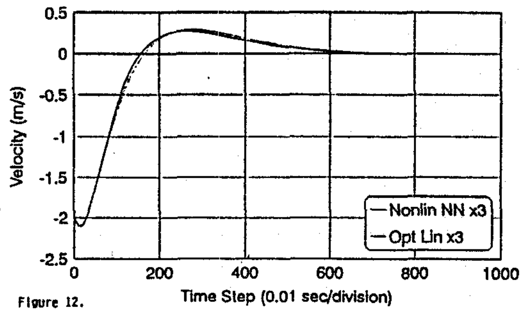

Figure 12.
State x3 (velocity) Using Control from NN with Nonlinear System Model and from Optimal Control of Linearized System for Pendulum

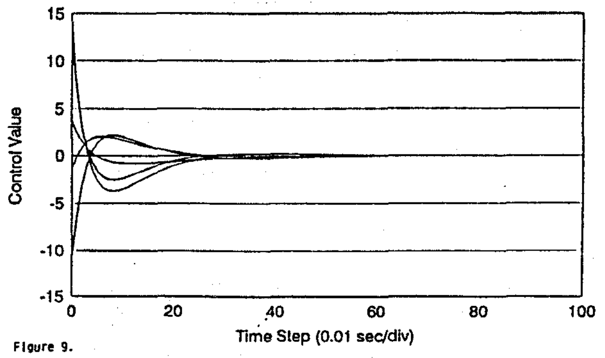

Figure 9.
Plot of Values for Control for Random Initial Conditions
solid- Neural Network Determined Value
dashed- Optimal

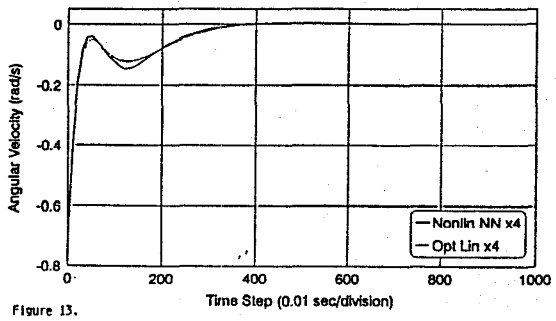

Figure 13.
State x4 (angular velocity) Using Control from NN with Nonlinear System Model and from Optimal Control of Linearized System for Pendulum