Pergamon

Mathl. Comput. Modelling Vol. 22, No. 1, pp. 63–78, 1995

0895-7177/95 $9.50 + 0.00

0895-7177(95)00101-8

Neurocontrol of an Aircraft: Application to Windshear

S. M. AMIN, E. Y. RODIN AND Y. WU
Center for Optimization and Semantic Control
Department of Systems Science and Mathematics
Campus Box 1040, Washington University in St. Louis
St. Louis, MO 63130-4899, U.S.A.
massoud@rodin.wustl.edu

(Received and accepted March 1995)

Abstract—In this report, we consider the part of our work which concerns the design of neuroidentifiers and neurocontrollers which attenuate the effects of disturbances. Examples for linear-systems identification and disturbance rejection, as well as nonlinear control of an aircraft encountering wind shear on take-off are briefly discussed, and the following three problems are addressed.

1. System identification via dynamic neural networks.
2. Disturbance attenuation via memory neurons.
3. Aircraft control in the presence of wind shear after takeoff.

Keywords—Dynamic neural networks, Robust control of nonlinear systems, Aircraft control, Systems identification and control.

1. INTRODUCTION

The lack of rigorous mathematical representation of control systems in current paradigms of feedforward and recurrent neural networks is a drawback to the development of research on neural networks for control. The feed-forward networks are known to work as a mapping between two information domains. Most of the current research in neurocontrol and related publications discusses this type of architecture for learning a model or a controller, which is usually either nonlinear or difficult to implement. The published results show that while these approaches yield satisfactory results in many cases, there is little development in relating the theories of classical and modern control systems to neural networks. Neural networks are usually treated as "Black Boxes" and thus, there is no direct contact with the "internal" information of the "Box." A linear control system, which may also be called a "Black Box," can be represented by transfer functions, matrix fraction representations, and/or other input-output, as well as frequency response parametrizations. Therefore, the input-output relationship, as well as performance, can be studied thoroughly. In our previous work [1–5], the "internal information" of the network is parametrized as a control system; the goal has been to represent the identifier and the controller in terms of this information, and thus integrate two types of dynamic neural network architectures into controllers. Suitability of feedforward architectures with dynamic neurons for identification

This research was supported in part by AFOSR under Grant No. F49620-93-1-0012. Earlier versions of this work have been reported in references [1–5] and at NASA Ames Research Center.

and control of dynamic systems has been shown; several important issues concerning controllability, observability, and feedback linearizability of such neural models are also investigated. The network itself is not only a control system, but is also capable of learning and improvement. In this report, we provide a brief literature survey as well as discuss a relevant aerospace-related example.

A noticeable advance and development in parallel computation and parallel algorithms has occurred in the past decade. A highly parallel structured computer is capable of performing multiple tasks simultaneously, and operates much faster. Parallel computations and architectures have become important issues in the control community. Some achievements have been made in utilizing a parallel computational mechanism to compute inverse dynamics of a robot arm. In particular, *Flavor*^{TM}*'s* Parallel Inference Machine exemplifies the utilization of parallelism in control systems. One important requirement for applying parallelism in control systems is to establish a solid mathematical foundation for it. Clearly, the interaction between computer developments and control systems mutually reinforces both disciplines. It is generally recognized that the rapid development in sequential computers in the 1950's–70's motivated a remarkable advancement in control system synthesis and design. During this period, optimal control and estimation, multivariable control, and adaptive control were making great advances, with many important results such as the Maximum Principle, Kalman-Bucy filter, State-Space analysis and techniques, etc. It is also interesting to notice that the developments of discrete-time systems and discrete-time state-space representation were mainly motivated by the availability of sequential computers of the time. In view of today's rapid development in parallel computers and parallel algorithms, it is natural to consider the problem of how to model control systems, by appropriate mathematical tools, using the mechanism of parallel computing. This has been the main motivation for our work during the past five years. In this period, we have developed theories and implemented simulators for the incorporation of dynamic and memory neurons for real-time system identification and control [1–5]. By going beyond the universal approximation property of neural networks, we have considered the internal state information of the recurrent neural networks so that a control system can be modeled using the highly parallel structure of this computational mechanism. Based on a new paradigm of neural networks consisting of Neurons with Local Memory (NLMs), the representation of a control system was discussed [2,5]. Modeled by NNLM, the resulting system is a nonlinear one that, through mathematical analysis [5], was shown to be locally linearizable via a static feedback and a nonlinear coordinate transformation. We have applied a similar methodology to the design of a neurocontroller for aircraft (using data for a Boeing 727), where a differential game-based neurocontroller formulation was used to reduce the effects of external disturbances on the aircraft.

2. LITERATURE SURVEY

A brief survey of previous work dealing with dynamic neural nets follows.

Farotimi *et al.* [6] provided a weight synthesis technique based on optimal control theory for dynamic neural networks. Gori *et al.* [7] presented a back propagation algorithm for a particular class of dynamic neural networks where some processing elements have a local feedback, and applied this class of neural networks to the problem of speech recognition. Gherrity [8] derived a learning algorithm for a recurrent neural network which is described by a system of coupled differential equations. Willis *et al.* [9] discussed advantages of a neural network estimator for feedback process control.

Perfetti [10] considered the effect of positive self feedback on neural networks, and showed that binary output can be guaranteed with finite sigmoid slope such that nonbinary solutions become unstable. Sudharsanan and Sundareshan [11] proposed a descent procedure as a learning rule to minimize the error between the stable equilibrium points of the network and the desired memory vectors. Sato *et al.* [12] discuss their work on an adaptive nonlinear pair oscillator with local

connections (APOLONN) used for speech synthesis. Tan *et al.* [13] discuss stability issues for asymmetric dynamic neural networks. Mckelvey [14] presents a method for developing controllers for nonlinear systems based on an optimal control formulation. The network was trained with the backpropagation algorithm where examples of optimal controls previously calculated with a differential dynamic programming technique were used. Werbos [15] and Yamaguchi *et al.* [16] present overviews of neurocontrol and fuzzy logic formulations for aerospace applications.[1] More recently, Hunt *et al.* [18] provided a survey of neurocontrollers for nonlinear systems. They point out that in the 1950's, the field of cybernetics consisted of control, information, and neural science. Since then, the disciplines of control, computing science, and neurobiology have tended to go separate ways. The authors trace the origins of neurocontrol to the work of Wiener (1948) and more recently to Tsypkin (1971). They also point out important properties of neural nets in control including

(1) their ability to approximate arbitrary nonlinear functions;
(2) parallel processing which provides fault tolerance and can be implemented in hardware;
(3) capability to learn and adapt to a changing environment;
(4) perform data fusion on both quantitative and qualitative data; and
(5) ready application to multivariable systems.

One appealing feature is the potential of neural nets to provide a generic "blackbox" representation for nonlinear models; this has led to investigation of the approximation capabilities of neural nets. An important question is that of system identifiability [19] i.e., can a system be "adequately" represented via a given model structure?

At the current stage of work, there are two fundamental approaches which can lead to satisfactory answers to the above question.

(1) Neuroidentifier design based on adaptive control architectures. For dynamic plants use a delay line; include past values of the plant outputs as inputs to the neuroidentifier. If the static and dynamic effects are inseparable, use a general neural network model where past inputs to the system as well as past outputs from the plant are fed as inputs to the neuroidentifier. The number of delays can be obtained either from the characteristics of the plant (e.g., order of the transfer function, if available) or from prior knowledge, experimentation, or physics of the problem (see [20–22]). Most of these approaches are based on replacing the traditional systems identification/controllers described, for example, in adaptive control literature [19,23–26] with a neuroidentifier or neurocontroller.
(2) Use of ontogenic neural networks which add layers and neurons during training. This is a promising area for investigation.

Adaptive control of unknown plants via dynamic neural networks is also discussed in [27]. In this report, we use a similar architecture for the design of a neuroidentifier. There are many more references for both theoretical studies as well as practical applications of neural nets. We refer the reader to several excellent papers collected in Neurocomputing and Neurocomputing 2, as well as more references in the above mentioned sources. There are more specialized surveys in the literature; for robotics applications (see [28,29]).

3. IDENTIFICATION

Recently, ANNs have been used to approximate dynamic systems [22,27,30–34]. The design of a neurocontroller is nontrivial when these systems are continuous and/or when an exact model is unknown. The main challenge in neurocontrol design is how to generate the input/output pairs for the controller training, where errors might be specified between the reference model and the system-neurocontroller outputs; in other words, how to backpropagate the errors at the system output to its input.

[1] For a collection of papers on neurocontrol, see [17].

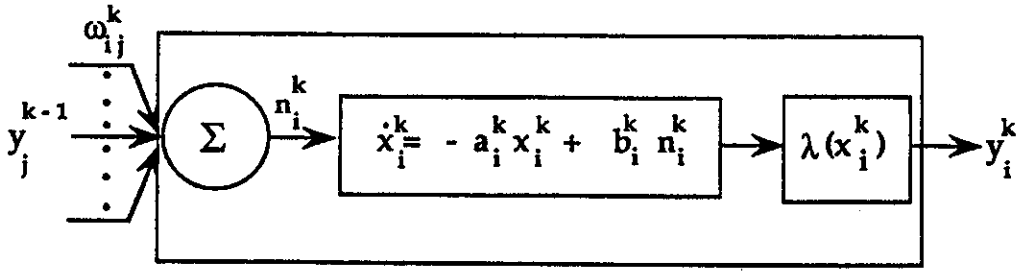


Figure 1. Note i in k^{th} hidden layer of the neuroidentifier.

In this implementation, dynamic neurons are used (Figure 1), with each node being governed by a first-order differential equation of the Hopfield-model type [35]. Due to the dynamic nodes, the common backpropagation algorithm cannot be used. The learning rule is derived using the Distributed Dynamic Backpropagation algorithm (DDBP) [33]. This simulator is implemented in C and has an easy interface for analyzing states of the nodes. The simulator defines hierarchical neural networks with either dynamic or static nodes with user-chosen numbers of layers and numbers of nodes in each layer. "Static" node means that the node output is only governed by the input and activation function.

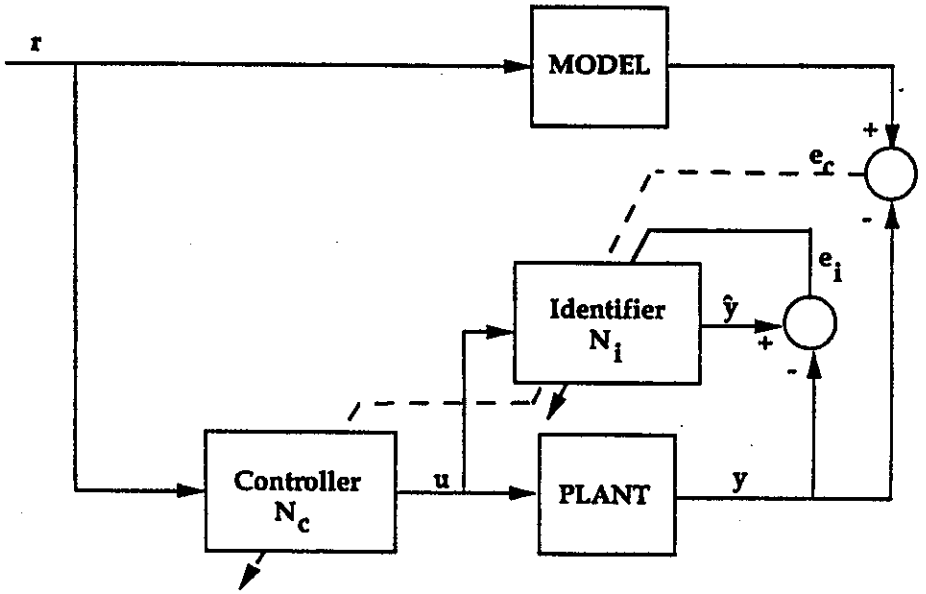


Figure 2. Block diagram of identification and control based on the indirect adaptive control architecture.

The identification network N_i in Figure 2 may be used for identifying the plant (e.g., an aircraft, a chemical process, or a power system). The network is trained so that an error function is minimized. The controller network N_c must be trained based on minimization of another error function [1–5]. The problem of propagating the controller error for the training of N_c is solved through N_i [1,5]. For a summary, see Figures 3–8 in this report, where performance of networks with dynamic neurons is compared with that of networks with static neurons.

This setup cannot solve all identification problems; however, it is useful for a class of systems with suitable assumptions such as bounded-input bounded-output (BIBO) stable reference and plant system, known upper bound of plant order, etc. [1,3–5]. The derivation of the training algorithm, activation and error functions, and simulation and analysis are all discussed in depth in [1–5]. The parameters to be adjusted include initial weights, and initial system states (the dynamic states of the plant to be identified), and initial states of the nodes. The network structure used for all subsequent simulations consists of one input layer, two hidden layers, and one output layer. In all the simulation studies of single-output (SISO) systems in [1], the weights are initialized as random numbers in $[-1, 1]$. The system to be identified in [1] was assumed to be controllable, observable, and was described by two first-order differential equations. This SISO

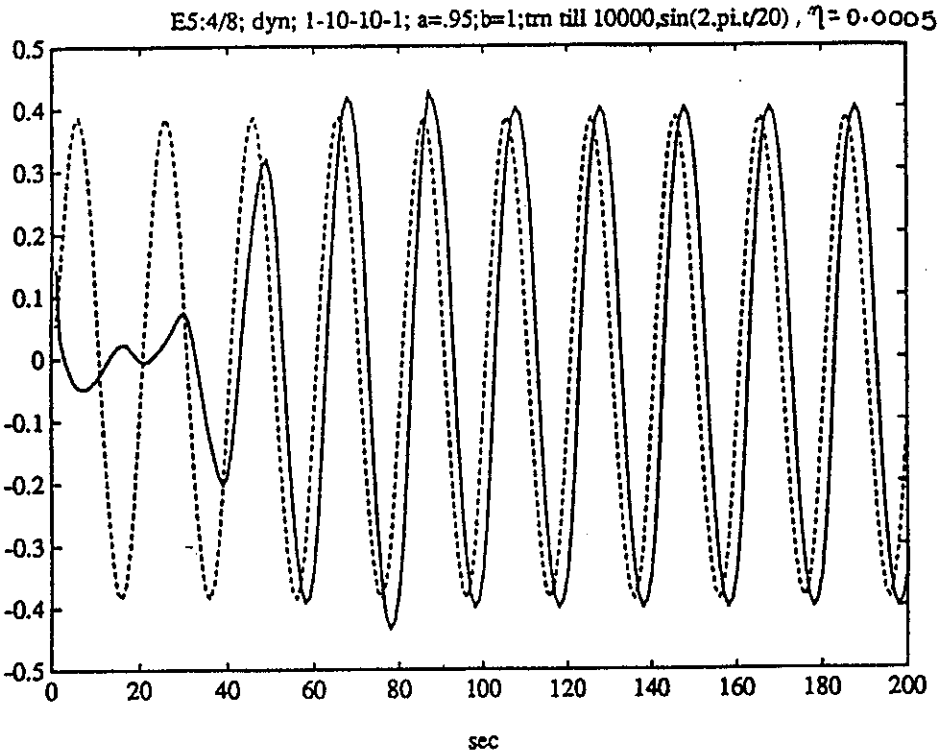


Figure 3. Dynamic nodes, $\eta = 5e^{-4}$, 1 - 10 - 10 - 1 architecture, $a = 0.95$, $b = 1.0$, trained for 100 seconds.

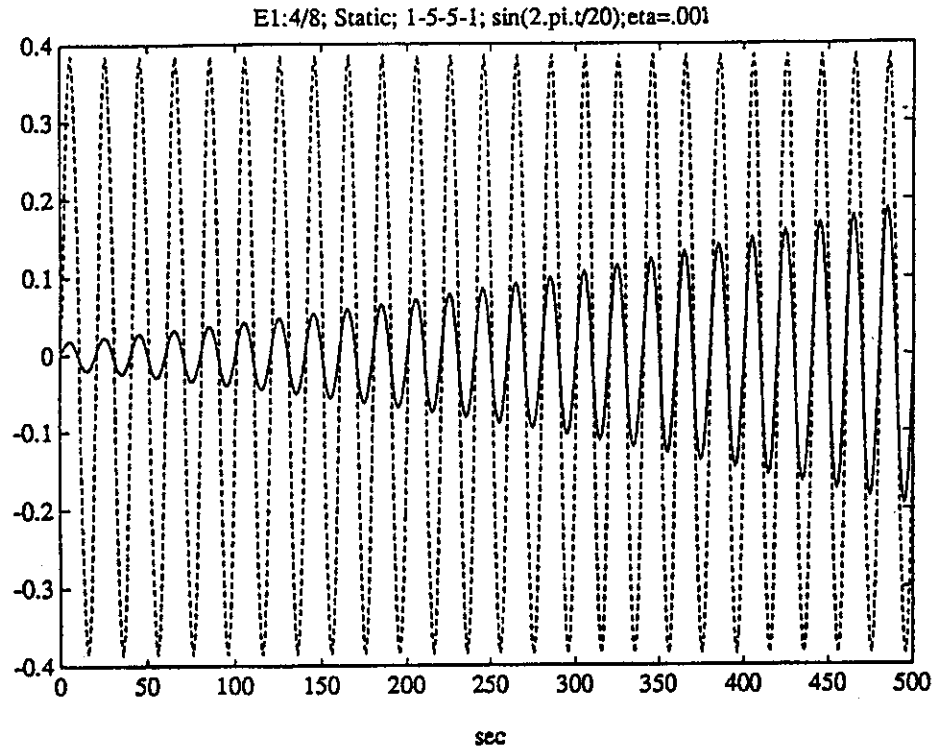


Figure 4. Static nodes, $\eta = 1e^{-3}$, 1 - 5 - 5 - 1 architecture.

system provided insight into the ANN system identification problem. The results showed that learning takes place much faster with dynamic nodes. The choice of appropriate error function also speeds up the convergence.

The simulator of the ANN used in [1] is very powerful and can easily be extended to multi-input and multi-output (MIMO) systems and controller networks. All simulation results in [1] were given for SISO system identification; however, setting up the network for MIMO system identification would require extensive performance analysis.

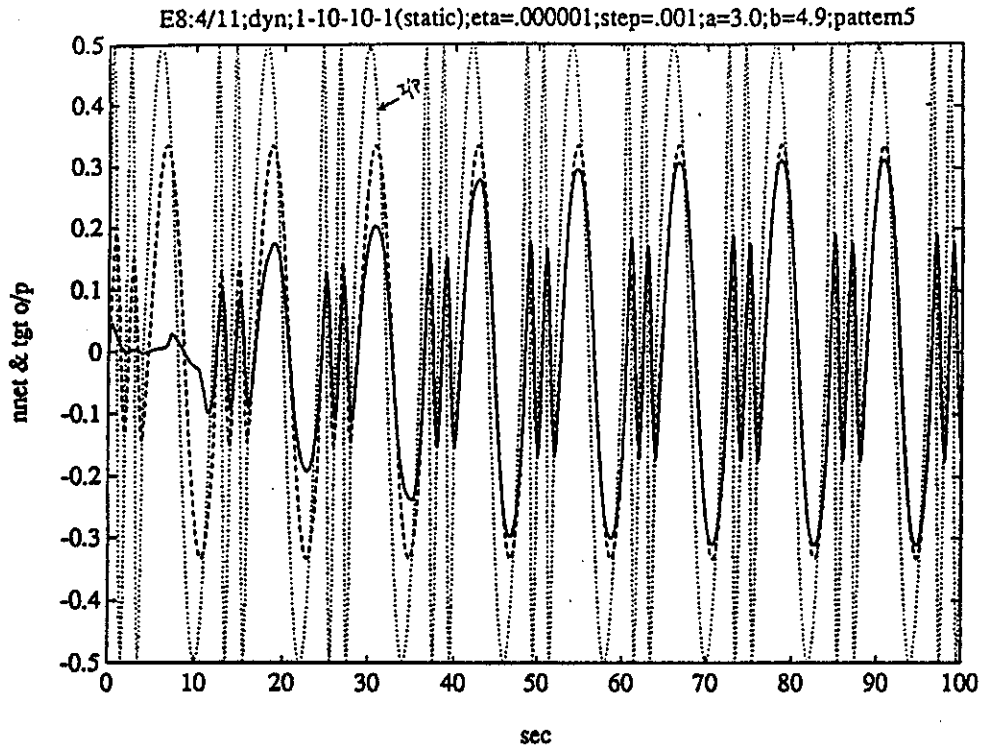


Figure 5. Dynamic nodes, $\eta = 1e^{-6}$, 1 - 10 - 10 - 1, (static output) $a = 3.0$, $b = 4.9$.

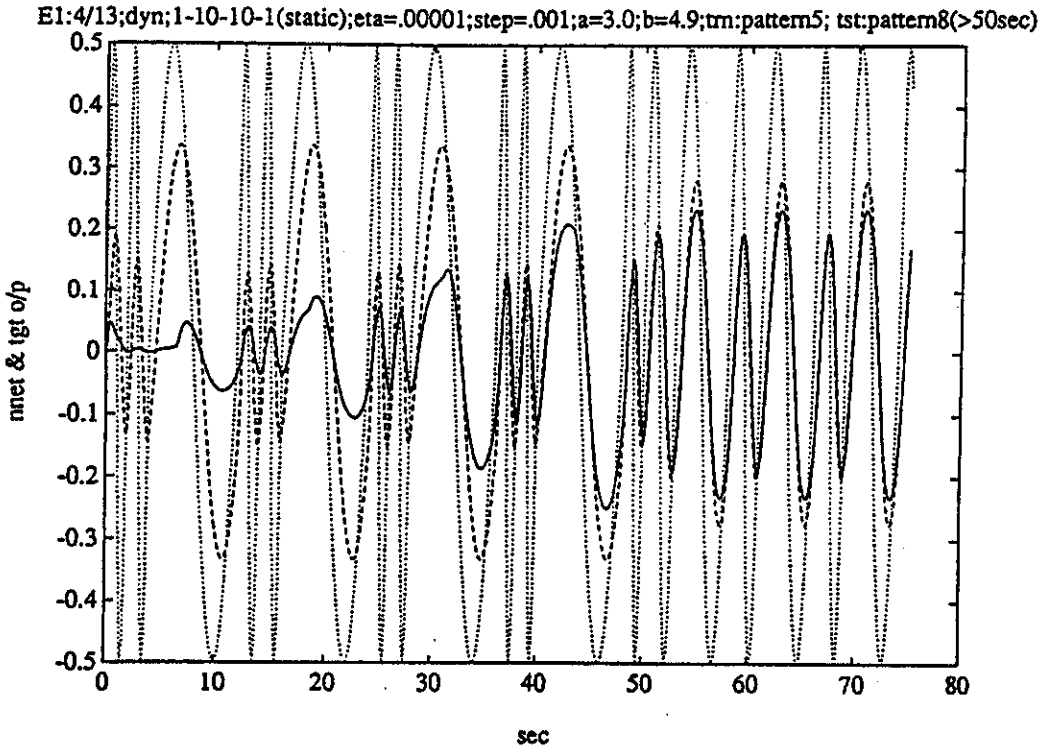


Figure 6. Dynamic nodes, $\eta = 1e^{-5}$, 1 - 10 - 10 - 1 (static output), $a = 3.0$, $b = 4.9$, trained for 50 seconds and tested with new input.

4. DISTURBANCE ATTENUATION VIA NETWORKS WITH LOCAL MEMORY NEURONS

The history of adaptive control spans at least 39 years and over a thousand publications. Our problem may also be classified as one of robust adaptive control. The reader is referred to the tutorials and texts for the subject of robust adaptive control in the presence of bounded as well as state-dependent perturbations [19,23–26]. It is noteworthy to mention that the performance of adaptive controllers improves when disturbances change slowly, because the short-term identifi-

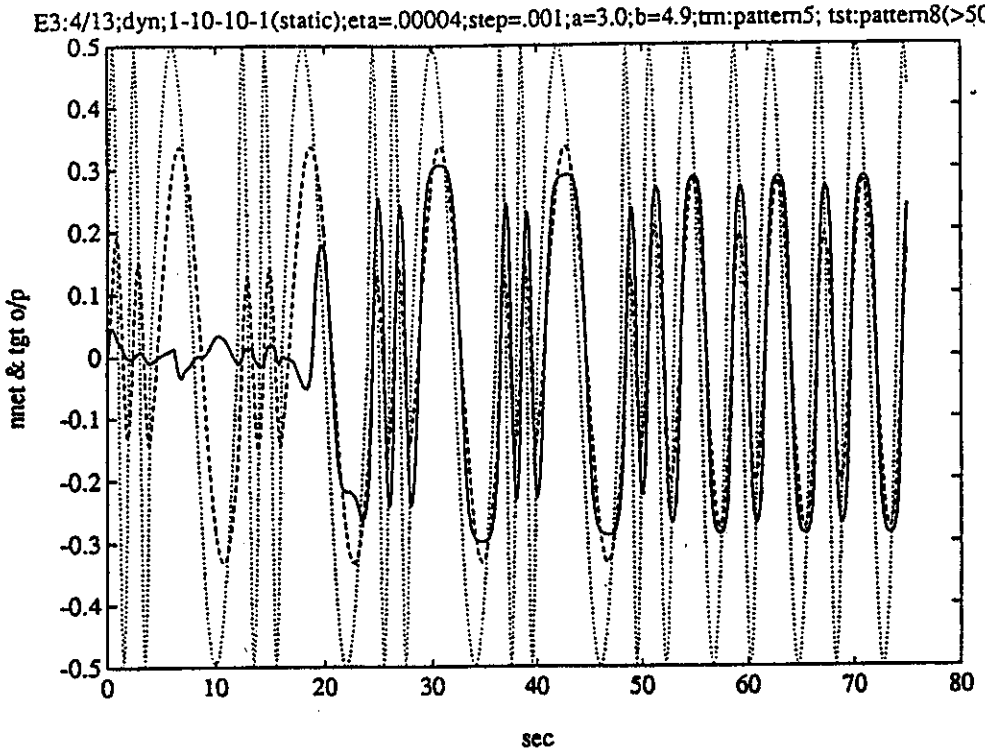


Figure 7. Dynamic nodes, $\eta = 4e^{-5}$, 1 - 10 - 10 - 1 (static output), $a = 3.0$, $b = 4.9$, trained for 50 seconds and tested with new input.

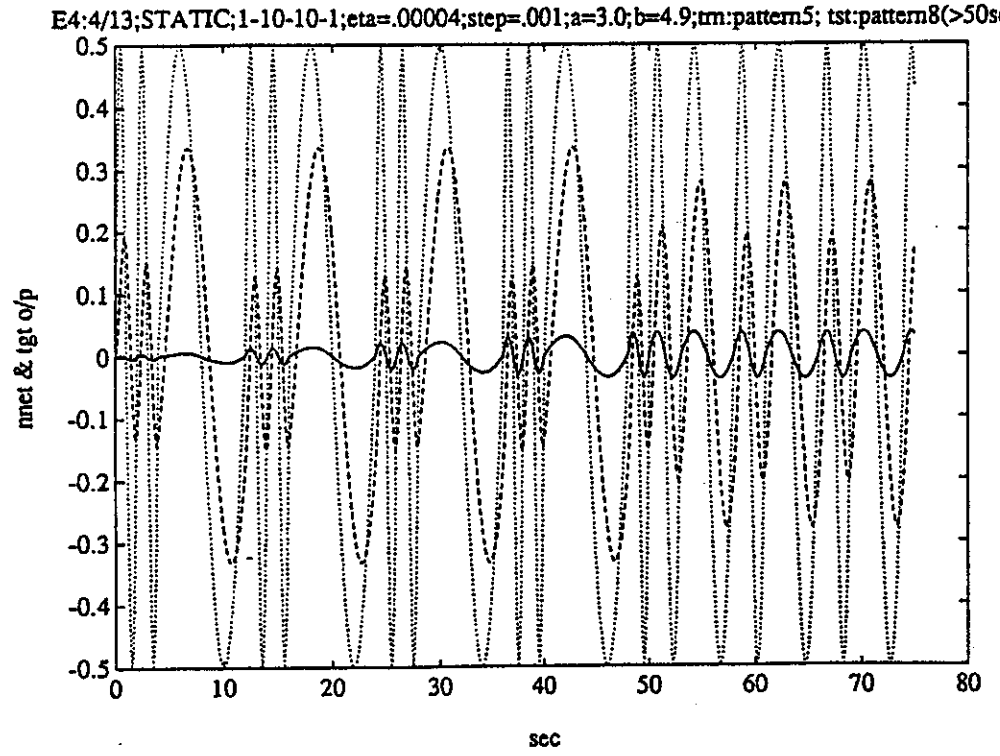
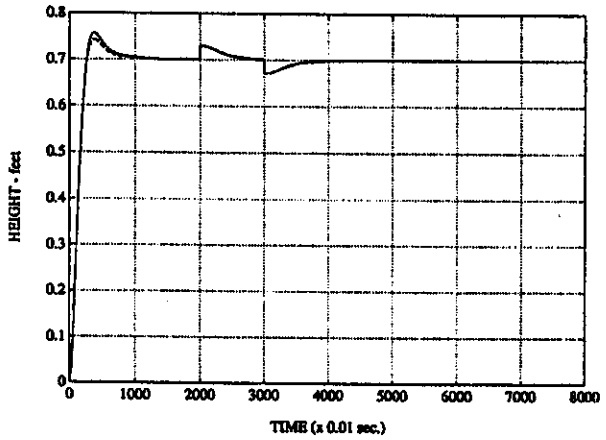


Figure 8. Static nodes, $\eta = 4e^{-5}$, 1 - 10 - 10 - 1, trained for 50 seconds and tested with new input.

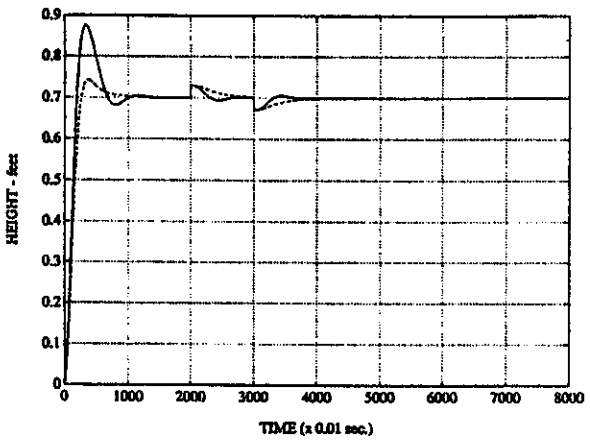
cation and prediction become possible. Most adaptive schemes also assume that an upper bound on the order to the unknown plant is known or assume prior knowledge which is needed for an observer-based control law [25]. Previous work dealing with dynamic neural networks includes that of Narendra and Parthasarathy [22], who used dynamic units, defining partial derivatives of the output error with respect to weights, assuming small changes in network weights over large time intervals, and therefore assuming the weights are constant so that a partial derivative can be defined. Additional references include [10,11,17].

Studies [2–5] consider neurocontrollers consisting of memory neurons for the purpose of real-time control problems. Update rules are provided to adjust the interconnections in the network. The dynamics of the memory neurons are subject to change; the resulting changes in system behavior were investigated. Simulation results show that neurocontroller performance varies with different desired output set-points. Furthermore, the dynamic neurocontroller containing memory neurons exhibits a greater step disturbance rejection capability than a static neurocontroller. The derivation of the updating rules, as well as a discussion of simulation results are given in [2].

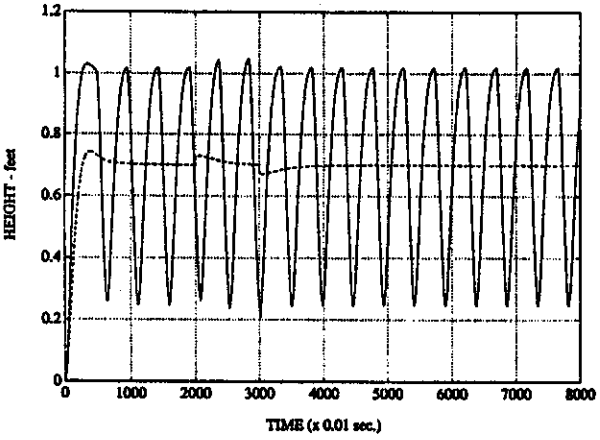
From the simulation results outlined in [2] and summarized in Figure 9 of this report, we can see that incorporating dynamics into the controller improves system performance under certain circumstances: when a high set-point is used, the controller with dynamics performs more or less similarly to the one without dynamics. However, with the same set-point and in the presence of step disturbance, the controller with dynamic neurons performs better than one without dynamics. The system converges more rapidly (the output converges to the setpoint faster) with the dynamic neurocontroller than with the static one. Thus, the dynamic neurocontroller has a better disturbance rejection property. The values of a_i greatly affect the system performance, which can be seen in the simulation results in [2–4]. When a_i has a high value ($a_i = 0.95$), the system response oscillates and performance declines; the choice parameters for better performance are discussed in [2–5]. In conclusion, for a high set-point value, the controller should have the lowest possible value of a_i. However, for a low setpoint value, the system should have a high value of a_i. For more information, please see references [2–5].



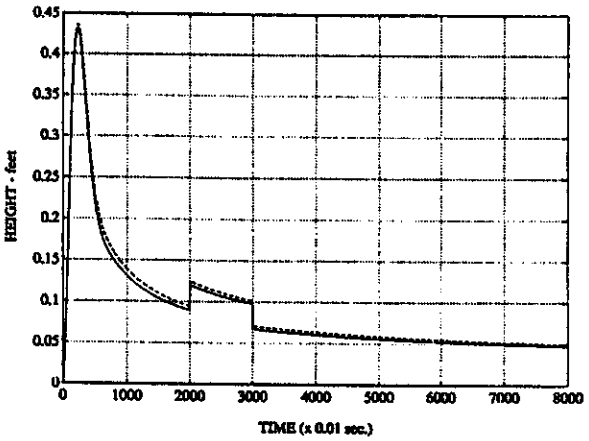
(a) Time (x 0.01 sec.). $y_r = 0.7$, $y_d = 0.03$, Solid: $a = 0.05$, Dash: $a = 0.0$.



(b) Time (x 0.01 sec.). $y_r = 0.7$, $y_d = 0.03$, Solid: $a = 0.4$, Dash: $a = 0.0$.

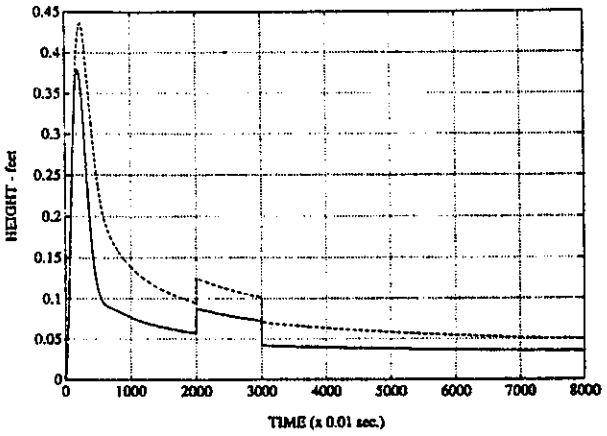


(c) Time (x 0.01 sec.). $y_r = 0.7$, $y_d = 0.03$, Solid: $a = 0.95$, Dash: $a = 0.0$.

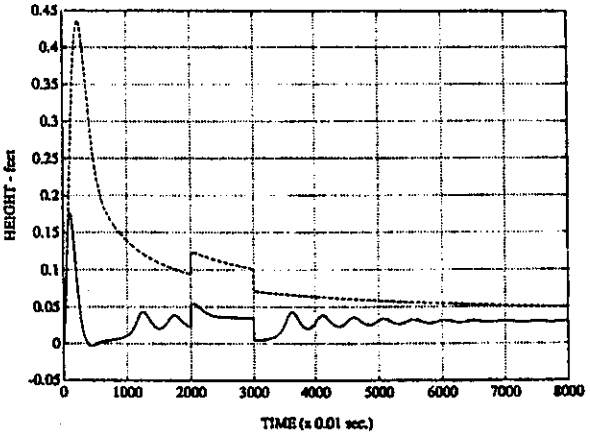


(d) Time (x 0.01 sec.). $y_r = 0.03$, $y_d = 0.03$, Solid: $a = 0.0$, Dash: $a = 0.05$.

Figure 9. Simulation results for dynamic/static neurocontrollers.



(e) Time (x 0.01 sec.). $y_r = 0.03$, $y_d = 0.03$, Solid: $a = 0.4$, Dash: $a = 0.0$.



(f) Time (x 0.01 sec.). $y_r = 0.03$, $y_d = 0.03$, Solid: $a = 0.95$, Dash: $a = 0.0$.

Figure 9. (cont.)

5. AIRCRAFT NEUROCONTROLLER DESIGN IN THE PRESENCE OF WIND SHEAR AFTER TAKE OFF

For this problem, we utilize a differential game formulation with a neural network. The "adversary" is represented by the wind shear and our task was to minimize its effects on the aircraft. We successfully utilized the above methodology, including the same updating rule, in the design of a neurocontroller for a Boeing 727 aircraft exposed to wind shear [3].

Comparison of results showed that the dynamic neurocontroller performed well in the presence of wind shear. Under the same conditions, the neurocontroller worked better than control laws developed by Miele *et al.* [36] and by Leitmann and Pandey [37] (Figures 10 (a–c)). In order to set a control standard for performance testing, we incorporate the same assumptions as Miele and Leitmann [36].

(1) The rotational inertia of the aircraft and the sensor and actuator dynamics are neglected.
(2) The aircraft mass is constant.
(3) Air density is constant.
(4) Flight is in the vertical plane.
(5) Maximum thrust is used.

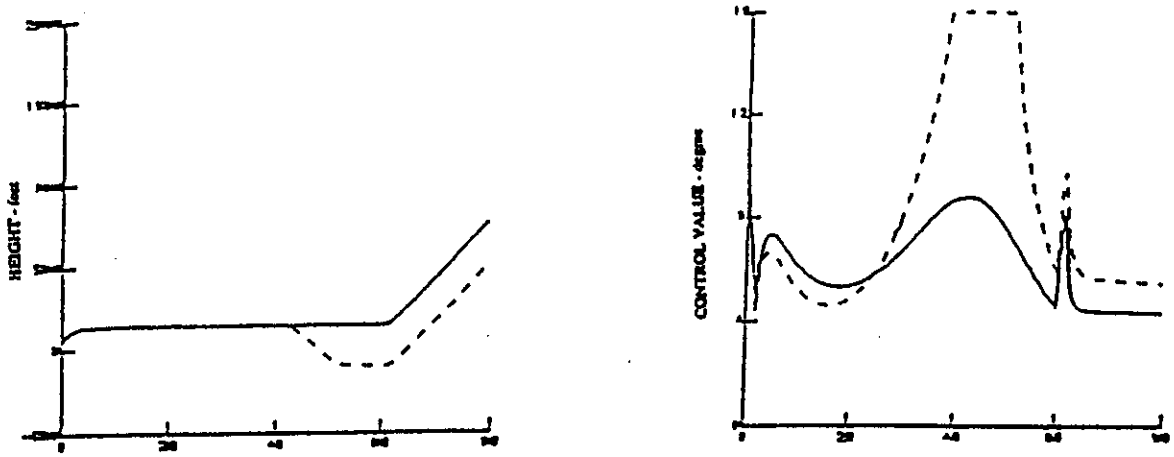
Following Miele's lead, we employ equations of motion for the center of mass of the aircraft in which the kinematic variables are relative to the ground while the dynamic ones are taken relative to a moving, but nonrotating, frame translating with the wind velocity at the aircraft's center of mass. The kinematic equations are

$$\dot{x} = V\cos(\gamma) + W_x,$$
$$\dot{h} = V\sin(\gamma) + W_h.$$

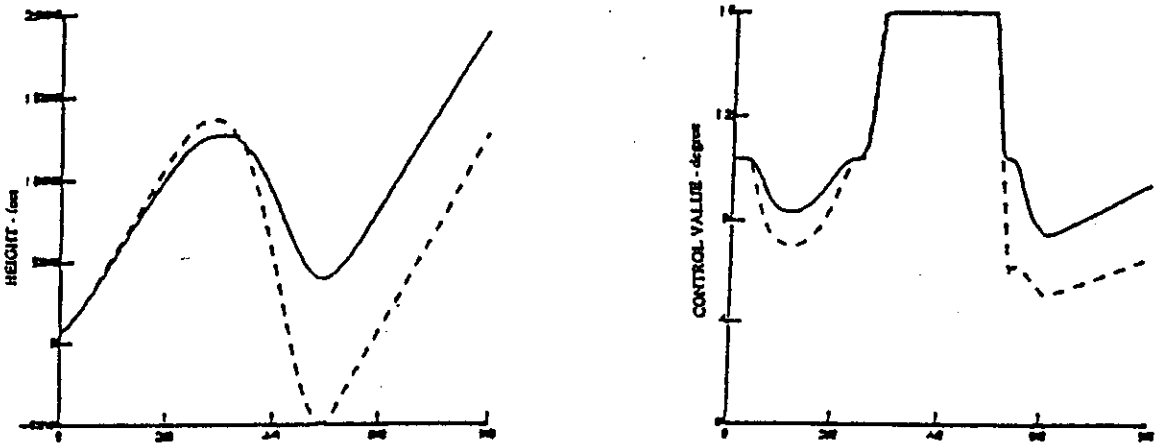
The dynamical equations are

$$m\dot{V} = T\cos(\alpha+\delta) - D - mg\sin\gamma - m(\dot{W}_x\cos\gamma + \dot{W}_h\sin\gamma), \tag{5.1}$$
$$mV\dot{\gamma} = T\sin(\alpha+\delta) + L - mg\cos\gamma + m(\dot{W}_x\sin\gamma - \dot{W}_h\cos\gamma), \tag{5.2}$$

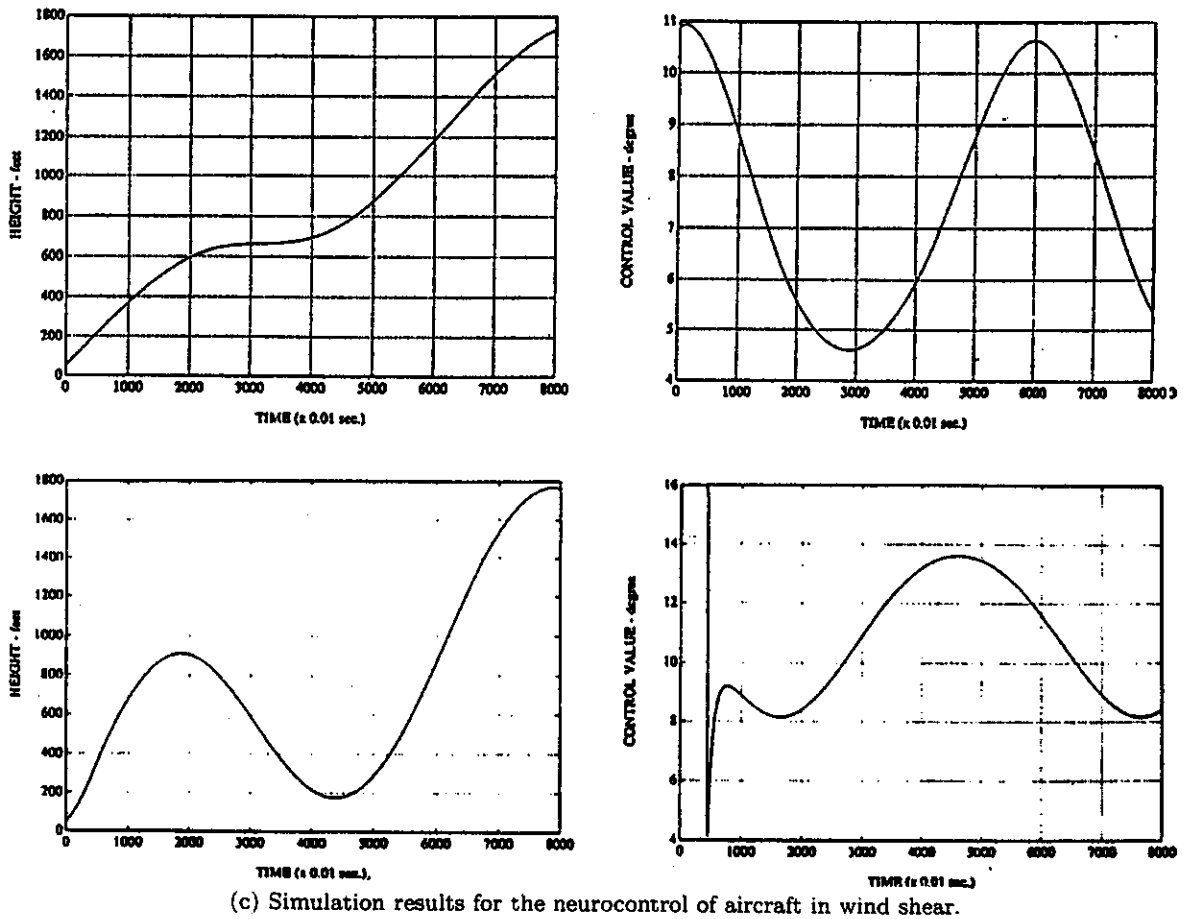
where $T = T(V)$ is the thrust force, $D = D(h, V, \alpha)$ is the drag, $L = L(h, V, \alpha)$ is the lift, and $W_x = W_x(x, h)$ and $W_h = W_h(x, h)$ are the horizontal and vertical wind shears, respectively. In these equations, $x(t), h(t), V(x), \gamma(t)$ are the state variables and the angle of attack $\alpha(t)$ is the control variable.



(a) Miele *et al.*'s controller [36].



(b) Leitmann's controller [37].



(c) Simulation results for the neurocontrol of aircraft in wind shear.

Figure 10.

A discrete-time version of equations (5.1) and (5.2) is given by

$$V_{k+1} = f_1(V_k, T_k, D_k, \gamma_k, \dot{W}_{xk}, \dot{W}_{hk}, \alpha_k)$$
$$= V_k + \frac{T_k \cos(\alpha_k + \delta)\Delta t}{m} - \frac{D_k \Delta t}{m} - g \sin\gamma_k \Delta t - (\dot{W}_{xk} \cos\gamma_k + \dot{W}_{hk} \sin\gamma_k)\Delta t,$$

$$\gamma_{k+1} = f_2(V_k, T_k, L_k, \gamma_k, \dot{W}_{xk}, \dot{W}_{hk}, \alpha_k)$$
$$= \gamma_k + \frac{T_k \sin(\alpha_k + \delta)\Delta t}{mV_k} + \frac{L_k \Delta t}{mV_k} - \frac{g \cos\gamma_k \Delta t}{mV_k} + \frac{(\dot{W}_{xk} \sin\gamma_k - \dot{W}_{hk} \cos\gamma_k)\Delta t}{V_k}.$$

NOTATION.

D	$\stackrel{\text{def}}{=}$	drag force, lb.;
g	$\stackrel{\text{def}}{=}$	gravitational force per unit mass, ft. sec.$^{-2}$;
h	$\stackrel{\text{def}}{=}$	vertical coordinate of aircraft center of mass (altitude), ft.;
L	$\stackrel{\text{def}}{=}$	lift force, lb.;
m	$\stackrel{\text{def}}{=}$	aircraft mass, lb. ft.$^{-1}$ sec.^{2};
O	$\stackrel{\text{def}}{=}$	mass center of aircraft;
S	$\stackrel{\text{def}}{=}$	reference surface, ft.^{2};
t	$\stackrel{\text{def}}{=}$	time, sec.;
T	$\stackrel{\text{def}}{=}$	thrust force, lb.;
V	$\stackrel{\text{def}}{=}$	aircraft speed relative to wind base reference frame, ft. sec.$^{-1}$;
W_x	$\stackrel{\text{def}}{=}$	horizontal component of wind velocity, ft. sec.$^{-1}$;
W_h	$\stackrel{\text{def}}{=}$	vertical component of wind velocity, ft. sec.$^{-1}$;
x	$\stackrel{\text{def}}{=}$	horizontal coordinate of aircraft center of mass, ft.;
α	$\stackrel{\text{def}}{=}$	relative angle of attack, rad;
γ	$\stackrel{\text{def}}{=}$	relative path inclination, rad;
δ	$\stackrel{\text{def}}{=}$	thrust inclination, rad;
ρ	$\stackrel{\text{def}}{=}$	air density, lb. ft.^{2} sec.^{2};

where the notations $\dot{W}_{xk}$, $\dot{W}_{hk}$ denote the variables $\dot{W}_x(t)$, $\dot{W}_h(t)$ at the time $t = k$. Our goal is to design a controller $\alpha = \alpha_k$ such that the quantity $[h_{k+1} - \hat{h}_r]^2$ is minimized, where h_{k+1} is a value calculated from V_{k+1} and α_{k+1} and $\hat{h}_r$ is a given value for the desired constant rate of climb. The cost function is given by

$$J(k+1) = \left(V_{k+1} \sin\gamma_{k+1} - \hat{h}_r\right)^2.$$

The neurocontroller is designed so that the weights update at each step to minimize $J(k+1)$.

Wind Shear Model

Much effort has gone into modeling and identifying wind shear, e.g., [38,39]. In this work, we utilize the wind shear model [37] described by the following equations:

$$W_x = -W_{x0} \sin\left(\frac{2\pi t}{T_0}\right),$$

$$W_h = -W_{h0} \frac{\left[1 - \cos\left(\frac{2\pi t}{T_0}\right)\right]}{2},$$

where W_{x0} and W_{h0} are given constants, reflecting the wind shear intensity, and T_0 is the total flight time through the downburst.

Bounded Quantities

In order to account for aircraft capabilities, it is assumed that there is a maximum attainable value of the relative angle of attack α; that is, $\alpha \in [0, \alpha_*]$, where $\alpha_* > 0$. The range of practical values of the relative aircraft speed V is also limited; that is,

$$\underline{V} \leq V \leq \bar{V}.$$

Force Terms

The thrust, drag, and lift force terms can be approximated

$$T = A_0 + A_1 V + A_2 V^2,$$
$$D = \frac{1}{2} C_D \rho S V^2,$$
$$L = \frac{1}{2} C_L \rho S V^2,$$

where $C_D = B_0 + B_1 \alpha^2$, $C_l = C_0 + C_1 \alpha$. The coefficients A_0, A_1, A_2 depend on the altitude of the runway, the ambient temperature, and the engine power setting. B_0, B_1, C_0, C_1, on the other hand, depend on the flap setting and the undercarriage position.

Controller Design

We employ a neurocontroller with one input layer of four neurons. The input variables to the network are $V, \dot{V}, \gamma, \dot{\gamma}$. The control output is given by $\alpha_k = \phi(w_1(V - V(0)) + w_2 \dot{V} + w_3(\gamma - \gamma(0)) + w_4 \dot{\gamma})$, where the initial values $V(0)$ and $\gamma(0)$ will be given in the next subsection. The threshold function ϕ is the sigmoidal function $T(x) = A(1/(1 + e^{-gx}))$, where g is a design gain and A is the saturation limit. In our study, $A = \alpha$. The formula for updating the weights is a gradient-based style

$$w_i(k+1) = w_i(k) - \alpha_i \frac{\partial J(k+1)}{\partial w_i(k)},$$

where $\frac{\partial J(k+1)}{\partial w_i(k)}$ is given by the following set of equations:

$$\frac{\partial J(k+1)}{\partial w_i(k)} = \frac{\partial J(k+1)}{\partial \alpha_k} \frac{\partial_{\phi_k}}{\partial w_i(k)},$$
$$\frac{\partial J(k+1)}{\partial \alpha_k} = \frac{\partial J(k+1)}{\partial \gamma_{k+1}} \frac{df_2(\cdot)}{d\alpha_k} + \frac{\partial J(k+1)}{\partial V_{k+1}} \frac{df_1(\cdot)}{d\alpha k},$$
$$\frac{\partial J(k+1)}{\partial \gamma_{k+1}} = 2(V_{k+1} \sin \gamma_{k+1} - \hat{h}_r) V_{k+1} \cos \gamma_{k+1},$$
$$\frac{\partial J(k+1)}{\partial V_{k+1}} = 2(V_{k+1} \sin \gamma_{k+1} - \hat{h}_r) \sin \gamma_{k+1},$$
$$\frac{df_2(\cdot)}{\alpha_k} = \frac{\partial f_2(\cdot)}{\partial \alpha_k} + \frac{\partial f_2(\cdot)}{\partial L_k} \frac{L_k}{\alpha_k},$$
$$\frac{\partial f_1(\cdot)}{\partial D_k} = -\frac{\Delta t}{m},$$
$$\frac{\partial f_2(\cdot)}{\partial \alpha_k} = \frac{T_k \Delta t}{m V_k} \cos(\alpha_k + \delta),$$
$$\frac{\partial f_2(\cdot)}{\partial L_k} = \frac{\Delta t}{m V_k},$$
$$\frac{\partial L_k}{\partial \alpha_k} = \frac{1}{2} C_1 \rho S V_k^2,$$
$$\frac{df_1(\cdot)}{d\alpha_k} = \frac{\partial f_1(\cdot)}{\partial \alpha_k} + \frac{\partial f_1(\cdot)}{\partial D_k} \frac{\partial D_k}{\partial \alpha_k},$$
$$\frac{\partial f_1(\cdot)}{\partial \alpha_k} = -\frac{T_k \Delta t}{m} \sin(\alpha_k + \delta),$$
$$\frac{\partial D_k}{\partial \alpha_k} = \frac{1}{2} B_1 \rho S V_k^2.$$

Numerical Data

As a specific model, we use a model for a Boeing 727 aircraft with JT8D-17 turbofan engines. We assume that the aircraft has been airborne from a runway located at sea level. The data are identical to those of Miele:

$$\alpha_* = 16^\circ, \qquad B_1 = 0.6266795,$$

$$C = 3^\circ/\text{sec.}, \qquad C_0 = 0.2624993,$$

$$A_0 = 44564.0 \text{ lb.}, \qquad C_1 = 5.3714832,$$

$$A_1 = -23.98 \text{ lb. ft.}^{-1} \text{ sec.}, \qquad mg = 180000 \text{ lb.},$$

$$A_2 = 0.01442 \text{ lb. ft.}^{-1} \text{ sec.}, \qquad \underline{V} = 184 \text{ ft. sec.}^{-1},$$

$$\delta = 2^\circ, \qquad \bar{V} = 422 \text{ ft. sec.}^{-1},$$

$$\rho = 0.002203 \text{ lb. ft.}^{-4} \text{ sec.}^2, \qquad \Delta t = 0.001 \text{ sec.},$$

$$S = 1560 \text{ ft.}^2, \qquad \hat{h}_r = 33.6807 \text{ ft. sec.}^{-1}$$

$$B_0 = 0.0218747,$$

while the initial conditions are $x(0) = 0$ ft., $h(0) = 50$ ft., and $V(0) = 276.8$ ft./sec., $\gamma(0) = 6.989^\circ$.

Simulation Results

Numerical simulations were carried out for

(i) Flight in the absence of wind shear using different gains.
(ii) Flight with wind shear, again using different gains.

For comparison, we also include the simulation results under the same conditions using the controller proposed by Leitmann, and that of Miele's simplified game guidance (Figures 10a, 10b).

Comparison of the results shows that the neurocontroller performs well in the presence of wind shear. Under the same conditions, the neurocontroller works better than those of Leitmann and Miele. On the other hand, for a more intense wind shear, e.g., $W_{x0}/W_{h0} = 80/48$, we need to adjust the four weights accordingly, since otherwise the controller will not perform as well. Using the neural controller, the situation depends on the sensitivity of the weight updating to the gradient of the cost function with respect to weights. With a suitable choice of learning rates α_1, α_2, α_3, α_4, the situation should be further improved.

6. DISCUSSION

Our objective has been to mathematically formulate the control systems inside the neural networks. We can easily represent each linear SISO system in the neural network by introducing a small feedback loop inside each neuron, rather than a feedback connection. For this paradigm of neural networks, we can directly use the internal states to construct a feedback control law. More

importantly, a network of this type is itself a system, but not an unknown "Black Box." Thus, its input-output performance can be studied just as in the case of the classical control system. Based on this observation, many conventional synthesis methods can be directly borrowed to design the system. The stationary property of the system can be preassigned by means of learning, a unique feature that the classical control system lacks.

As stated by Williams in [17]: "While much of the recent emphasis in the field has been on a multilayer network having no feedback connections, it is likely that the use of recurrently connected networks will be of particular importance for applications to the control of dynamical systems." Indeed, because of the incorporation of feedback and/or dynamics inside the networks, the recurrent and dynamic networks show great promise for the future of neurocontrol research. The property that the Hopfield net has a Content Addressable Memory provides a way to implement many practical problems; e.g., traveling salesman problems. Another particular type of recurrent network, a *settling network*, has also been widely recognized as important in connectionist research. Such a network converges to a stable state from any starting state. The final state of such a network can be viewed as the solution to a certain constraint-satisfaction-type of search, as in *relaxation labeling*, or it might be viewed as a retrieved item from a content-addressable-associative memory. Despite this, the ambiguity of information stored in networks hinders the networks' direct use of the information, and thus there is very limited use for this type of network alone for control purposes.

Our results only constitute a first step in this direction of research. Many interesting open problems still exist, such as:

1. Designing a controller which is also a neural network of the same structure, and then utilizing the controller in the system modeled by the neural networks discussed in this paper. It would be interesting to study this type of hybrid network and to explore its properties as well as applications to the control of a damaged aircraft.
2. Carrying out research in the case of multivariable systems. It is chiefly straightforward to extend current results to a multi-input and multi-output system. However, extension of the results of linearizability is not trivial and requires further study.
3. Considering how to construct the training set. By the Separation Principle of Learning and Control, the systems of this paper can be regarded as static networks when the dynamic parameters are set to zero. Thus, they have the capacity to learn. Also, by this same principle, it is not difficult to show that it is possible to construct a training set such that after the network has learned, it also has the desired stationary properties. Thus, the problem of how to construct a training set so that the trained system has the desired stationary property needs to be investigated.
4. Applying the results of our research to the differential game problems encountered in pursuit-evasion games. Differential game problems can be modeled by an NNLM, and a control strategy for each player can be obtained using various controller design techniques. In this case, the NNLM used for modeling the differential game should have at least two inputs, since differential games have at least two players. This application seems interesting and is probably worthy of further study.

It may be appropriate to point out that our work shows how such nonlinear neural systems can be locally linearized, and thus how various synthesis methods for nonlinear systems can be employed to design a neural control law. Two steps are usually employed to study a nonlinear system [40–43].

STEP 1. Locally linearize a nonlinear system.

STEP 2. Design a control law for the resulting linear system.

Significantly, an NNLM can be viewed in two ways: as a neural network, and as a system to be controlled. If an NNLM is considered a neural network, it has the capacity to learn. As discussed in [44], such types of neural networks can learn the steady state properties of a system. A

backpropagation learning algorithm is used to realize such a purpose. Incorporating the learning capacity of a neural network into a control system is another important contribution of this paper. On the other hand, by incorporating dynamics into a conventional neural network, we can view such types of neural networks as systems to be controlled. The dynamics are described by a set of nonlinear difference equations. Together with [2,5], our work has revealed many interesting properties of an NNLM from this point of view.

REFERENCES

1. P.K. De, S.M. Amin and E.Y. Rodin, System identification with dynamic networks, In *Intelligent Engineering Systems through ANN,* Vol. 2, (Edited by Dagli, Burke & Shin), pp. 97–102, ASME Press, New York, (1992).
2. Y. Wu, S.M. Amin and E.Y. Rodin, Control and disturbance rejection with a dynamic neurocontroller, In *Intelligent Engineering Systems through ANN,* Vol. 2, (Edited by Dagli, Burke & Shin), pp. 643–648, ASME Press, New York, (1992).
3. E.Y. Rodin and Y. Wu, On a learning algorithm of feedback control for an aircraft in wind shear: A case study, In *Proc. of the 31st IEEE Conference on Decision and Control,* Tucson, AZ, (December 1992).
4. E.Y. Rodin and Y. Wu, On neural networks with local memory for control systems, *Appl. Math. Lett.* **4** (5), 97–101 (1991).
5. Y. Wu and E.Y. Rodin, Control systems in neural networks with local memory, *Computers Math. Applic.* **26** (2), 1–24 (1993).
6. O. Farotimi, A Dembo and T. Kailath, Absolute stability and optimal training for dynamic neural networks, In *23rd Annual Asilomar Conference on Circuits, Systems and Computers,* Pacific Gove, CA, October 1989, Vol. 1, pp. 133–137.
7. M. Gori, Y. Benjio and R. Demori, BPS: A learning algorithm for capturing the dynamic nature of speech, In *IJCNN,* Washington, DC, June 1989, pp. 417–423.
8. M. Gherrity, Learning algorithms for analog, fully recurrent neural networks, In *IJCNN,* Washington, DC, 1989, pp. 643–644.
9. M.J. Willis, C. DiMassimo, G.A. Montague, M.T. Tham and A.J. Morris, Artificial neural networks in process engineering, In *IEEE Proceedings, Part D: Control Theory and Applications,* Vol. 138, No. 3, pp. 256–266, (May 1991).
10. R. Perfetti, Effect of feedback in dynamic neural networks, *Electronics Letters* **27** (15), 1367–1369 (July 1991).
11. S.I. Sudharsanan and M.K. Sundareshan, Systematic design of associative memory networks: Equilibrium confinement, exponential stability and gradient descent, In *IJCNN,* June 1990, pp. 755–762.
12. M. Sato, K. Joe and T. Hirahraa, APOLONN brings us to the real world: Learning nonlinear dynamics and fluctuations in nature, In *IJCNN,* San Diego, CA, June 1990, pp. 581–587.
13. S. Tan, L. Vandenberghe and J. Vandewalle, Remarks on the stability of asymmetric dynamical neural networks, In *IJCNN* San Diego, CA, June 1990, pp. 461–467.
14. T. Mckelvey, Neural networks applied to optimal flight control, In *Proceedings of IFAC Symp. on AI in Real-Time Control,* (Selected papers from the IFAC/IFIP/IMACS Symposium), (Edited by H.B. Verbruggen and M.G. Rodd), pp. 19–23, Pergamon Press, Oxford, UK, (1992).
15. P.J. Werbos, Neurocontrol and elastic fuzzy logic: Capabilities, concept, and applications, *IEEE Trans. on Industrial Electronics* **40** (2), 170–180 (April 1993).
16. T. Yamaguchi *et al.*, Intelligent control of a flying vehicle using fuzzy associative memory system, In *IEEE International Conference on Fuzzy Systems,* San Diego, March 1992, pp. 1139–1149.
17. W.T. Miller III, R.S. Sutton and P.J. Werbos, *Neural Networks for Control,* MIT Press, (1990).
18. K.J. Hunt, D. Sbarbaro, R. Zbikowski and P.J. Gawthrop, Neural networks for control systems—A survey, *Automatica* **28** (6), 1083–1111 (November 1992).
19. L. Ljung, *System Identification—Theory for the User,* Prentice Hall, Englewood Cliffs, NJ, (1987).
20. P.D. Wasserman, *Advanced Methods in Neural Computing,* Van Nostrand Reinhold, New York, (1993).
21. S.A. Billings *et al.*, Properties of neural networks with applications to modelling nonlinear dynamical systems, *International Journal of Control* **55** (1), 193–224 (1992).
22. K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks* **1** (2), 4–27 (March 1990).
23. K.J. Aström and Björn Witternmark, *Adaptive Control,* Addison-Wesley, (May 1989).
24. K.S. Narendra and A.M. Annaswamy, Robust adaptive control, In *Adaptive and Learning Systems: Theory and Applications* (Edited by K.S. Narendra), pp. 3–33, Plenum Press, New York, (1986).
25. K.J. Aström, Theory and applications of adaptive control—A survey, *Automatica* **19**, 471–481 (1983).
26. G.C. Goodwin and K.S. Sin, *Adaptive Filtering Prediction and Control,* Prentice-Hall, Englewood Cliffs, NJ, (1984).
27. G.A. Rovithakis *et al.*, Adaptive control of unknown plants using dynamical neural networks, *IEEE Trans. on SMC* **24** (3), 400–412 (March 1994).

28. B. Horne, M. Jamshidi and N. Vadiee, Neural networks in robotics: A survey, *J. of Intelligent and Robotic Systems* (3), 51–66 (1990).
29. S. Lee and G.A. Bekey, Application of neural networks to robotics control and dynamic systems, *Control and Dynamic Systems* **39**, 1–69 (1991).
30. M. Nikolaou and V. Hanagaudi, Input-output exact linearization of nonlinear dynamical systems modelled by recurrent neural networks, In *Proceedings of the 1st Conference of Artificial Neural Networks in Engineering,* St. Louis, MO, November 1991, pp. 575–579.
31. M. Nikolaou, Y. You and H. Sarimveis, Process modelling with recurrent neural networks, In *Proceedings of the 1st Conference of Artificial Neural Networks in Engineering,* St. Louis, MO, November 1991, pp. 595–600.
32. T. Troudet, S. Garg, D. Mattern and W. Merrill, Towards practical control design using neural computation, In *Proceedings of the IJCNN,* 1991, Vol. II, pp. 675–681.
33. P. Ioannou *et al.*, Two algorithms for nonlinear system identification using dynamical neural networks, *IEEE ACC,* (1991).
34. B. Fernandes, Nonlinear dynamic system identification using artificial neural networks: The discrete time case, *IEEE CDC* (1990).
35. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, In *Proc. Natl. Acad. Sci.*, Vol. 81, pp. 3088–3092, (1987).
36. A. Miele, T. Wang, C.Y. Tzeng and W.W. Melvin, Optimization and guidance of abort landing trajectories in a wind shear, In *AIAA Guidance, Navigation and Control Conference,* Paper No. AIAA-87-2341, (1987).
37. G. Leitmann and S. Pandey, Aircraft control for flight in an uncertain environment: Takeoff in windshear, In *Conference on Modeling and Control of Uncertain Systems,* September 1990, pp. 283–302.
38. W. Frost and D.W. Camp, Wind shear modeling for aircraft hazard definition, FAA Report, FAA-RD-77 (1977).
39. S. Zhu and B. Etkin, Fluid dynamic model of a downburst, UTLAS Report #271.
40. J.W. Grizzle, Feedback linearization of discrete-time systems: Analysis and optimization of systems, In *Lecture Notes on Control and Information Science,* Vol. 83, pp. 273–281, Springer-Verlag, (1986).
41. H.G. Lee, A. Arapostathis and S.I. Marcus, On the linearization of discrete-time systems, *Int. J. Control* **45**, 1803–1822 (1987).
42. H.G. Lee and S.I. Marcus, Approximate and local linearizability of nonlinear discrete-time systems, *Int. J. Control* **44**, 1103–1124 (1986).
43. H. Nijmeijer and A.J. Vanderschaft, *Nonlinear Dynamical Control Systems*, Springer-Verlag, (1990).
44. J.J. Hopfield, Neural computations of decisions in optimization problems, *Biological Cybernetics* **52**, 14–152 (1985).
45. T. Kohonen, An introduction to neural computing, *Neural Networks* **1** (1), 3–16 (1988).
46. J.J. Hopfield and D. Tank, Computing with neural circuits: A model, *Science* **233**, 625–733 (1986).
47. D. Tank and J.J. Hopfield, Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit, *IEEE Transaction on Circuits and Systems* **33** (5), 533–541 (1986).
48. T. Kailath, *Linear System*, Prentice-Hall, Englewood Cliffs, NJ, (1980).
49. J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proc. Nat. Acad. Sci., U.S.* **79**, 2554–2558 (April 1982).