HYPERSONIC VEHICLE TRAJECTORY OPTIMIZATION AND CONTROL

S. N. Balakrishnan
J. Shen
J. R. Grohs

Dept. of Mechanical & Aerospace Engineering
and Engineering Mechanics
University of Missouri-Rolla
Rolla, MO 65409-0050

FINAL REPORT

July 1997

Grant Number: NAG 1 1728
Hypersonic Vehicles Office
Nasa Langley Research Center
Hampton, VA 23681-0001

TABLE OF CONTENTS

EXECUTIVE SUMMARY 1

BACKGROUND 2

WHY NEURAL NETWORKS 3

LITERATURE REVIEW 4

PROBLEM FORMULATION 18

OPTIMIZATION/CONTROL 20

NUMERICAL RESULTS 30

CONCLUSIONS 32

ACKNOWLEDGMENT 32

BIBLIOGRAPHY 33

FIGURES 37

APPENDIX 52

HYPERSONIC VEHICLE TRAJECTORY OPTIMIZATION AND CONTROL

EXECUTIVE SUMMARY

Two classes of neural networks have been developed for the study of hypersonic vehicle trajectory optimization and control.

The first one is called an 'adaptive critic'. The uniqueness and main features of this approach are that: 1) they need no external training, 2) they allow variability of initial conditions, and 3) they can serve as feedback control. This is used to solve a 'free final time' two-point boundary value problem that maximizes the mass at the rocket burn-out while satisfying the pre-specified burn-out conditions in velocity, flightpath angle, and altitude.

The second neural network is a recurrent network. An interesting feature of this network formulation is that when its inputs are the coefficients of the dynamics and control matrices, the network outputs are the Kalman sequences (with a quadratic cost function); the same network is also used for identifying the coefficients of the dynamics and control matrices. Consequently, we can use it to control a system whose parameters are uncertain.

Numerical results are presented which illustrate the potential of these methods.

I. BACKGROUND

For the United States to maintain its leadership in space technology, cheaper means of space transportation - alternatives to space shuttle must be developed. In order to develop such an alternative, different configurations of hypersonic vehicles must be studied from the perspectives of cost-effective performance. A major part of such study involves optimal trajectory design for its mission and control of vehicles. Since current state of knowledge of hypersonic vehicles (in atmospheric flight, especially) is limited, it is imperative that any tool that is developed for trajectory optimization and control be usable with variations in flight parameters. There are quite a few methods - direct and indirect - available in the existing literature which deal with trajectory optimization and optimal control. However, they are either ill-suited for design or do not consider the design phase of a vehicle. First, for each scenario, typically, a two-point boundary value problem needs to be solved. This process could lead to an enormous amount of time when several combinations of scenarios are considered. Second, many trajectory optimization methods do not directly yield a feedback form of control that can be used in flight.

In this study, two new neural network based approaches have been formulated which address the two problems mentioned. The resulting design technique enables the user to study optimal trajectory of hypersonic vehicles with a set of predetermined neural networks. **For an envelope of scenarios, this approach is expected to yield near optimal trajectories.** We formulate the problem in such a way as to produce a feedback control directly. In the case of recurrent networks, the gains of the matrices used in a linearized control.

II. WHY NEURAL NETWORKS

Use of direct or indirect methods of optimization necessitates having to solve a problem for each set of initial conditions. This requires determining a separate solution for each possible initial condition for a given system. Dynamic programming is also a method of determining optimal control for a family of initial conditions. However, the usual method of solution becomes very difficult to solve in higher dimensions and 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 two-point boundary value problem (TPBVP) allows an approximate solution over a limited 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.

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 with neural networks 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. In the following sections, we trace the development of neural networks and development of learning control in particular.

III. LITERATURE REVIEW

The development of intelligent control system design techniques has a long and rich history as does the field of control systems engineering in general. Neural network techniques have also been used in control systems for quite a long time but recently have become very popular. This section contains a brief survey of the history of control ranging from cybernetics in the 1940's through learning control systems and the beginning of neural control in the 1960's. The next important landmark occurred with the use of critic architectures in reinforcement learning systems. We conclude the section with a brief survey of current literature in neural control organized in the areas of system identification, nonlinear, adaptive, and optimal neural control.

1. Cybernetics, Neural Networks and Learning Control. Norbert Wiener is recognized as the father of cybernetics, a field which he describes as "the control and communication in the animal and in the machine" [1]. Cybernetics also provided some of the motivation for the development of control theory and neural networks during the 1950's and 1960's. For example, Ashby contributed two complementary monographs in cybernetics, *Design for a Brain [2]* and *An Introduction to Cybernetics* [3] which discussed control and communication in biological systems. In the former, Ashby gave an early implementation of an artificial neural network called the hemostat. The latter contribution was a careful development of cybernetics intended to popularize the technology. Topics discussed include feedback, stability, a black box theory for large systems, regulation and control in biological systems, and hierarchical control.

K. S. Fu gives one of the first formal descriptions of learning control in [4]. A learning control system is a control system capable of modifying its behavior based on experience in order to maintain acceptable performance in the presence of uncertainties. Possible measures of performance include the amount of time required to adapt to changes and the evaluation of suitable performance indices. A learning control system is distinguished from adaptive control systems through its ability to recognize familiar patterns in a situation and, based on past experience, to adjust in order to improve performance. Adaptive control systems emphasize a control system's ability to react to new situations.

Sklansky gives an early survey of learning control [5]. According to Sklansky, learning in the automatic control literature is associated with a hierarchical arrangement of three feedback loops. These are the controller, a system identifier or pattern recognizer, and a teacher. The pattern recognizer transforms observable quantities in the system into a fixed set of categories, each of which corresponds to a set of controller parameters. Categories are represented by fixed regions in an intermediate feature space. The teacher provides information to the pattern recognizer for adjusting the boundaries between categories in the feature space so that improved control system performance results. An adaptive control system uses only the first two loops. The learning loop, which distinguishes a learning control system from an adaptive control system, sends reinforcement signals in the form of a reward or a punishment to the pattern recognizer based on an assessment of current control system performance.

The advantage of the use of the learning loop is that it provides a means of training the pattern recognizer on-line. Sklansky describes five techniques for the design of learning control systems and notes their interrelationships and pattern classification. These techniques are decision theory,

trainable threshold logic, hill climbing, sample set construction, and Markov chains. In the decision theoretic approach, the boundaries between classes are determined by estimating joint probability densities using measurements taken from the system during operation. The trainable threshold logic method which Sklansky describes is actually a precursor to the use of neural networks for control. In this method, category boundaries are moved by adjustment of weighted sums of components in a feature vector. this weighted sum is then passed through a threshold function to produce a bipolar control signal. The teacher in a threshold logic learning system provides information for adjusting weights in the categorizer. The sample set construction technique breaks categories into subcategories based on distances measured in the feature space. During training a fixed set of prototype feature vectors are developed with the subcategories given by open balls surrounding the prototype. We then form the category regions as the unions of subcategories. The boundary between categories is formed as a sequence of hyperplanes perpendicular to hyperplanes joining prototypes from each category.

Ideas from decision theory, trainable threshold logic, and sample set construction are prominent in the development of neural network theory. In 1966 Nikolic and Fu [6] describe an algorithm based on decision theory for on-line learning control of an unknown discrete time plant without an external teacher. Control actions are chosen from a finite set. The performance index is the conditional expectation of the instantaneous performance evaluations with respect to observed states and allowable control actions. The model used by Nikolic and Fu is very similar to Sklansky's general learning control system and they include provisions for the case when the teacher does not have perfect knowledge of the plant being controlled. This work provides the foundation for later critic based schemes.

Tsypkin also makes contributions in learning control systems based on decision theory and optimization. In an article about 'self-learning' [7], Tsypkin distinguishes between three methods for determining decision rules in the pattern recognizer. The first method assumes that statistical information is available in advance. In this case statistical decision theory can be used to determine the decision rule. In the second method, the designer assumes that a sequence of correctly classified patterns exists. In this case, the decision rule is determined based on data in the training set and the method is called learning with reinforcement. In the third case, no information is assumed initially and the decision rule is found using observed but unclassified patterns from the system. Tsypkin calls this third case self-learning. Extensions of the idea of self-learning in automatic systems applied to pattern recognition, identification, dual control, and the allocation of resources are discussed in a later work [8] and compiled into a text [9].

The improvement in performance with respect to given performance objectives and based on experience is a common theme in learning control. There are three components related to performance in the control system: 1) the specification optimal performance objectives, 2) the assessment of the system's level of performance, and 3) a means for improving performance over time. Cybernetics and learning control are based on the use of pattern recognition, optimization, and control of uncertain dynamic systems using biologically inspired models of intelligent behavior. Rudimentary neural networks in the form of linear threshold logic units have been used as an implementation medium for learning control systems cited above. We now turn to a discussion of a subclass of learning control systems called reinforcement learning systems which build in methods for assessing and improving control system performance.

2. Learning with a Critic. The ground breaking work on learning control in the 1960's, along with studies in cybernetics, has led to a study of critic-based systems for two decades and this study has recently been revived even in the current decade. In 1970, Mendel and McLaren introduced a concept in learning control which they call reinforcement learning [10]. Reinforcement learning control is developed as a subclass of learning control discussed above with the addition of performance assessment and a method for modifying controller actions. The idea is to provide a means of control for unstructured environments where the plant model may not be known or where a complex performance measure is used [11]. In reinforcement learning systems, a critic is used to monitor plant inputs and outputs and to provide an evaluation signal which represents an indication of current performance to the controller.

Widrow, Gupta and Maitra [12] describe the concept of the critic for adaptation of neural networks. Widrow et al., delineate three separate modes of learning. A supervised learning system, also known as learning with a teacher, modifies the parameters of the neural network using error between network output signals and the desired output signals. The assumption here is that the desired output signals corresponding to each input signal are known at the time that learning is taking place. In an unsupervised learning procedure, also called learning without a teacher or decision-directed learning, the parameter adjustments are not guided by knowledge of a desired output signal. Learning with a critic bridges the gap between the two previous methods. Learning with a critic does not assume that desired output signals are known for each input signal but rather that some indication can be made with respect to network performance over a series of trials.

Barto, Sutton and Anderson [13] extend the idea of learning with a critic through the development of a learning system which includes both an adaptive critic element and an adaptive

search element. As in the learning with a critic approach, explicit desired control actions are unknown. The objective is to provide control signals which tend to optimize a performance index. The purpose of the adaptive search element is to implement a trial-and-error procedure to associate control vectors with respective observations of the state of the system being controlled. The adaptive critic element receives a success/failure signal from an outside source as a result of a series of control actions. This signal is called an external reinforcement signal. The adaptive critic element also receives weighted signals from each of the state variable of the controlled system. The external reinforcement signal provides feedback for modifying the strengths of these connections. The adaptive critic element uses the external reinforcement signal and weighted state signals to provide a continuous evaluation of performance to help guide the search for appropriate control actions. Sutton calls the critic based adaptation algorithm the "Adaptive Heuristic Critic" and develops its application in credit assignment problems in his Ph.D. dissertation [14].

The implementation of the adaptive critic is based on Widrow's method for learning with a critic but provides a higher level of feedback to the control system. Two sets of connection weights connecting two processing elements are adjusted during the learning procedure. This, in conjunction, with the active search distinguishes the adaptive critic architecture from previous work. The adaptive critic architecture is capable of learning to balance a pole mounted on a movable cart by applying control signals to a movable cart with no prior knowledge of the system to be controlled. This ability to determine control actions assuming no previous knowledge is a great strength of the adaptive critic architecture. The disadvantage of the architecture is that many failed trials occur before a successful run is completed. The cart-pole solution also depends on the partitioning of the problem state space into a finite number of regions. This partitioning may not be practical in problems where finer control

is required. In this case the number of regions required may be too large for effective results. Examples of such problems include those with time-varying dynamics, tracking problems, and some nonlinear problems.

Barto et al. [13], distinguish between supervised learning paradigms and reinforcement learning used in the adaptive critic approach. In the supervised learning approach training proceeds in several steps. First an input pattern is presented to a neural network. An output response is produced based on the current parameters embedded within the network. The response is then compared with a desired response and error is used to modify the neural network parameters to improve its mapping. Reinforcement learning is based on an evaluation of the current network output in relationship with current external factors (states in a system for example). This evaluation may be as simple as a binary decision indicating a reward for proper response or punishment for inappropriate response. The quality of feedback for a system using reinforcement learning is lower than that available in a supervised learning system. This property makes reinforcement learning methods useful for situations when a quantitative answer is not available.

Werbos [15] defends the use of neural networks for control applications. He suggests that neural networks will be able to solve difficult problems faced by modern controls engineers including the real-time control of nonlinear possibly unknown systems with high noise levels and high throughput. Werbos describes five dominant paradigms for use in neural control systems. These are Supervised Control, Inverse Dynamics, Stabilization Systems, Backpropagation Through Time and Adaptive Critics with Reinforcement Learning. The Supervised Control architecture uses a neural network trained to map current state vectors to corresponding control vectors. In the inverse dynamics approach, observed system state is assumed to be a function of the current control and

previous system state. The neural network is trained to invert the plant in order to provide control actions which lead to desired states. Stabilization systems are designed to provide stable control in tracking and regulator problems. Backpropagation through time depends on a plant model and a performance index written in terms of control and state actions. The neural network predicts a sequence of states given a sequence of control actions. The backpropagation algorithm then provides derivatives of the performance index which can be used to update control actions at each step along the way. Adaptive critic architectures and reinforcement learning are the focal point in [33]. Werbos describes systems based on the adaptive critic as an approximation to dynamic programming and presents the notion of the backpropagated critic.

Jameson [16] claims to be the first to publish results using a backpropagated critic. The primary difference between the adaptive critic architecture of Barto Sutton, and Anderson and the backpropagated critic is the maximization of the critic output providing gradient information via a plant model network to the controller so that future control actions can be improved. The purpose of the critic network in this architecture is to predict future reinforcement signals from the environment. The critic network and a model of the plant are used to calculate derivatives of the predicted reinforcement signal with respect to control actions. The control actions are then modified to improve performance. The prediction provided by the critic network is also improved by comparing the actual reinforcement signal with previously stored predictions. The backpropagated critic, like previous critic designs, assumes no knowledge of the plant and results are improved by making multiple attempts at a solution.

Sofge and White [17] advocate the development of neural control architectures which can be adapted on-line for stable operation of unknown, nonlinear plants which may include noise in the

feedback loop. They suggest that adaptive critic architectures may be used in manufacturing the process control applications to provide flexibility and efficient adaptability through changes which occur during the life-cycle of equipment. The authors use an adaptive critic architecture based on Albus' CMAC neural network [18] to do process control in a thermoplastic composite manufacturing process. According to Sofge and White, "the goal of on-line learning is the real-time optimization of a large scale non-linear process at minimal computational cost." The authors have designed and built an adaptive critic system for control of manufacturing processes.

Watkins gives a recent implementation of reinforcement learning called Q-learning in his dissertation [19]. Q-learning is based on the approximation of a real valued function, called the Q-function by Watkins. The q-function is a function that maps current plant state and control into an estimation of the future performance of the system. This estimate is based on the assumption that optimal control is applied to the plant from the next time instant forward. A Q-learning algorithm is an algorithm which iteratively improves the estimation for the Q-function. There is a close correspondence between Q-learning and dynamic programming used in the control of dynamical systems [20]. Bradtke [21] distinguishes between two types of Q-learning algorithms. Bradtke calls the form described above an optimizing Q-learning algorithm because it tries to learn the Q-function directly. A slightly modified form called the policy-based Q-learning algorithm tries to learn an optimal sequence of plant control inputs (the control policy).

Many recent control system applications of the ideas of reinforcement learning and adaptive critic architectures exist. Gullapalli describes a reinforcement learning algorithm for learning control. This method uses radial basis functions and the adjustable parameters of the network are means and variances of normal distribution functions. The method is applied to a simulated 3 degree-of-freedom

robotic arm [22]. Stamenkovich uses adaptive critic and adaptive search elements for learning to guide a ship through a channel [23]. Shelton [24] demonstrates an adaptive critic design for controlling a truck with a CMAC (Cerebellar Model Articulated Controller, [18). Tham and Prager compare the adaptive heuristic critic algorithm with the Q-Learning algorithm for obstacle avoidance and control in multi-linked robotic manipulators [25]. Gachet et al., present an adaptive heuristic critic based control system for learning goal based behavior for autonomous robot control. The three types of behavior discussed are: 1) move to a goal state, 2) do surveillance, and 3) follow a specified path [26].

3. Neural Identification and Control. There has been an explosion of reported research in the use of neural networks in control systems in recent years. Bavarian [27] gives an introduction to the use of neural networks for intelligent control. Several monographs have been compiled including a well known work edited by Miller, Sutton, and Werbos [28]. White and Sofge have compiled a book which includes several chapters dealing with the use of neural networks in intelligent control systems [29]. Hunt et al., have produced a comprehensive survey of the field [30].

Psaltis, Sideris and Yamamura describe three possible architectures for neural control systems [31]. The indirect learning architecture attempts to invert the plant in order to provide control signals which track a given input signal. In the generalized learning architecture the desired plant input signal is assumed known and the neural network is trained to produce input signals for the next sampling interval given the current plant output. The result is an output feedback control. The third architecture is called the specialized learning architecture where the neural network is trained to provide control to track an input function by minimizing the tracking error.

Levin and Narendra [32] present a theory for the design of neural control systems which stabilize nonlinear dynamic systems about an equilibrium point. This theory is based on nonlinear control theory. The article contains necessary background information in nonlinear control theory and many examples illustrating the interaction between nonlinear theory and the use of neural networks for stable regulation. Possible control methods for nonlinear systems include: 1) the use of a linear controller which assumes that the plant can be linearized about the operating point, 2) stabilizing control using feedback stabilization where a change in state variables and a feedback control law are used to transform a system into one which is linear about an operating point, and 3) direct stabilization through the use of a nonlinear control law. Neural control designs are given for the feedback stabilization and direct stabilization methods.

As stated above, adaptive and learning control systems depend on the ability to identify plant dynamics. There have been a number of contributions in the use of neural networks for system identification. Narendra and Parthasarathy discuss feedforward and recurrent neural network structures for identification and control of systems [33]. The authors present a method for training recurrent neural networks and describe necessary assumptions for well posed neural control problems. Fernandez, Parlos, and Tsai investigate nonlinear system identification with neural networks by using a recurrent network to identify nonlinear dynamic systems in discrete time based on input-output measurements. The results are applied to the identification of boiler dynamics [34]. Polycarpou and Ioannou present a stability theory approach to synthesis and analysis of identification and control schemes in nonlinear systems using neural networks [35]. Both gradient and Lyapunov synthesis approaches are applied.

Applications of neural networks in adaptive control have also been investigated by several researchers. Guez, Eilbert, and Kam [36] propose a neural network architecture for neural model reference adaptive control. This system adjusts feedback gains so that the closed loop time response matches a desired time response of a given reference model. Hoskins, Hwang and Vagners [37] use iterative inversion of a neural plant model to provide control signals to the plant. The method is applied to a problem in redundant manipulator kinematics, a model reference adaptive control system, and a linear mass-spring-damper system. Hoskins and Himmelblau use similar techniques with an emphasis on reinforcement learning applied to process control [38].

Goldenthal and Farrell [39] backpropagate the error between the actual plant and a reference model through a neural network model of the plant and then continue the backpropagation procedure through the controller network to update controller weights. The technique is demonstrated in a model reference neural adaptive control system applied to the cart-pole problem. To accomplish this, the backpropagation algorithm is extended so that the network can function as a closed-loop controller and to force the closed loop system to match desired reference response.

Lan and Chand also investigate the discrete time linear quadratic regulator problem [40]. They point out that the conventional solution of the problem is an off-line solution. The computed control history is stored and used later in an open loop control. The disadvantage to this approach is that it is not robust and does not work for time-varying systems. Lan and Chand formulate an augmented performance index with the linear constraint equations of the controlled system embedded. The augmented performance index is then related to parameters in the energy function of a Hopfield network [41]. The Hopfield network then minimizes the performance index in an iterative fashion producing the required optimal control.

Iiguni, Sakai and Tokumaru [42] report a nonlinear regulator design which uses feedforward neural networks to augment a linear quadratic regulator design for a nonlinear plant with parameter uncertainties. The authors assume that the nonlinear plant can be modeled using a known linear state space model. This linear model is then used as the basis for a linear quadratic regulator (LQR) design. The LQR design procedure yields gains for plant state feedback which minimizes a linear quadratic performance index. We now have a regulator design which may be used with the actual plant, however, the range of optimal control operation is limited.

Bouzerdoum and Pattison give a method for mapping a class of optimization problems onto a recurrent neural network architecture [43]. The method minimizes a static quadratic performance index,

$$J(x) = \frac{1}{2} x^T Q x - x^T y \tag{1}$$

with respect to vectors $x \in \mathbb{R}^n$ subject to bound constraints

$$\mu_i \leq x_i \leq \nu_i , \qquad i = 1, \cdots, n \tag{2}$$

where the subscripts indicate components of the respective vectors. This static optimization problem has a known solution. However, a matrix inversion is necessary and this is computationally intensive for large dimensional spaces and difficult for ill-conditioned weighting matrices. The recurrent neural network solution provides a parallel implementation for solving the problem.

Antony and Acar develop algorithms for real-time optimal control of discrete systems with respect to a quadratic performance index over a finite time interval [44]. Problem formulations based on the discrete time Hamiltonian for linear and partially unknown nonlinear systems are given. The method depends on a model of the plant dynamics using a feedforward neural network. Two distinct methods are given. For the first method, control vectors at each sample instant are modified during

every iteration of the algorithm. The second method develops the optimal control by a backward sweep beginning at the final time. The second method has slower convergence rates but requires less storage and fewer computations during each iteration.

In this research, we have formulated two types of neural networks. The first one is called an "Adaptive Critic' architecture. The reason for choosing this structure for formulating the hypersonic vehicle optimal control problems are: 1) this structure obtains an optimal controller through solving dynamic programming equations, 2) this approach (see, Figure 1), has a supervisor (critic) which 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, and 5) it preserves the same structure regardless of the problem (linear or nonlinear).

The adaptive critic method determines an optimal control law for a system by successively adapting two networks, an action and a critic network. The control law does not need to be determined *a priori* mathematically. This method simultaneously computes and adapts the neural networks to the optimal control policy for both linear and nonlinear systems. In addition, it is important to know that the form of control does not need to be known in order to use this method. Since the control law is computed for a range of initial conditions, this approach is ideal for design studies.

The second approach is to formulate a neural network for simultaneous identification and control. This uses a modified form of Hopfield neural networks. The need for this network arose after the customer indicated that there is a large level of uncertainty in the system parameters. We anticipated the need for this during the second year and formulated the network while awaiting the

POST3D program and inputs. Research and development based on this approach are presented as a conference paper at the end. This paper was presented at the 1996 Atmospheric Flight Mechanics Conference in July 1996 at San Diego, CA. This paper is enclosed in the Appendix.

The first part of the rest of this report deals with the adaptive critic approach, problem formulation, algorithm development and results.

IV. PROBLEM FORMULATION

1. Statement of the General Problem

In this study a problem of the form (finite-time with terminal constraints) where a cost function, J, given by

$$J=\phi(x(t_f))+\int_0^{t_f}\psi(x(\tau),u(\tau))d\tau \tag{3}$$

subject to differential constraints

$$\dot{x}=f(x,u) \tag{4}$$

$$t_f \equiv \text{given} \qquad x_o \equiv \text{given} \tag{5}$$

is considered. x is an n-dimensional state vector, u is an m-dimensional control vector, $\phi(\,)$, $\psi(\,)$, and f( ) are linear or nonlinear functions of state and/or control. x_o are the initial conditions and t_f is the final time.

2. Dynamic Programming Background

We can rewrite Eq. (3)

$$J(x(t))=U(x(t),u(x(t)))+<J(x(t+1))> \tag{6}$$

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. $<J(x(t+1))>$ 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

$$\lambda(x(t))\equiv\frac{\delta J(x(t))}{\delta x(t)} \tag{7}$$

then

$$\begin{aligned}\lambda(x(t))=&\frac{\delta U(x(t),u(t))}{\delta x(t)}+\frac{\delta U(x(t),u(t))}{\delta u(t)}\\&+\left\langle\lambda(x(t+1))\frac{\delta x(t+1)}{\delta x(t)}\right\rangle+\left\langle\lambda(x(t+1))\frac{\delta x(t+1)}{\delta u(t)}\frac{\delta u(x(t))}{\delta x(t)}\right\rangle\end{aligned} \tag{8}$$

From this it can be seen that if $<\lambda(x(t+1))>$, 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

$$\frac{\delta J(x(t))}{\delta u(t)} = 0 \tag{9}$$

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

3. Training Methods (Approximation Techniques)

This study uses Eqns. (8) and (9) 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, or approximating $\lambda(x(t))$. Both networks are assumed to be feedforward multiple layer perceptron networks.

The schematics of the controller (action) and critic networks are presented in Figures 2 and 3. 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.

In order to train the critic network for the time step t, $x(t)$ 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. (8) 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. In an infinite-dimensional problem, the training ends with one stage; however, for a finite dimensional problem, such as this study, this series of steps is used at each stage. This process will be explained in detail in the next section.

V. OPTIMIZATION/CONTROL

Motivation for the formulation in this section comes from the need of the customer in that they would like to study the trajectories from the scramjet turn-off to the rocket burn-out conditions

of a certain vehicle. The reason for this is the uncertainties in the parameters of the earlier stage designs. Consequently, there will be an envelope of conditions from which the rocket will have to start and yet carry the payload to the pre-specified burn-out conditions. It is assumed that the rocket burn-out conditions will ensure a proper apogee through the coasting period.

The cost function is given by

$$J = - S_1 m_f + \frac{1}{2} S_2 \left(v_f - v_{f_D}\right)^2 + \frac{1}{2} S_3 \left(\gamma_f - \gamma_{f_D}\right)^2 + \frac{1}{2} S_4 \left(h_f - h_{f_D}\right)^2 \tag{10}$$

where

J	≡	cost function to be minimized
m	≡	mass
v	≡	velocity
γ	≡	flightpath angle
h	≡	altitude
S_i	≡	weights on the final conditions

Subscripts

f	≡	final
f_D	≡	desired final

Note that this cost function maximizes the final payload while ensuring that the velocity, the flightpath angle and the altitude at the final time are as close to the final/desired burn-out conditions as possible. The equations of motion are given by

$$\dot{m} = -\frac{T}{g I_{sp}} \tag{11}$$

$$\dot{h} = v \sin\gamma \tag{12}$$

$$\dot{v} = (T\cos\alpha - D)/m - \mu \sin\gamma / r^2 \tag{13}$$

$$\dot{\gamma} = (T\sin\alpha + L)/mv + \left(\frac{v}{r} - \mu / r^2 v\right)\cos\gamma \tag{14}$$

where

T	$\equiv$	thrust
L	$\equiv$	$k_1, \alpha \frac{1}{2}\rho v^2 S \equiv$ lift
D	$\equiv$	$\left(C_{D_o} + k_2 \alpha^2\right)\frac{1}{2}\rho v^2 S \equiv$ drag
μ	$\equiv$	gravitational constant
r	$\equiv$	radial distance from the center of the earth
R_e	$\equiv$	radius of the earth
I_{s_p}	$\equiv$	specific impulse
α	$\equiv$	angle of attack

A schematic of the scenario is presented in Figure 4. Final time is unknown. That means, this is a 'free-final time' problem. There is no solution in the current literature for solving the 'free-final time' problem for an envelope of initial conditions (other than the general method of dynamic programming).

In order to solve this problem with neural networks, we transform it to one where altitude is the independent variable. Through this step, we convert it to a problem where we can break it down

into several segments of altitude; this also allows us to reach the final desired altitude in all cases. The initial conditions for this scenario are the possible final conditions from the termination of scramjet.

This is a two-point boundary value problem where the initial conditions are known but the final conditions are unknown. Usually, it is solved for a given set of initial conditions; however, in this project we develop an adaptive critic-based solution which will solve the problem for an envelope of initial conditions. By reformulating the model, we are able to remove altitude from the cost function since the final condition in altitude is satisfied exactly.

The reformulated equations of motion with altitude, **h** as the independent variable, are given by

$$dm/dh = T / gI_{sp} \cdot 1 / v \sin\gamma \tag{15}$$

$$dv/dh = \left[\frac{T\cos\alpha - \left(C_{D_o} + K_2\alpha^2\right)\frac{1}{2}\rho v^2 S}{mv\sin\gamma}\right] - g/v \tag{16}$$

where the drag coefficient has been approximated with a parabolic drag polar with a least squares fit.

$$d\gamma/dh = \left[\frac{T\sin\alpha + k_1\alpha\frac{1}{2}\rho v^2 S}{mv^2\sin\gamma}\right] + \left[\frac{v}{r} - \frac{g}{v}\right]\frac{\cos\gamma}{v\sin\gamma} \tag{17}$$

In Eqns. (15-17), where lift coefficient C_L has been approximated with a linear least squares fit.

C_{D_o}, K_2, K_1 = constants

g = local acceleration due to gravity

In order to calculate the flight time, a fourth equation is added as,

$$dt/dh = \frac{1}{v \sin \gamma} \tag{18}$$

For solutions with neural networks, we convert these nonlinear differential equations to single-step discrete equations as:

$$m_{k+1} = m_k - \left(\frac{T_k}{g_k I_{sp}} \cdot \frac{1}{v_k \sin \gamma_k} \right) \Delta h \tag{19}$$

$$v_{k+1} = v_k + \left[\left(T_k \cos \alpha_k - D_k \right) / m_k v_k \sin \gamma_k - g_k / v_k \right] \Delta h \tag{20}$$

$$\gamma_{k+1} = \gamma_k + \left[\left(T_k \sin \alpha_k + L_k \right) / m_k v_k^2 \sin \gamma_k + \left(\frac{v_k}{r_k} - \frac{g_k}{v_k} \right) \frac{\cos \gamma_k}{v_k \sin \gamma_k} \right] \Delta h \tag{21}$$

$$t_{k+1} = t_k + \left(\frac{1}{v_k \sin \gamma_k} \right) \Delta h \tag{22}$$

where

Δh = step size in altitude

k = stage

The corresponding Hamiltonian of the optimized problem is

$$H_k = \lambda_{m_{k+1}} m_{K+1} + \lambda_{v_{k+1}} v_{k+1} + \lambda_{\gamma_{k+1}} \gamma_{k+1} \tag{23}$$

where

$\lambda_{x_{k+1}}$ = Lagrangian multiplier for variable x at stage (k+1).

The propagation equations for the Lagrange's multipliers are obtained by partial differentiation of the Hamiltonian with respect to the states. They are:

$$\lambda_{x_k} = \frac{\partial H_k}{\partial x_k} \quad , \quad x \equiv [m, v, \gamma]^T \tag{24}$$

$$\lambda_{m_k} = \lambda_{m_{k+1}} - \frac{\Delta h}{m_k^2}\left[\frac{T_k \cos\alpha_k - D_k}{v_k \sin\gamma_k} \cdot \lambda_{v_{k+1}} + \frac{T_k \sin\alpha_k + L_k}{v_k^2 \sin\gamma_k} \lambda_{\gamma_{k+1}}\right] \tag{25}$$

$$\begin{aligned}\lambda_{v_k} = \lambda_{v_{k+1}} + \frac{\Delta h}{v_k \sin\gamma_k}\Bigg[&\lambda_{m_{k+1}} \frac{T_k}{g_k I_{sp}} \cdot \frac{1}{v_k} \\ &+\lambda_{v_{k+1}}\left[-\frac{T_k \cos\alpha_k}{m_k v_k} + \frac{g_k \sin\gamma_k}{v_k} - \frac{D_k}{m_k v_k}\right] \\ &+\lambda_{\gamma_{k+1}}\left[-\frac{2T_k \sin\alpha_k}{m_k v_k^2} + \frac{2g_k \cos\gamma_k}{v_k^2}\right]\Bigg]\end{aligned} \tag{26}$$

$$\begin{aligned}\lambda_{\gamma_k} = \lambda_{\gamma_{k+1}} + \frac{\Delta h}{v_k \sin\gamma_k}\Bigg[&\frac{T_k \cot\gamma_k}{g_k I_{sp}} \lambda_{m_{k+1}} \\ &+ \lambda_{v_{k+1}}\left[\frac{-(T_k \cos\alpha_k - D_k)\cot\gamma_k}{m_k}\right] \\ &+ \lambda_{\gamma_{k+1}}\left[-\frac{(T_k \sin\alpha_k + D_k)\cot\gamma_k}{m_k v_k^2} - \left(\frac{v_k}{r_k} - \frac{g_k}{v_k}\right)\frac{1}{\sin\gamma_k}\right]\Bigg]\end{aligned} \tag{27}$$

Note that λ_{k+1} is needed to solve for λ_k. The boundary conditions for the multiplier equations are

$$\lambda_f \equiv \left(\frac{\partial J}{\partial X} \right)_{X = X_f} \tag{28}$$

Optimal control is obtained by partially differentiating the Hamiltonian with respect to the control. In our case, angle of attack, α, is the control variable. We get

$$\frac{\partial H_k}{\partial \alpha_k} = 0 \tag{29}$$

This gives

$$\begin{aligned} &\lambda_{v_{k+1}} \left[-\left(T_k \sin\alpha_k + k_2 \rho_k v_k^2 s \alpha_k \right) \right] \\ &+ \lambda_{\gamma_{k+1}} \frac{\left[T_k \cos\alpha_k + k_1 \frac{1}{2} \rho_k v_k^2 s \right]}{v_k} = 0 \end{aligned} \tag{30}$$

First, we solve for the control at the $(N-1)^{th}$ stage where N is the preselected number of stages. That is, (after using small angle (α)) assumption

$$\begin{aligned} &\lambda_{v_N} \left[-\left(T_{N-1} + K_2 \rho_{N-1} v_{N-1}^2 s \right) \alpha_{N-1} \right] \\ &+ \lambda_{\gamma_N} \left[T_{N-1} + k_1 \frac{1}{2} \rho_{N-1} v_{N-1}^2 s \right] / v_{N-1} = 0 \end{aligned} \tag{31}$$

Note that

$$\lambda_{v_N} = S_2 \left(v_N - v_{f_D} \right) \tag{32}$$

$$\lambda_{\gamma_N} = S_3 \left(\gamma_N - \gamma_{f_D} \right) . \tag{33}$$

By substituting for λ_{v_N} and λ_{γ_N} in Eq. (31), we get

$$\begin{aligned} & S_2 \left[v_N - v_{fD} \right] \left[- \left(T_{N-1} + k_2 \rho_{N-1} v_{N-1}^2 s \right) \right] \alpha_{N-1} \\ & + S_3 \left[\gamma_N - \gamma_{fD} \right] \left[T_{N-1} + k_1 \frac{1}{2} \rho_{N-1} v_{N-1}^2 s \right] / v_{N-1} = 0 \end{aligned} \tag{34}$$

We substitute for v_N and γ_N in Eq. (34) in terms of v_{N-1} and γ_{N-1} by using propagation equations, Eq. (25-27).

$$\begin{aligned} & S_2 \left[v_{N-1} + \left[\frac{- T_{N-1} \left(C_{D_D} + k_2 \alpha_{N-1}^2 \right) q_{N-1} S}{m_{N-1} v_{N-1} \sin \gamma_{N-1}} - \frac{q_{N-1}}{v_{N-1}} \right] \Delta h \right. \\ & \qquad \left. - v_{fD} \right] \left[- \left(T_{N-1} + 2 k_2 q_{N-1}^s \right) \alpha_{N-1} \right] \\ & + S_3 \left[\gamma_{N-1} + \left[\frac{\left(T_{N-1} + k_1 q_{N-1} S \right) \alpha_{N-1}}{m_{N-1} v_{N-1}^2 \sin \gamma_{N-1}} + \left(\frac{v_{N-1}}{r_{N-1}} - \frac{q_{N-1}}{v_{N-1}} \right) \frac{\cot \gamma_{N-1}}{v_{N-1}} \right] \right. \\ & \qquad \left. - \gamma_{fD} \right] \left[\frac{T_{N-1} + k_1 q_{N-1} S}{V_{N-1}} \right] = 0 \end{aligned} \tag{35}$$

where the dynamic pressure q_{N-1} is

$$q_{N-1} = \frac{1}{2} \rho_{N-1} v_{N-1}^2 \tag{36}$$

This leads to a cubic equation in α_{N-1} as

$$T_2 T_3 \alpha_{N-1}^3 + \left(T_1 T_3 + T_5 T_6\right) \alpha_{N-1} + T_4 T_5 = 0 \tag{37}$$

where

$$T_1 = S_2 \left[v_{N-1} + \left(\frac{T_{N-1} - C_{D_D} q_{N-1} S}{m_{N-1} v_{N-1} \sin \alpha_{N-1}} - \frac{q_{N-1}}{v_{N-1}} \right) \Delta h - v_{fD} \right] \tag{38}$$

$$T_2 = - \frac{S_2 k_2 q_{N-1} S}{m_{N-1} v_{N-1} \sin \gamma_{N-1}} \Delta h \tag{39}$$

$$T_3 = - \left(T_{N-1} + 2 k_2 q_{N-1} S\right) \tag{40}$$

$$T_4 = S_3 \left[\gamma_{N-1} + \left(\frac{v_{N-1}}{r_{N-1}} - \frac{q_{N-1}}{v_{N-1}} \right) \frac{\cot \gamma_{N-1}}{v_{N-1}} \Delta h - \gamma_{fD} \right] \tag{41}$$

$$T_5 = \frac{T_{N-1} + k_1 q_{N-1} S}{v_{N-1}} \tag{42}$$

$$T_6 = S_3 \, T_5 / \left(m_{N-1} \sin \gamma_{N-1} v_{N-1}\right) \tag{43}$$

We can observe that all quantities are known in terms of quantities at N-1. That is α_{N-1} is available as a feedback control based on states at N-1.

For all other stages, k, we obtain the expression for control in terms of the Lagrangian multipliers at k + 1.

$$\alpha_K = \frac{\lambda_{\gamma_{k+1}}}{\lambda_{v_{k+1}}} \cdot \frac{1}{v_k} \cdot \frac{T_k + k_1 q_k S}{T_k + 2k_2 q_k S} \tag{44}$$

How do we construct the neural networks to solve this problem?

1. Solve for α_{N-1} in terms of m_{N-1}, v_{N-1}, γ_{N-1}

   Generate various α_{N-1} by changing m_{N-1} , v_{N-1}, γ_{N-1} .

   Use a neural network to output α_{N-1} for m_{N-1} , v_{N-1}, γ_{N-1} .... called α_{N-1} network.

   [We have optimal α_{N-1} now]

2. In order to solve for α_k, (k=0,1,2...N-2), of m_k, v_k, γ_k, we need $\lambda_{m_{k+1}}$, $\lambda_{v_{k+1}}$, $\lambda_{\gamma_{k+1}}$. So, use λ_N, m_{N-1}, v_{N-1}, γ_{N-1} and α_{N-1} from step 1 to solve for $\lambda_{m_{N-1}}$, $\lambda_{v_{N-1}}$, and $\lambda_{\gamma_{N-1}}$ using the λ-(backward) propagation equations, Eq. (25-27). Train a neural network with m_{N-1}, v_{N-1}, γ_{N-1} as inputs and λ_{N-1} as output. Call this λ_{N-1} network.

   [We have optimal λ_{N-1} now.]

How do we construct other networks?

3. Assume different values of m_{N-2}, v_{N-2}, γ_{N-2} and use a neural network to output α_{N-2}. This will <u>not</u> be optimal. Use m_{N-2}, v_{N-2}, γ_{N-2}, α_{N-2} in state equations, Eq. (19-21), to obtain m, v, γ at N-1. Use these states in λ_{N-1} network to output λ_{N-1}. Use these λ_{N-1} in optimal α_k equation, Eq. (44), to compute $(\alpha_{N-1})_{\text{target}}$. Continue this process till convergence.

   [We have optimal α_{N-2} now.]

4. Assume different values of m_{N-2}, v_{N-2}, γ_{N-2} and use them to get α_{N-2} from αN-2 network. Use all these in the state propagation equations to calculate states at N-1. Input these states in λ_{N-1} network to get λ_{N-1}. Use this λ_{N-1} and states and control at N-2 to find λ_{N-2} from the λ propagation equations. Construct a λ_{N-2} network to output λ_{N-2} with m_{N-2}, v_{N-2}, γ_{N-2} as inputs. [We have optimal γ_{N-2} now.]

5. Assume different values of m_{N-3}, v_{N-3}, γ_{N-3} and construct an α_{N-3} network similar to α_{N-2} network in step 3.

6. Construct a λ_{N-3} network similar to λ_{N-2} network in step 4.

Continue this process from k = N-1, N-2, ...0

How do we use these networks to generate optimal trajectory from given initial conditions? Assume any m_o, v_o,γ_o and h_o [within the trained range]. Use α_o neural network to find optimal α and integrate till h for α_1 network is reached. Use the m_1, v_1, γ_1 values to find α_1 from the α_1 neural network and integrate till h_2 is reached, and so on, till h_f is reached.

Note that the forward integration can be done in terms of time and note that the Lagrange multiplier network, used in the controller synthesis, is not needed now.

VI. NUMERICAL RESULTS

In order to verify the applicability of the adaptive critic approach to flexible trajectory optimization, we used the rocket vehicle contained in a test case sent by the customer. We present the results corresponding to two stages of neural-controlled trajectories from the burn-out of the rocket in Figures 5-15. The desired end conditions are v_f = 7617 ft/sec, γ_f = 16.636 deg., h_f = 243,600 ft. In trying to match the final conditions, the values of S_1, S_2, and S_3 are chosen to be 1,

1, and 10^6. This means that we desire to try and match the final flightpath angle more closely related to maximization of final weight and matching the desired final velocity. Effect of changes to initial flightpath angle are presented in Figures 5-7. We have fixed the initial velocity and mass and changed the initial flightpath angles. It can be observed from Figure 5 that after following different paths of velocity in Stage 1 for the first 12.2 seconds, all the 10 paths try to converge; the same trend can be seen in Figure 6 which shows the flightpath angle histories. Due to the relative emphasis on the flightpath angle, we can observe that the flightpath angles are more convergent to the desired final value than the velocities. The weight history is almost invariant since the thrust is almost constant. Figures 8-11 represent the mass, flightpath angle, velocity, and altitude histories with time where we change the initial mass in steps. The effectiveness of this formulation is clear from the flightpath angle history presented in Figure 9. Even though the initial step (due to changes in mass) leads to different flightpath angles, the control from the last stage brings them very close. Although velocities appear divergent, it should be observed that they are scattered close to the desired final value. The altitude history is very close to the same in all the cases as expected and satisfies the final condition. Figures 12-15 represent the state variable histories due to changes in the velocities. Due to the divergence of the flightpath angle value at the end of the first stage, the second stage velocities show apparent deviations from the desired value so that the resulting second stage flightpath angles can be closer to the desired value. The slight variations in the final altitude are due to the forward integration in time which we limited to 20.4 seconds.

VII. CONCLUSIONS

An approach to solving 'free final time' problems with an envelope of initial conditions has been proposed. This approach called 'the adaptive critic' consists of two neural networks at stage developed in a backward sweep. After development, only the controller is used in forward integration of trajectories. Numerical results from the last stage of a launch vehicle trajectory (provided by the customer) show that this approach works well and can be used in design. Further work will involve integration with POST3D, consideration of the other phases of flight etc.

VIII. ACKNOWLEDGMENT

We gratefully acknowledge the partial support provided by NASA grant NAG-1-1728.

BIBLIOGRAPHY

[1 N. Wiener, *Cybernetics: or Control and Communication in the Animal and the Machine.* Cambridge, MA: MIT Press, 1949.

[2] W. Ashby, *Design for a Brain.* New York, NY: Wiley & Sons, 1952.

[3] W. Ashby, *An Introduction to Cybernetics.* New York, NY: Chapman & Hall, Ltd., 1957.

[4] K. S. Fu, "Learning Control Systems," in *Computer and Information Sciences* (J. T. Tou and R. H. Wilcox, eds.), pp. 318-343, Washington D. C.: Spartan Books, 1964.

[5] J. Sklansky, "Learning Systems for Automatic Control," *IEEE Transactions on Automatic Control*, Vol. AC-11, pp. 6-19, January 1966.

[6] Z. Nikolic and K. Fu, "An Algorithm for Learning without External Supervision and Its Application to Learning Control Systems," *IEEE Transactions on Automatic Control*, Vol. AC-11, pp. 414-422, July 1966.

[7] Y. Tsypkin, "Optimization, Adaptation, and Learning in Automatic Systems," in *Computer and Information Sciences-KK* (J. T. Tou, ed.), pp. 15-32, New York, NY: Academic Press, 1967. Proceedings of the Second Symposium on Computer and Information Sciences held at Battelle Memorial Institute, August 22-24, 1966.

[8] Y. Tsypkin, "Self-Learning--What is it?," *IEEE Transactions on Automatic Control*, Vol. AC-13, pp. 608-612, December 1968.

[9] Y. Tsypkin, *Adaptation and Learning in Automatic Systems.* New York, NY: Academic Press, 1971.

[10] J. Mendel and R. McLaren, "Reinforcement Learning Control and Pattern Recognition Systems," in *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications* (J. Mendel and K. S. Fu, eds.), pp. 287-318, New York, NY: Academic Press, 1970.

[11] A. Barto, "Connectionist Learning for Control: An Overview," in *Neural Networks for Control* (W. T. M. II, R. Sutton, and P. Werbos, eds.), ch. 1, pp. 5-58, Cambridge, MA: MIT Press, 1990.

[12] B. Widrow, N. Gupta, and S. Maitra, "Punish/Reward: Learning with a Critic in Adaptive Threshold Systems," *IEEE Transactions on Systems, Man., and Cybernetics*, Vol. SMC-3, pp. 455-465, September 1973.

[13] 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*, Vol. SMC-13, pp. 834-846, September/October 1983.

[14] R. Sutton, *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, 1984. This is the origin of the Adaptive Heuristic Critic algorithm.

[15] P. Werbos, "Backpropagation and Neurocontrol: A Review and Prospectus," in *Proceedings of the International Joint Conference on Neural Networks*, Vol. I, (Washington D.C.), pp. 209-216, June 18-22, 1989.

[16] J. Jameson, "A Neurocontroller Based on Model Feedback and the Adaptive Heuristic Critic," in *Proceedings of the International Joint Conference on Neural Networks*, (San Diego, CA), pp. II37-II44, IEEE, June 1990.

[17] D. A. Sofge and D. A. White, "Neural Network Based Process Optimization and Control," in *Proceedings of the 29th Conference on Decision and Control*, (New York, NY), pp. 3270-3276, IEEE, December 1990.

[18] J. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulated Controller (CMAC)," *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 97, pp. 220-227, September 1975.

[19] C. Watkins, *Learning with Delayed Rewards*. PhD Thesis, Cambridge University, 1989.

[20] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, Vol. 8, No. 3-4, pp. 279-292, 1992.

[21] S. Bradtke, "Reinforcement Learning Applied to Linear Quadratic Regulation," in *Advances in Neural Information Processing Systems*, pp. 295-302, San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[22] V. Gullapalli, "A Stochastic Reinforcement Learning Algorithm for Learning Real-Valued Functions," *Neural Networks*, Vol. 3, pp. 671-992, 1990.

[23] M. Stamenkovich, "An Application of Artificial Neural Networks for Autonomous Ship Navigation Through a Channel," in *Proceedings of the Vehicle Navigation & Information Systems Conference*, (Dearborn, MI), pp. 475-481, October 20-23, 1991.

[24] R. Shelton and J. Peterson, "Controlling a Truck with an Adaptive Critic CMAC Design," *Simulation*, Vol. 58, pp. 319-326, May 1992.

[25] C. Tham and R. Prager, "Reinforcement Learning Methods for Multi-Linked Manipulator Obstacle Avoidance and Control," Tech. Rep., Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, UK, March 25, 1993.

[26] D. Gachet, M. Salichs, and J. Pimentel, "Learning Emergent Tasks for an Autonomous Mobile Robot," Tech. Rep., Dpto. Ingenieria, Universidad Carlos III de Madrid, Spain, 1994.

[27] B. Bavarian, "Introduction to Neural Networks for Intelligent Control," *IEEE Control Systems Magazine,* Vol. 8, pp. 3-7, April 1988.

[28] W. Miller, R. Sutton, and P. Werbos (Eds.), *Neural Networks for Control.* Cambridge, MA: The MIT Press, 1990.

[29] D. White and D. Sofge (Eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches,* New York, NY: Van Nostrand Reinhold, 1992.

[30] K. Hunt, D. Sbarbaro, R. Zbikowski, and P. Gawthrop, "Neural Networks for Control Systems-A Survey," *Automatica*, Vol. 28, No. 6, pp. 1083-1112, 1992.

[31] D. Psaltis, A. Sideris, and A. Yamamura, "A Multilayered Neural Network Controller," *IEEE Control Systems Magazine*, Vol. 8, pp. 17-21, April 1988.

[32] A. Levin and K. Narendra, "Control of Nonlinear Dynamic Systems Using Neural Networks: Controllability and Stabilization," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 192-206, March 1993.

[33] K. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks,* Vol. 1, pp. 4-27, March 1990.

[34] B. Fernandez, A. Parlos, and W. Tsai, "Nonlinear Dynamic System Identification Using Artificial Neural Networks (Anns)," in *Proceedings of the International Joint Conference on Neural Networks,* Vol 2. (New York, NY), pp. 133-141, IEEE, June 17-21, 1990.

[35] M. Polycarpou and P. A. Ioannou, "Identification and Control of Nonlinear Systems Using Neural Network Models: Design and Stability Analysis," Tech. Rep. Report 91-09-01, Electrical Engineering, University of Southern California, Sept. 1991.

[36] A. Guez, J. Eilbert, and M. Kam, "Neural Network Architecture for Control," *IEEE Control Systems Magazine,* Vol. 8, pp. 22-25, April 1988.

[37] D. Hoskins, J. Hwang, and J. Vagners, "Iterative Inversion of Neural Networks and Its Application to Adaptive Control," *IEEE Transactions on Neural Networks,* Vol. 3, pp. 292-301, March 1992.

[38] J. Hoskins and D. Himmelblau, "Process Control Via Artificial Neural Networks and Reinforcement Learning," *Computers and Chemical Engineering,* Vol. 16, pp. 241-251, April 1992.

[39] W. Goldenthal and J. Farrell, "Application of Neural Networks to Automatic Control," in *Proceedings of the AIAA Conference on Guidance, Navigation, and Control.,* (Portland, OR), pp. 1108-1112, August 20-22 1990.

[40] M.-S. Lan and S. Chand, "Solving Linear Quadratic Discrete-time Optimal Controls Using Neural Networks," in *Proceedings of the 29th Conference on Decision and Control,* (Honolulu, HI), pp. 2770-2772, December 1990.

[41] D. Tank and J. Hopfield, "Simple Neural Optimization Networks: An a/d Converter, Signal Decision Circuit and a Linear Programming Circuit," *IEEE Transactions on Circuits and Systems,* Vol. CAS-33, pp. 533-541, May 1986.

[42] Y. Iguni, H. Sakai, and H. Tokumaru, "A Nonlinear Regulator Design in the Presence of System Uncertainties using Multilayered Neural Networks," *IEEE Transactions on Neural Networks,* Vol. 2, No. 4, pp. 410-417, 1991.

[43] A. Bouzerdoum and T. Pattison, "Neural Network for Quadratic Optimization with Bound Constraints," *IEEE Transactions on Neural Networks,* Vol. 4, pp. 293-304, March 1993.

[44] J. Antony and L. Acar, "Real Time Nonlinear Optimal Control Using Neural Networks," in *Proceedings of the American Control Conference,* Vol. 3, (Baltimore, MD), pp. 2926-2930, June 29-July 1, 1994.

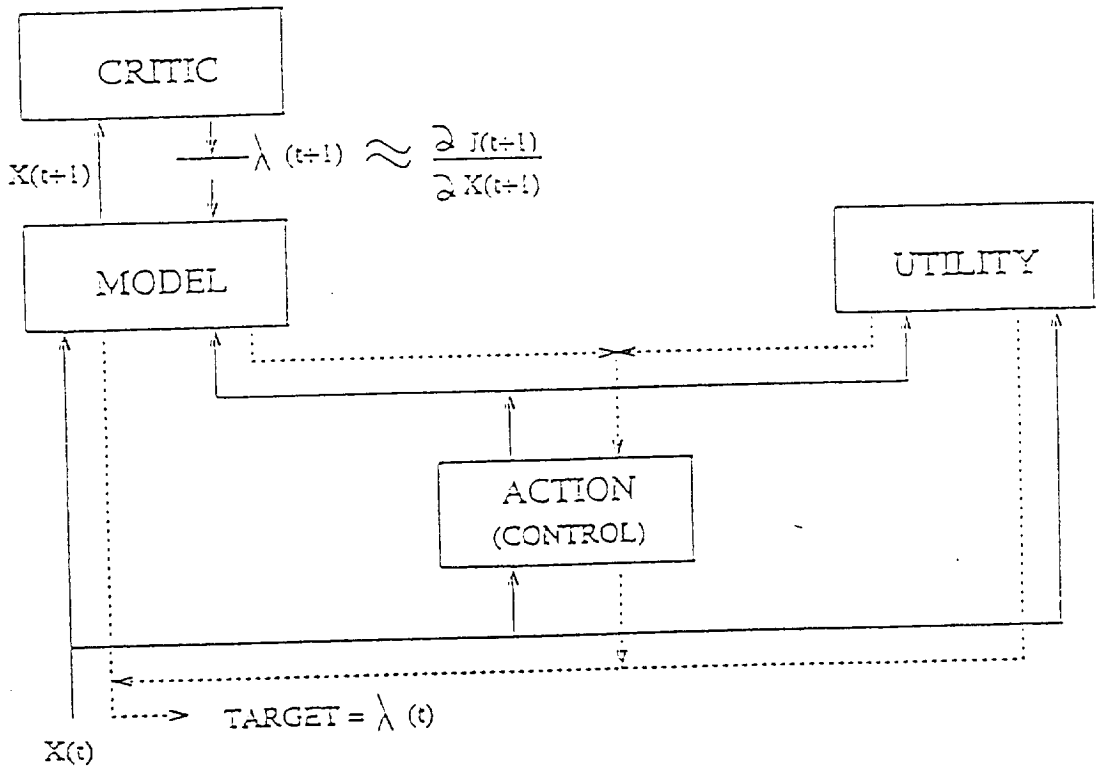

Figure 1: Adaptive Critic for Control

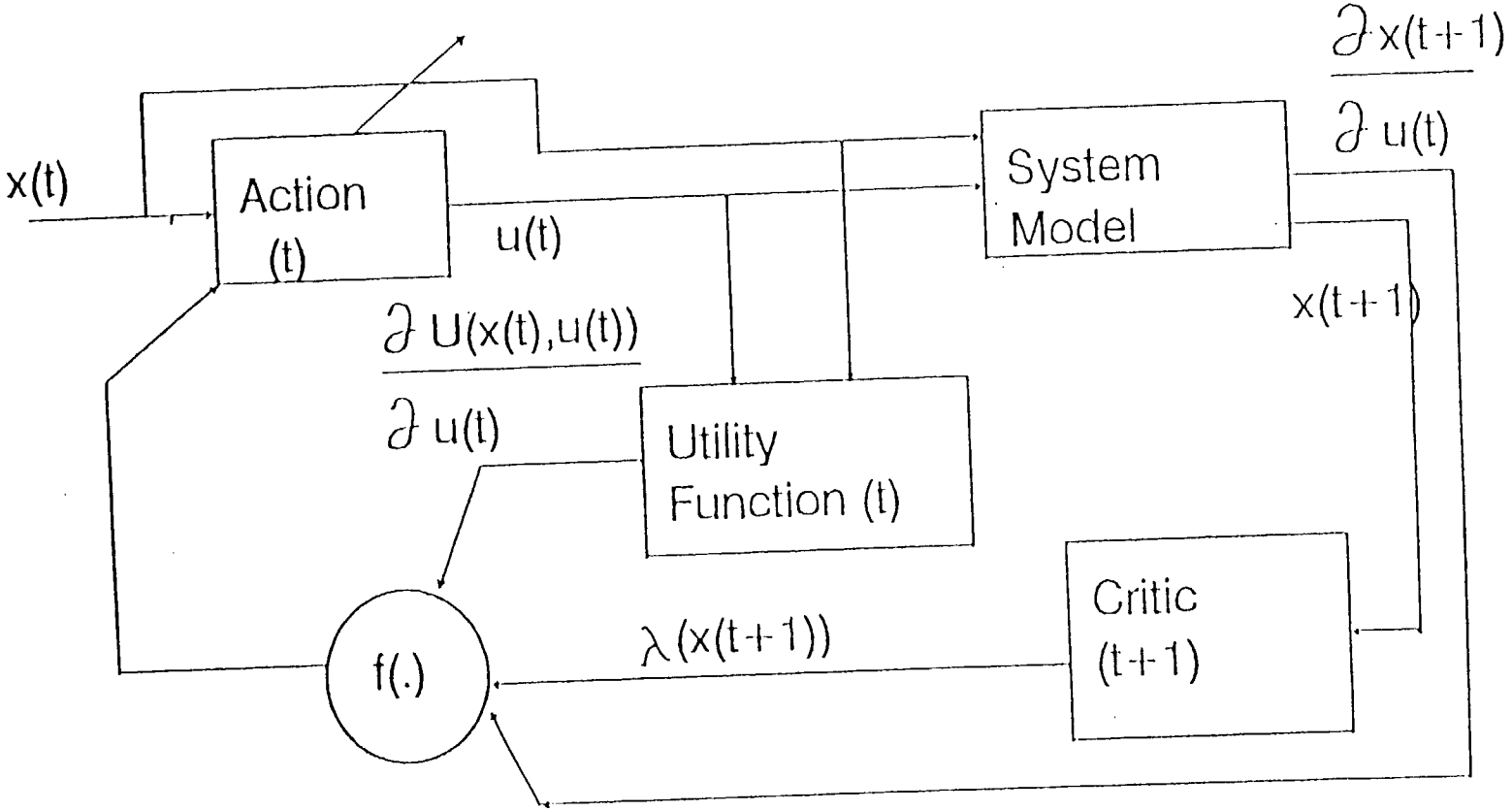

Figure 2: Action Network Training Diagram

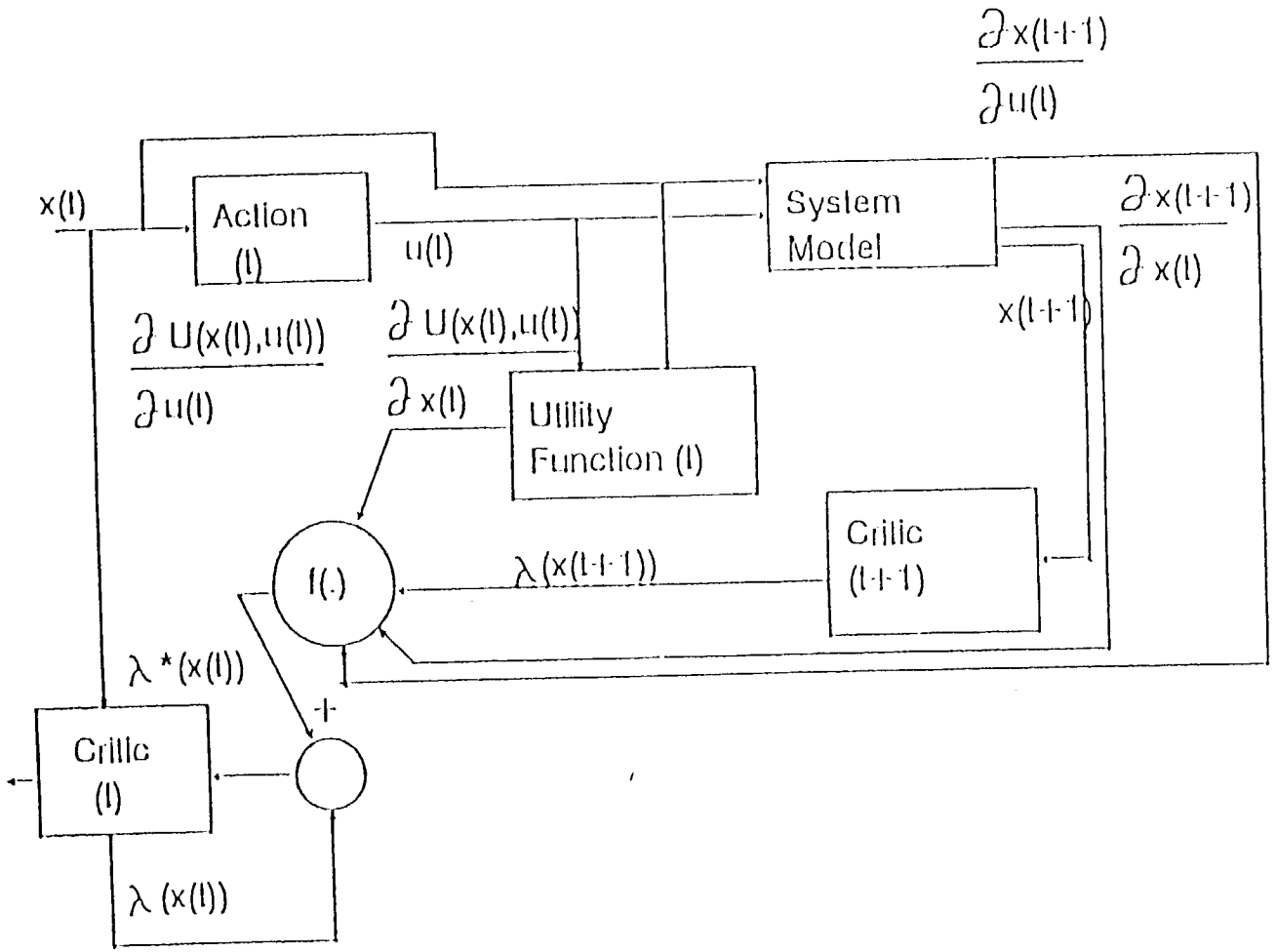

Figure 3: Critic Network Training Diagram

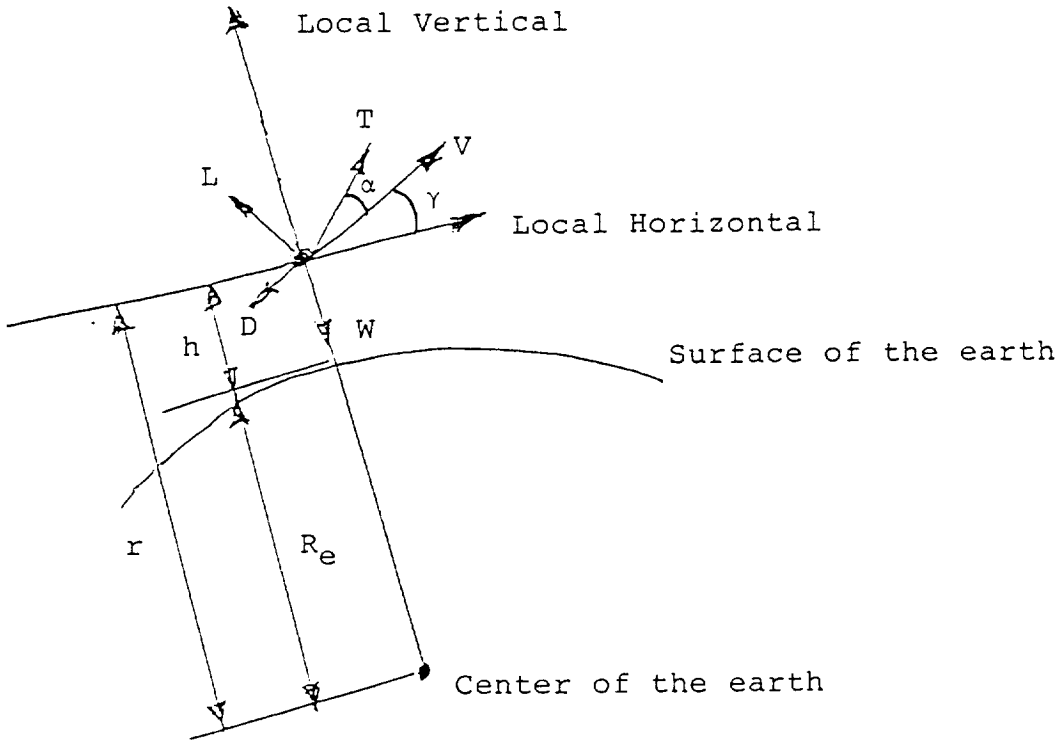

Figure 4: Schematic of the Trajectory Optimization Scenario

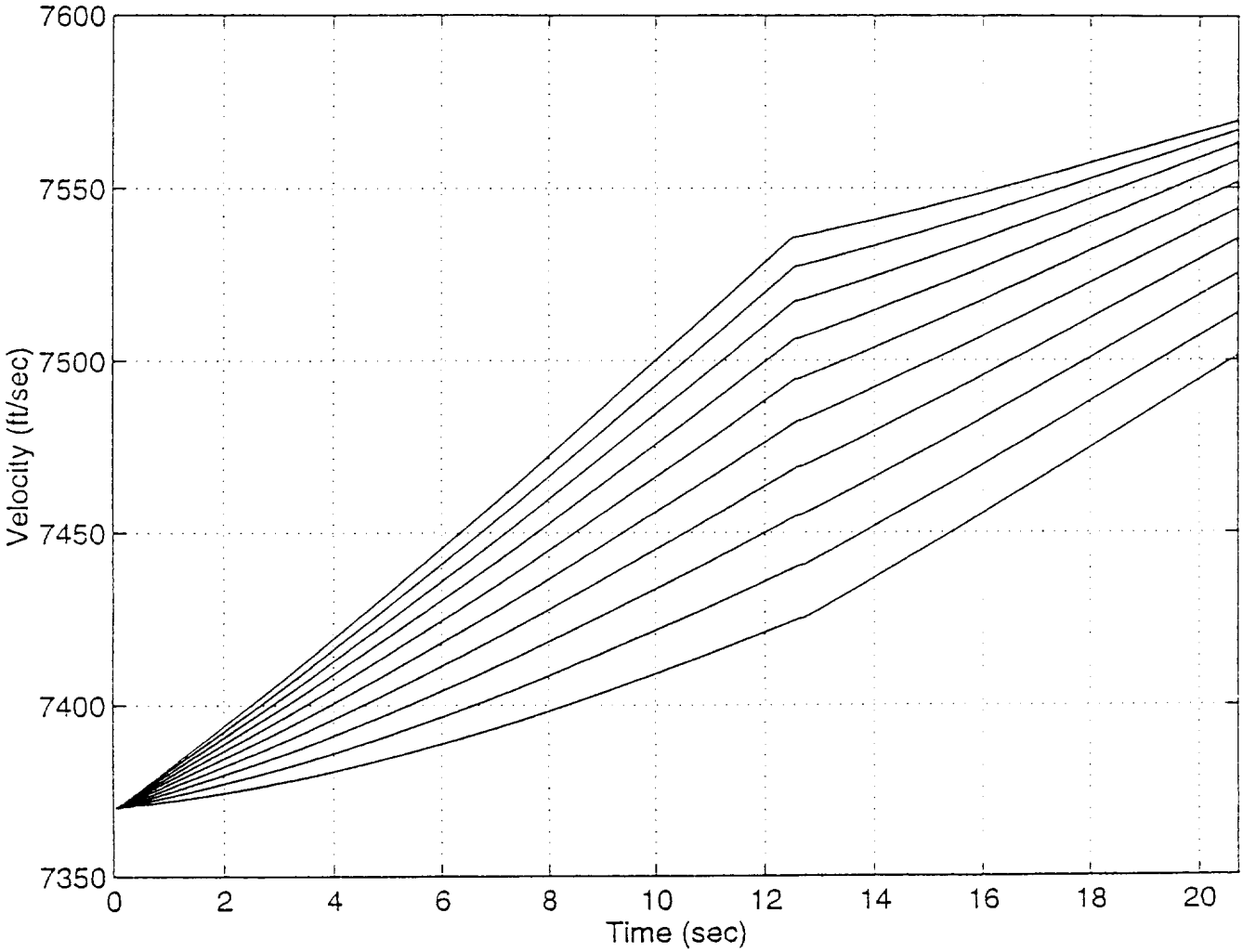

Figure 5: Velocity History
(with changes in initial gamma)

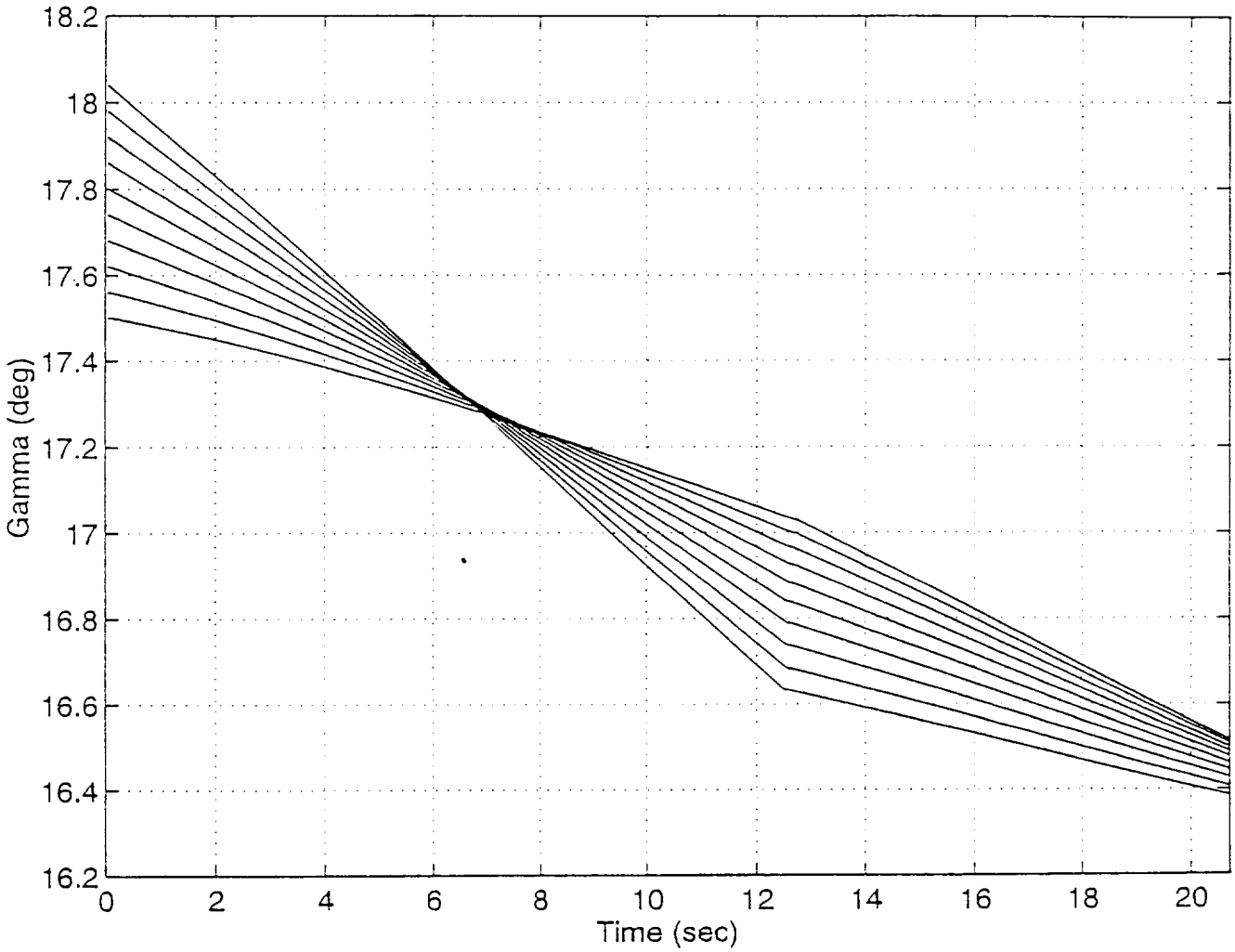

Figure 6: Gamma History
(with changes in initial gamma)

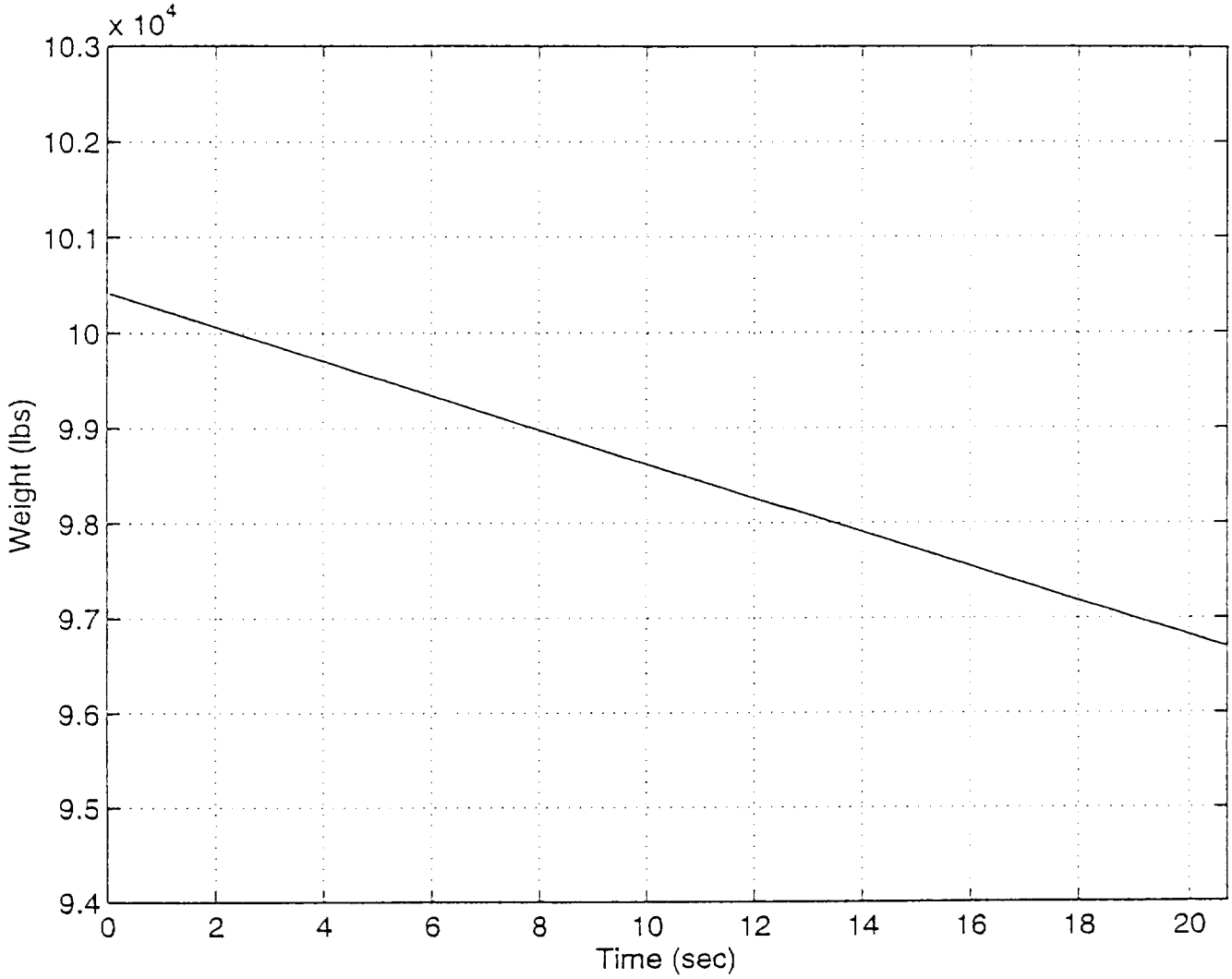

Figure 7: Weight History
(with changes in initial gamma)

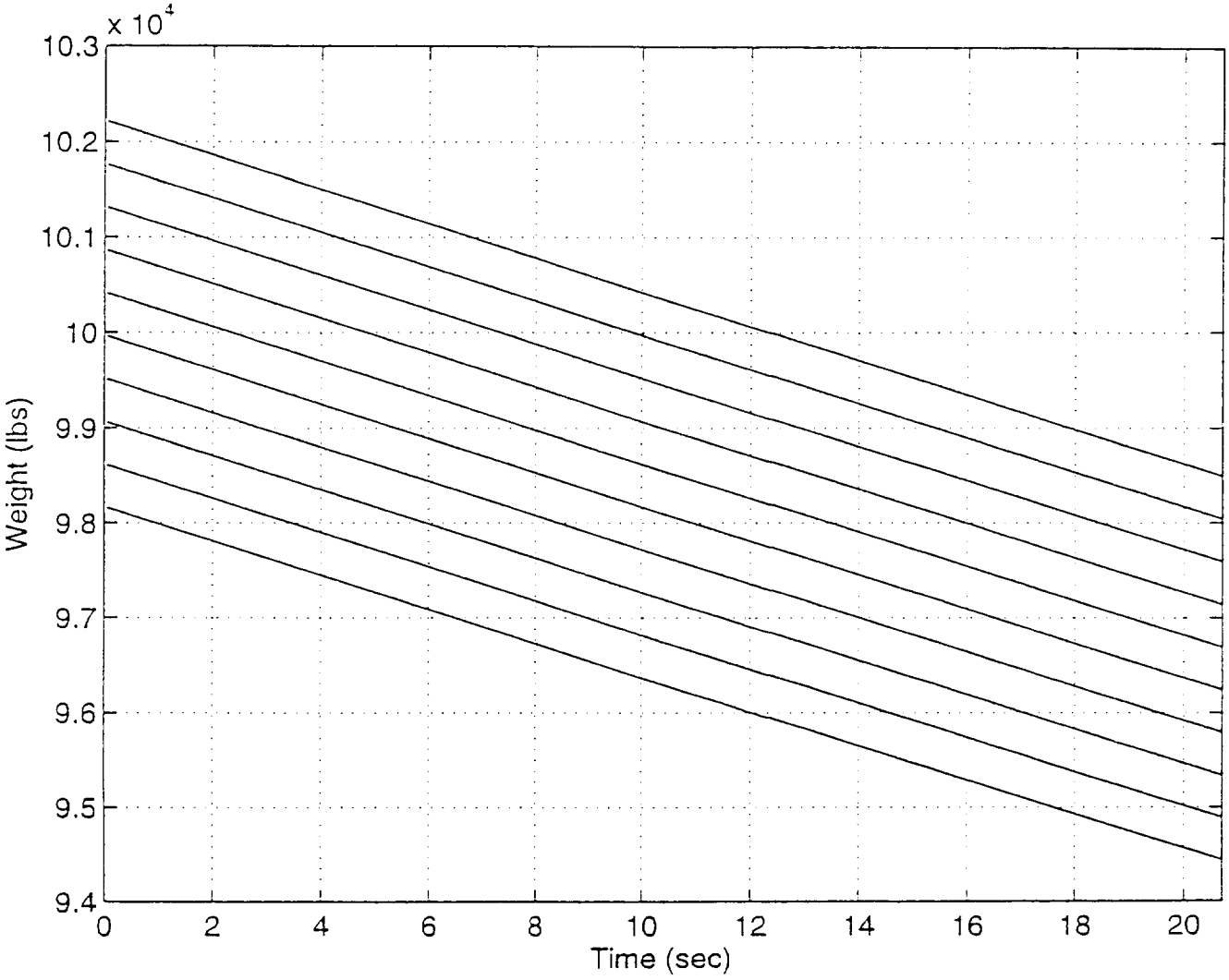

Figure 8: Weight History
(with changes in initial mass)

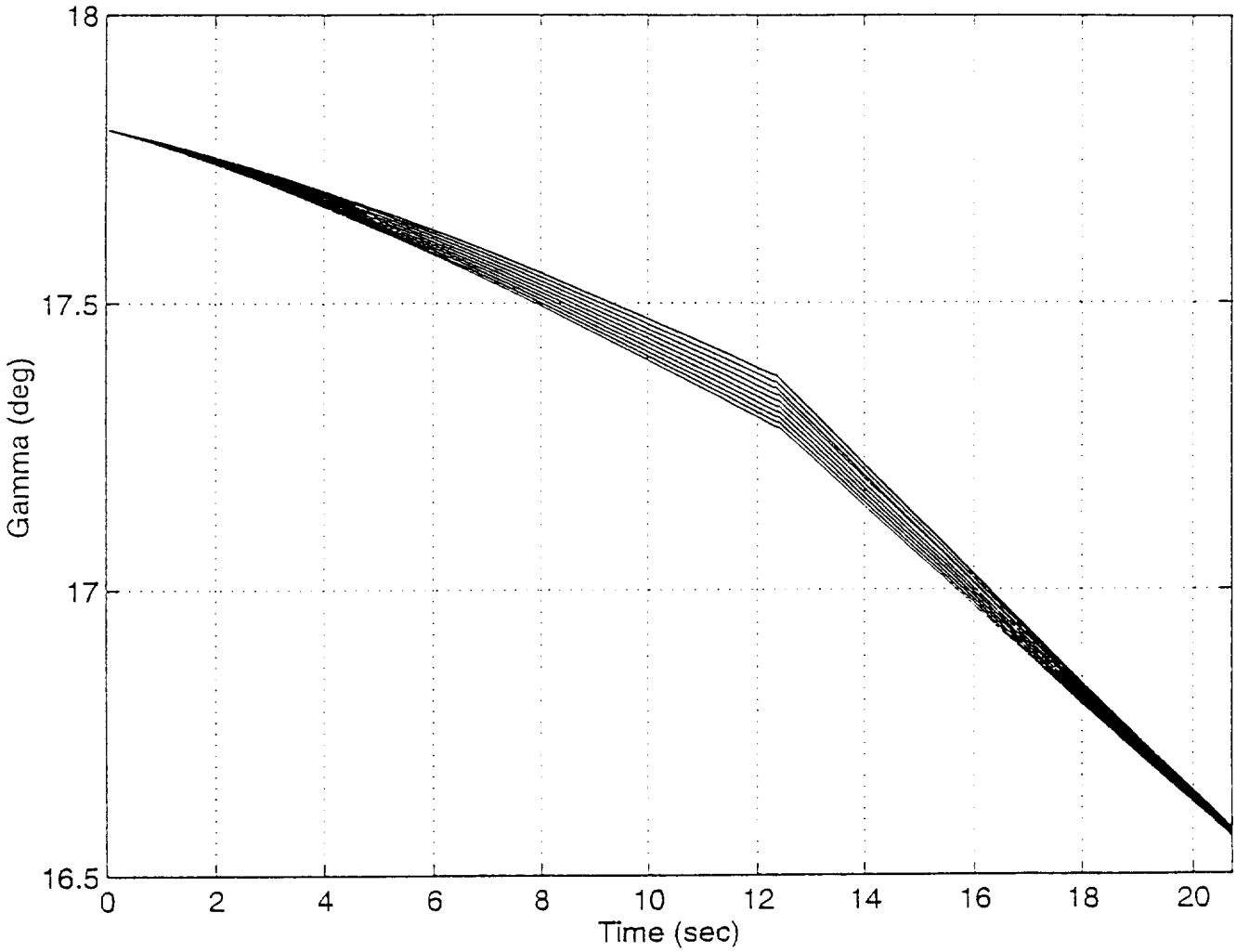

Figure 9: Gamma History
(with changes in initial mass)

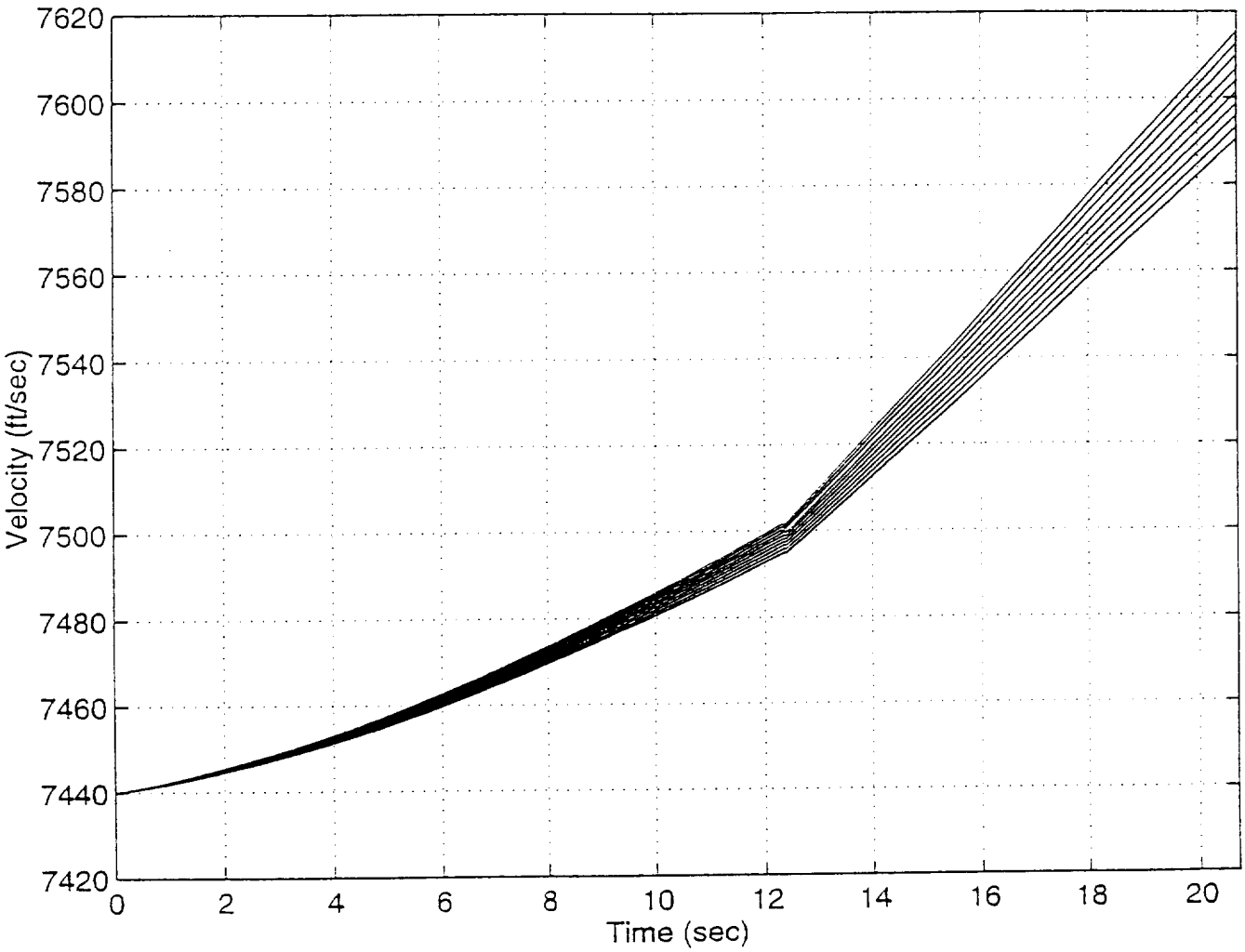

Figure 10: Velocity History
(with changes in initial mass)

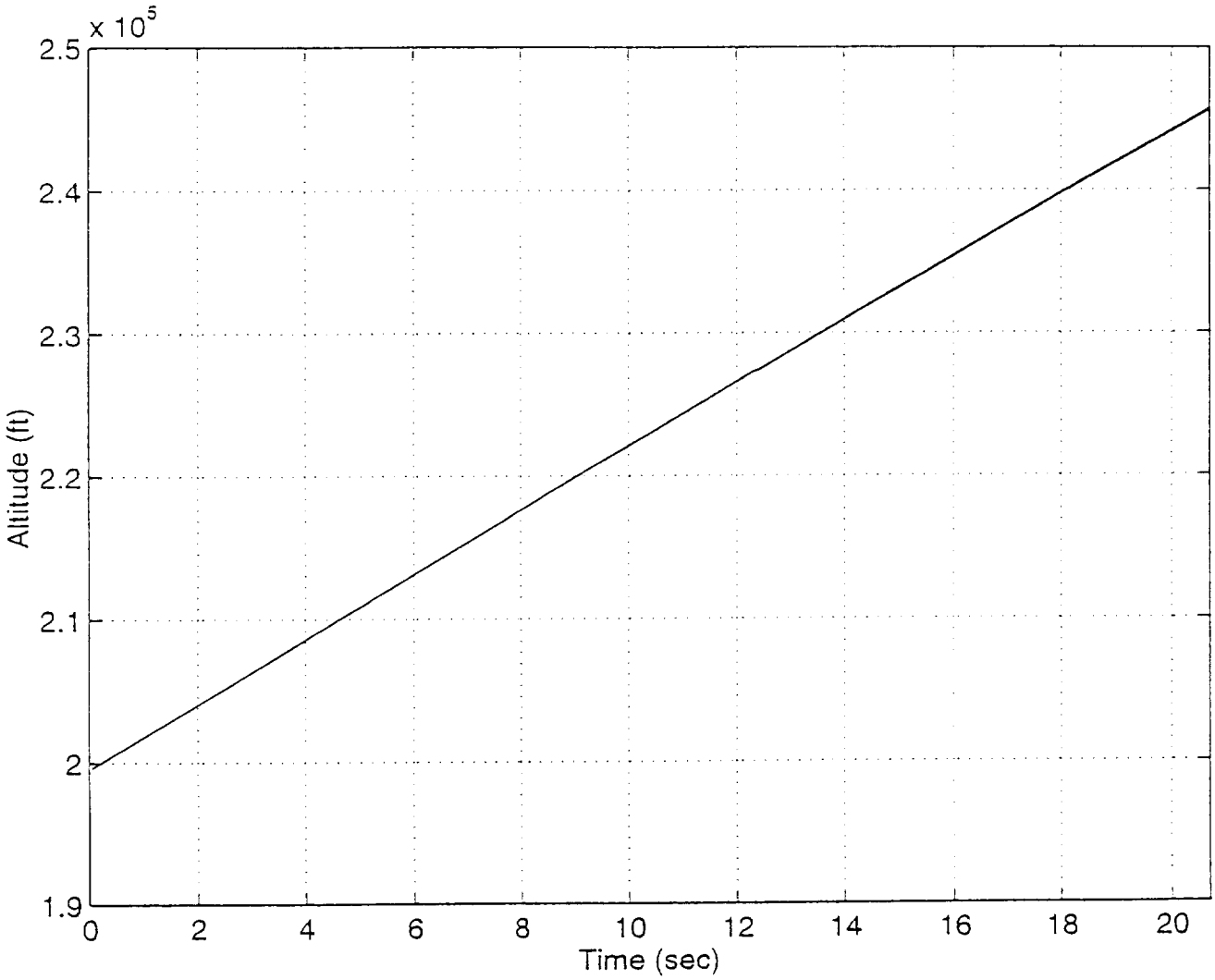

Figure 11: Altitude History
(with changes in initial mass)

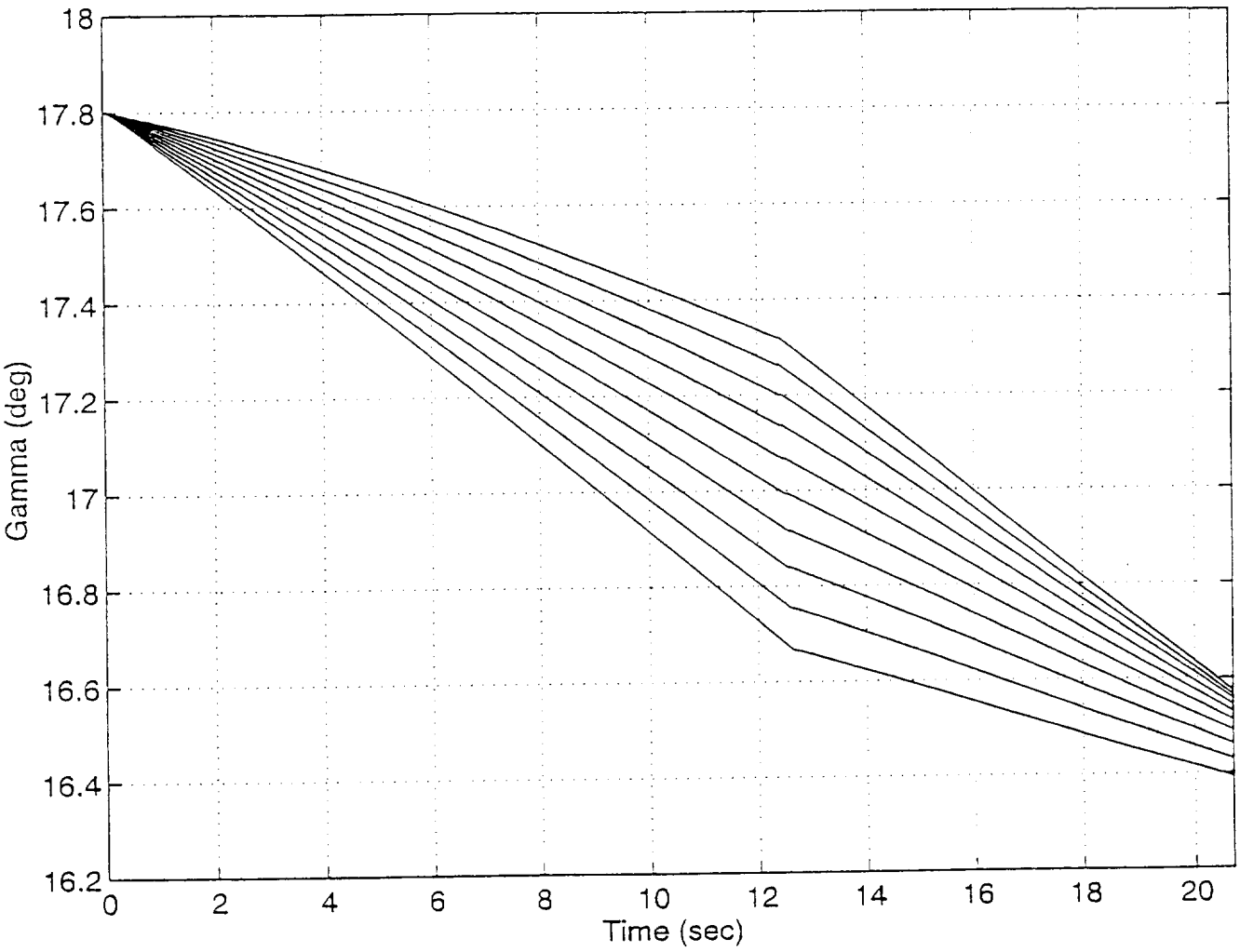

Figure 12: Gamma History
(with changes in initial velocity)

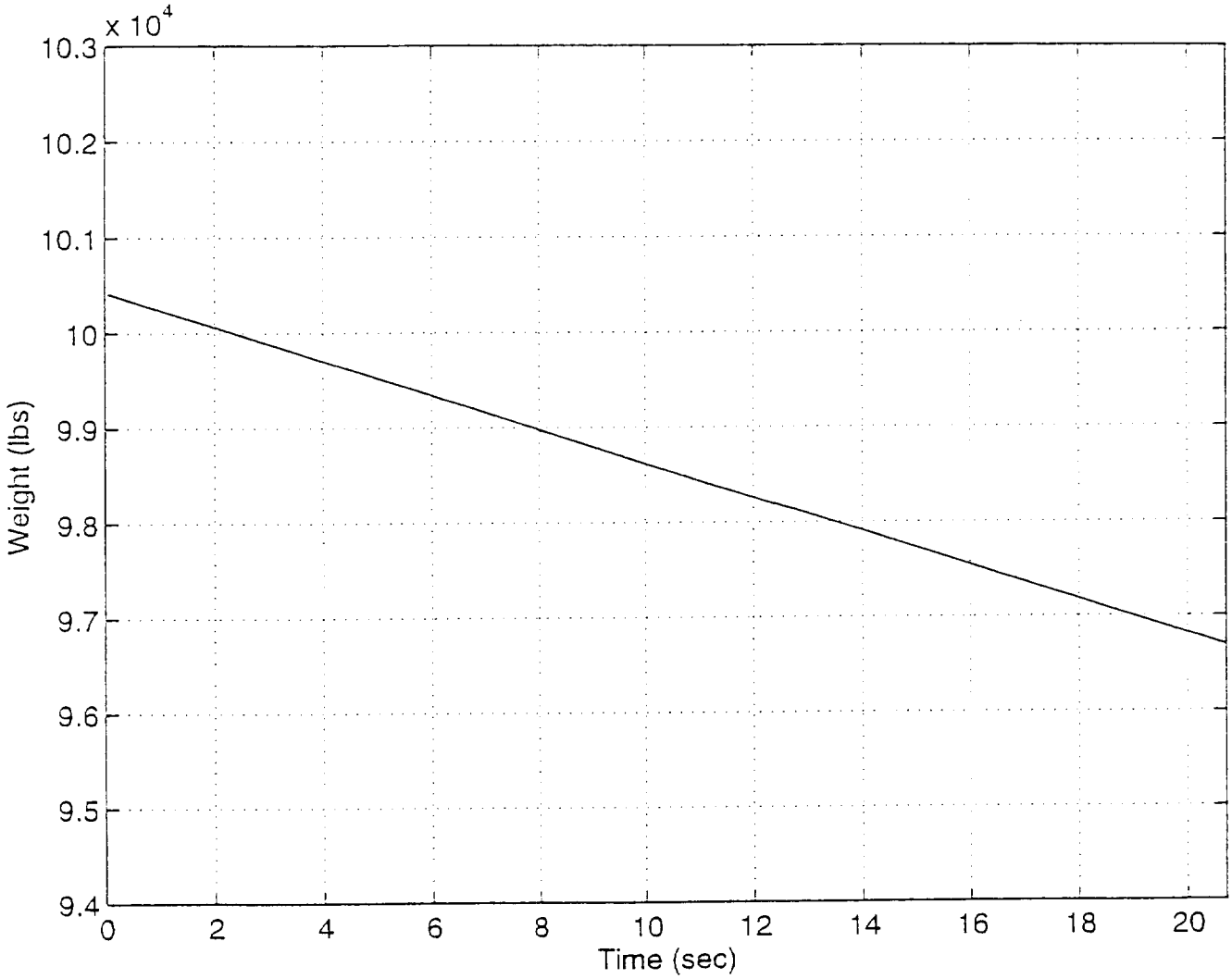

Figure 13: Weight History
(with changes in initial velocity)

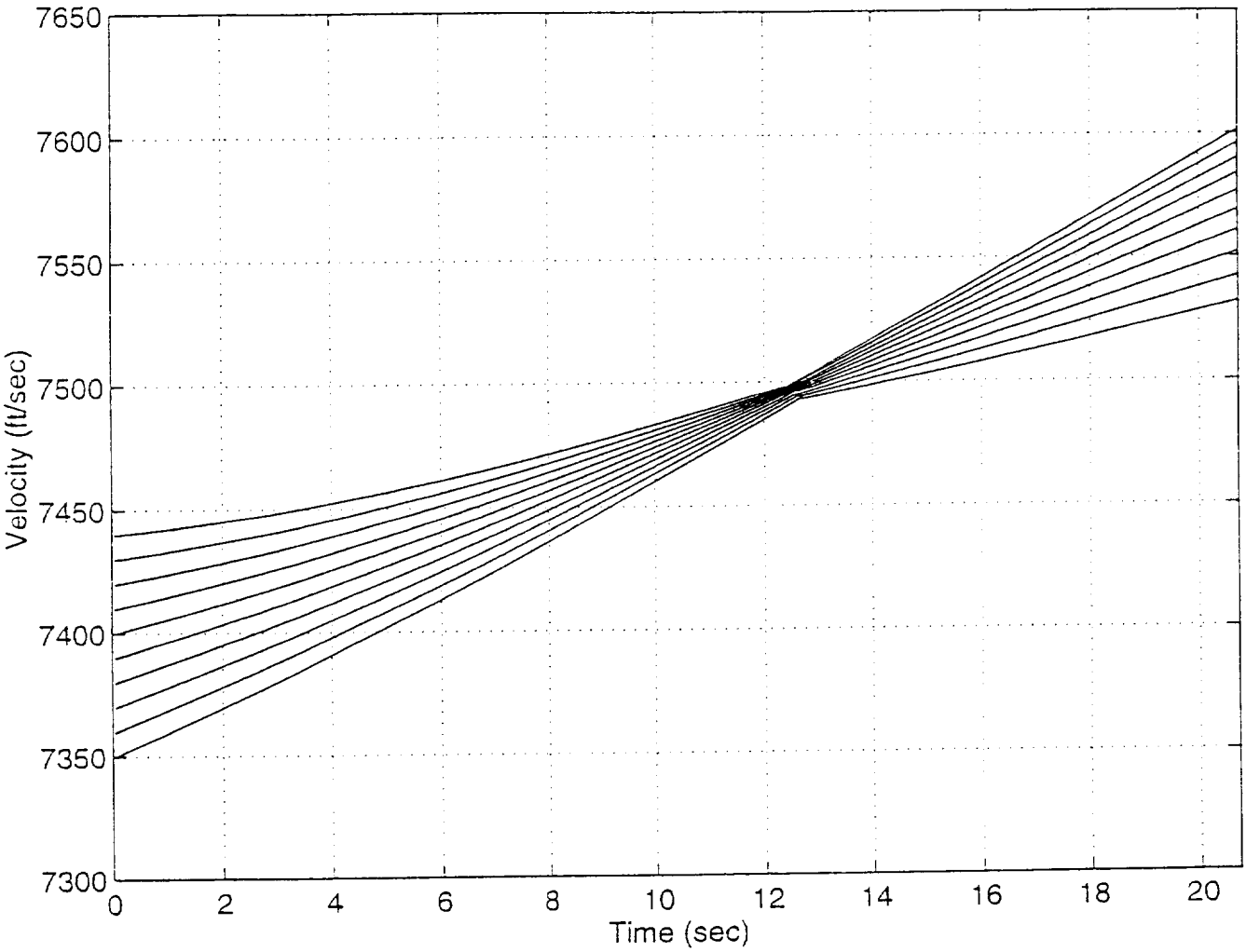

Figure 14: Velocity History
(with changes in initial velocity)

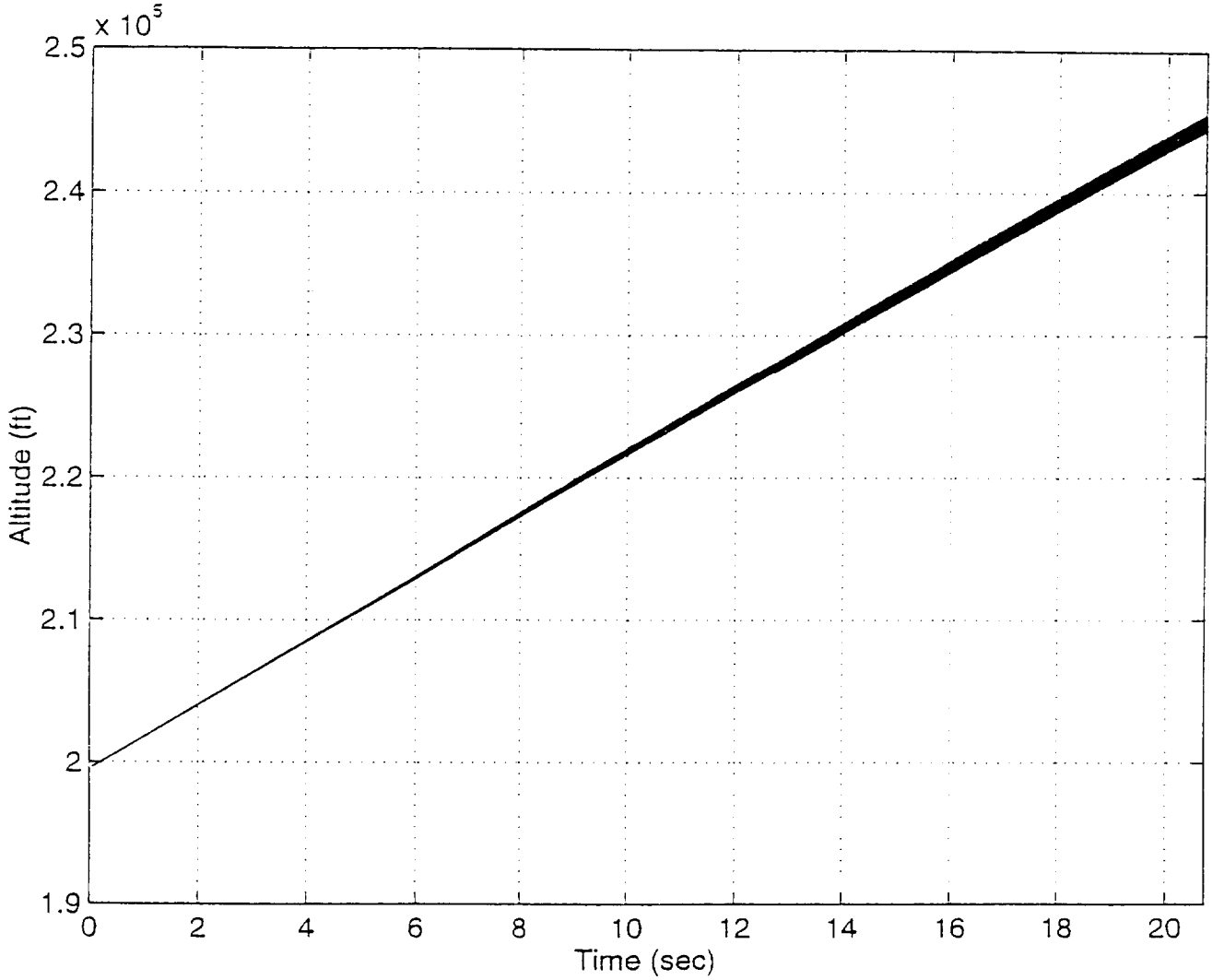

Figure 15: Altitude History
(with changes in initial velocity)

APPENDIX

A Class of Modified Hopfield Networks for Aircraft Identification and Control

Jie Shen S. N. Balakrishnan*
Department of Mechanical and Aerospace Engineering
and Engineering Mechanics
University of Missouri-Rolla
Rolla, MO 65401

(573)341-4675

Abstract

This paper presents a class of modified Hopfield neural networks and their use in solving aircraft optimal control and identification problems. This class of networks consists of parallel recurrent networks which have variable dimensions that can be changed to fit the problems under consideration. It has a structure to implement an inverse transformation that is essential for embedding optimal control gain sequences. Equilibrium solutions are discussed. Energy minimization of the networks leads to identification of the system parameters. Numerical results are provided to identify the dynamics of an aircraft, and the corresponding optimal control is calculated on-line. Comparison of the neural network solutions with point-wise optimal control using LQR formulation for this multivariable control problem shows near identical results throughout the trajectories.

1 Introduction

There has been a spurt of activities in the area of artificial neural networks (ANN) during the last ten years. For a survey of the ANN work done in the areas of identification and control, see bibliography. There are two types of networks used in almost all ANN applications. The first is the more widespread feedforward network and the second is a less understood recurrent network. The feedforward networks where data flow is unidirectional are essentially static; the recurrent networks, on the other hand, are based on feedback connections. Due to feedback connections, the recurrent networks are better suited for control problems which are based on closed-loop solutions.

In this paper, a variation of the Hopfield network is proposed. Compared to the classic Hopfield network, it keeps the characteristic of energy minimization, which is used to minimize the identification errors. The mean-square error is used as a performance criterion in system identification, and is formulated in an energy form to utilize the network functionality. Based on the equilibrium analysis, these networks can perform an inverse transformation on matrices and other auxiliary mathematical operations. This feature allows the networks to give out optimal control gain sequences based on the identified system parameters. In addtion, this class of networks has more degrees of freedom than the classic Hopfield networks. The network architecture can be augmented according to the problems at hand.

The modified Hopfield network is analyzed in section 2. Its identification application is presented in section 3, while the control application is in section 4. Both the principles and examples are given in

*Associate Fellow, AIAA (to whom all correspondence should be sent)

each individual section. Conclusions are presented in section 5.

2 Modified Hopfield Networks

2.1 Stability

The modified Hopfield network is a variant of the classical Hopfield network. Fig (1) shows its basic features.

We will demonstrate its stability by analyzing its dynamics and using energy function. The network has two clusters of neurons. The right part of the networks is characterized by outputs Φ_j which are nonlinear functions f of their state u_j

$$\Phi_j = f(u_j) \tag{1}$$

where

$$u_j = \sum_{i=1}^{n} w_{ji} v_i - b_j, \qquad j = 1, 2, \ldots, m \tag{2}$$

with b_j the exogenous input current, and v_i the output of the left cluster of amplifiers. Conductance w_{ij} connects the output of the j'th neuron to the input of the i'th neuron, which are indicated in Fig (1) as ■.

The left part of the networks is characterized by the dynamics. The amplifiers have input conductances and capacitances denoted as g_i and c_i, respectively. They both represent the amplifiers' parasitic input impedance and are responsible for the appropriate time-domain behavior of the entire network. At the same time, we assume that the response time of $\Phi(u_j)$ is negligibly small compared to that of the amplifiers $g(u_i)$.

Under these assumptions, *Kirchhoff's law* gives us

$$C_i \frac{du_i}{dt} = -a_i - G_i u_i - \sum_{j=1}^{m} w_{ji} \Phi_j, \qquad (i = 1, 2, \ldots, n) \tag{3}$$

where G_i denotes the sum of all conductances connected to the input of the ith neuron and is equal to

$$G_i \triangleq g_i - \sum_{j=1}^{n} w_{ij} \tag{4}$$

and a_i is the exogenous input current.

Using Equation (1), the above formula can be expressed as follows

$$C_i \frac{du_i}{dt} = -a_i - G_i u_i - \sum_{j=1}^{m} w_{ji} f\left(\sum_{k=1}^{n} w_{ki} v_k - b_j\right) \qquad (i = 1, 2, \ldots, n) \tag{5}$$

We now define the following *Liapunov* function as an energy function $E$for the modified Hopfield networks

$$E(v) \triangleq \sum_{k=1}^{n} a_k v_k + \sum_{j=1}^{m} F\left(\sum_{k=1}^{n} w_{kj} v_k - b_j\right) + \sum_{i=1}^{n} G_i \int_0^{v_i} g^{-1}(v) dv \tag{6}$$

Define

$$f(z) = \frac{dF(z)}{dz} \tag{7}$$

The components of the gradient vector of the assumed energy function (6) can be expressed by finding its derivatives as follows

$$\frac{\partial E(v)}{\partial v_i} = a_i + G_i u_i + \sum_{j=1}^{m} w_{ji} f\left(\sum_{k=1}^{n} w_{kj} v_k - b_j\right) \tag{8}$$

The time derivative of the energy function can now be expressed using the above equations

$$\begin{aligned} \frac{dE}{dt} &= \sum_{i=1}^{n} \frac{dv_i}{dt} \left(a_i + u_i G_i + \sum_{j=1}^{m} w_{ji} \cdot f\left(\sum_{k=1}^{n} w_{kj} v_k - b_j\right) \right) \\ &= -\sum_{i=1}^{n} C_i \frac{dv_i}{dt} \cdot \frac{du_i}{dt} \\ &= -\sum_{i=1}^{n} C_i g^{-1\prime}(v_i) \left(\frac{dv_i}{dt}\right)^2 \end{aligned} \tag{9}$$

Since $C_i > 0$, and $g^{-1}(v_i)$ is a monotonically increasing function n, the sum on the right sight of (9) is nonnegative, and therefore w e have $dE/dt \leq 0$, unless $dv_i/dt = 0$, in which case $dE/dt = 0$. This means that the evolution of dynamic system (5) in state space always seeks the minima of the energy surface E. Integration of Eqs. (5) and (6) shows that the outputs v_j do follow gradient descent paths on the E surface.

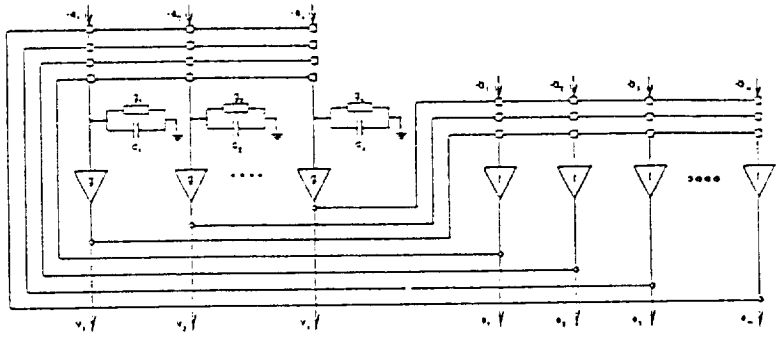

Figure 1: Modified Hopfield Networks

2.2 Solution

In order to get the analytic expression for the converged value of the networks, we assume small signals and that they work in the linear region of the amplifier. Note that in the above derivation, there is no difference if we denote the connection matrices in the left and right adjoint subnets separately. These connection matrices are nothing but the weights w_{ij}. Let the right connection matrix be D_1, and the left connection matrix be D_2, the stability conclusion still holds. Under these mild assumptions, and with *Kirchhoff*'s law, we can have a relation in a matrix form as

$$\begin{aligned} C\frac{dU}{dt} &= -a - GU - D_1^T Q \qquad (10) \\ Q &= K_2(D_2 V - b) \\ &= K_2(K_1 D_2 U - b) \qquad (11) \end{aligned}$$

where a and b are the exogenous inputs of the adjoint networks G and F respectively. U is the input to G and V is the output of G. We also assume that all amplifier gains K_1 in G are equal. Similarly the gains of amplifiers in F are K_2. K_1 and K_2 are scalars. Substitute Equation (11) into Equation (10) to get

$$\begin{aligned} C\frac{dU}{dt} &= -a - GU - \\ &\qquad D_1^T K_2(K_1 D_2 U - b) \qquad (12) \\ &= -(G + K_2 K_1 D_1^T D_2)U \\ &\qquad + K_1 D_1^T b - a \qquad (13) \end{aligned}$$

When the networks reach equilibrium, $dU/dt = 0$, and

$$\begin{aligned} V &= K_2 U \\ &= \left(D_1^T D_2 + \frac{G}{K_1 K_2}\right)^{-1} \left(D_1^T b - \frac{a}{K_1}\right) \qquad (14) \end{aligned}$$

2.3 Discussion

Equation (14) gives the general solution for the modified Hopfield networks. Compared with the classical Hopfield networks, an obvious feature is that it involves more parameters. We may find some applications in which these parameters can be taken advantage of. Also some of them can be avoided depending upon the desired objective.

Note we get two factors involved in the inverse operation. As a result, the structure of this kind of recurrent networks is quite flexible. while the classical Hopfield is self-recurrent, that is, it feeds back its own output; the variation is mutually recurrent, that is, it feeds back the outputs of its two-adjoint

parts. This architecture can be expanded further with ease to three or four subnets or several layers as needed. Some special applications may need that computational relationship, but it is not needed for the application considered here.

The dimensions of parameters a, b, D_1 and D_2 depend on the applications. K_1 and K_2 also can be designed to provide appropriate magnitudes. If K_1 is large, then G and a will both have less effect on the output V or ignorable. If we want a have reasonable influence in the expression while G should not, then we design K_2 large, and determine K_1 according to the requirements on a.

3 System Identification

3.1 Problem Formulation

The proposed structure for system identification in the time domain is shown in Fig (2). The dynamics of a linear plant (to be identified) are defined by the usual equations, where A_p and B_p are unknown matrices and x and u are the state and control respectively.

$$\dot{\mathrm{x}} = \mathrm{A_p x} + \mathrm{B_p u} \tag{15}$$

The dynamic equation of the system model depends on e, which is the error vector between actual system states x and estimated values y.

$$\dot{\mathrm{y}} = \mathrm{A_s}(\mathrm{e}, t)\mathrm{x} + \mathrm{B_s}(\mathrm{e}, t)\mathrm{u} - \mathrm{Ke} \tag{16}$$

Therefore, the error dynamics equation is a function of state and control.

$$\dot{\mathrm{e}} = (\mathrm{A_p} - \mathrm{A_s})\mathrm{x} + (\mathrm{B_p} - \mathrm{B_s})\mathrm{u} + \mathrm{Ke} \tag{17}$$

The goal is to minimize simultaneously square-error rates of all states utilizing a Hopfield network. To ensure global convergence of the parameters, the energy function of the network must be quadratic in terms of the parameter errors, $(\mathrm{A_p} - \mathrm{A_s})$ and $(\mathrm{B_p} - \mathrm{B_s})$. However, the error rates $\dot{\mathrm{e}}$ in Eq. (17) are functions of the parameter errors and the state errors. The state error depends on y, which, in turn, is influenced by $\mathrm{A_s}$ and $\mathrm{B_s}$. Hence, an energy function based on $\dot{\mathrm{e}}$ will have a recurrent relation with $\mathrm{A_s}$ and $\mathrm{B_s}$. To avoid this, we use the following energy function, where *tr* defines the trace of a matrix, and $(\cdot)^T$ is the transpose of matrix. (see, Raol, Bibliography)

$$\begin{aligned} E &= \frac{1}{T}\int_0^T \frac{1}{2}\dot{\mathrm{e}}_q(t)^T\dot{\mathrm{e}}_q(t)dt \\ &= \frac{1}{T}\int_0^T \frac{1}{2}(\dot{\mathrm{x}} - \mathrm{A_s x} - \mathrm{B_s u})^T \\ &\quad \cdot(\dot{\mathrm{x}} - \mathrm{A_s x} - \mathrm{B_s u})dt \end{aligned} \tag{18}$$

In order to facilitate the derivation, we expand the items in the factors of the energy function, and utilize the trace identities to simplify.

$$\begin{aligned} E &= tr\left(\mathrm{A_s}\left[\frac{1}{T}\int_0^T \frac{1}{2}\mathrm{x}\mathrm{x}^T dt\right]\mathrm{A_s}^T\right) \\ &+ tr\left(\mathrm{B_s}\left[\frac{1}{T}\int_0^T \frac{1}{2}\mathrm{u}\mathrm{u}^T dt\right]\mathrm{B_s}^T\right) \\ &+ tr\left(\mathrm{A_s}\left[\frac{1}{T}\int_0^T \mathrm{x}\mathrm{u}^T dt\right]\mathrm{B_s}^T\right) \\ &- tr\left(\mathrm{A_s}\left[\frac{1}{T}\int_0^T \mathrm{x}\dot{\mathrm{x}}^T dt\right]\right) \\ &- tr\left(\mathrm{B_s}\left[\frac{1}{T}\int_0^T \mathrm{u}\dot{\mathrm{x}}^T dt\right]\right) \\ &+ \left[\frac{1}{T}\int_0^T \frac{1}{2}\dot{\mathrm{x}}^T\dot{\mathrm{x}}dt\right] \end{aligned} \tag{19}$$

Equation (19) is quadratic in terms of $\mathrm{A_s}$ and $\mathrm{B_s}$. Substituting $\mathrm{A_p x} + \mathrm{B_p u}$ for $\dot{\mathrm{x}}$ in Eq. (19) indicates that E is also a quadratic function of the parameter errors. Based on Eq. (19), we can program a Hopfield network that has neurons with their states representing different elements of the $\mathrm{A_s}$ and $\mathrm{B_s}$ matrices. From the convergence properties of the Hopfield network, the equilibrium state is achieved when the partial derivatives $\partial E/\partial \mathrm{A_s}$ and $\partial E/\partial \mathrm{B_s}$ are zero. We use the following identitiesto find the partial derivatives of E.

$$\frac{\partial}{\partial A}tr\,(ABA^T) = 2AB \tag{20}$$

$$\frac{\partial}{\partial A}tr\,(ABD) = B^T D^T \tag{21}$$

This results in the following, where $\mathrm{A_s^*}$ and $\mathrm{B_s^*}$ are optimum solutions of the estimation problem.

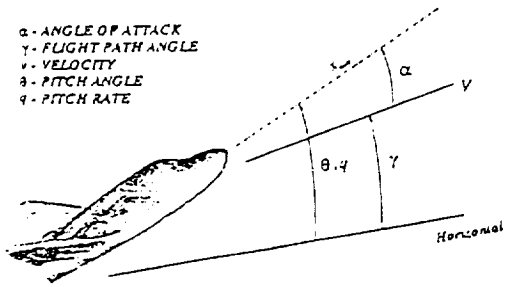

Figure 3: Schematic of Longitudinal Flight

Define,

$$[w_{ij}] = \frac{1}{T}\int_0^T \begin{bmatrix} xx^T & 0 & 0 & 0 & xu & 0 & 0 & 0 \\ 0 & xx^T & 0 & 0 & 0 & xu & 0 & 0 \\ 0 & 0 & xx^T & 0 & 0 & 0 & xu & 0 \\ 0 & 0 & 0 & xx^T & 0 & 0 & 0 & xu \\ ux^T & 0 & 0 & 0 & u^2 & 0 & 0 & 0 \\ 0 & ux^T & 0 & 0 & 0 & u^2 & 0 & 0 \\ 0 & 0 & ux^T & 0 & 0 & 0 & u^2 & 0 \\ 0 & 0 & 0 & ux^T & 0 & 0 & 0 & u^2 \end{bmatrix} dt$$

$$[a_i] = \frac{1}{T}\int_0^T \begin{bmatrix} \dot{x}_1 x^T & \dot{x}_2 x^T & \dot{x}_3 x^T & \dot{x}_4 x^T & u\dot{x}^T \end{bmatrix}^T dt \quad (29)$$

With these as weights and biases of the networks, a_{ij}, and b_j can be solved through Eqs. (27) and (28). Derivation of $[w_{ij}]$ and $[a_i]$ assumes that the neuron input conductance, G_i, is low enough so that the the second term in Eq. (3) can be neglected.

3.2 Numerical Example

We present a representative numerical example to validate the capacities of the modified Hopfield networks. The orientation of an aircraft involving longitudinal dynamics is shown in Fig (3). The linearized equations of motion of an aircraft in a vertical plane are given by

$$\dot{x} = Ax + Bu \quad (30)$$

where, the elements of the state space x are

$$x = [u' \quad \alpha \quad \theta \quad q]^T \quad (31)$$

The matrix A represents the dynamic stability derivatives and is given by

$$A_p = \begin{bmatrix} -0.0148 & -13.88 & -32.2 & 0 \\ -0.00019 & -0.84 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0.00005 & -4.8 & 0 & -0.5 \end{bmatrix}$$

The matrix B represents the control derivatives and is given by

$$B_p = \begin{bmatrix} -1.1 \\ -0.11 \\ 0 \\ -8.74 \end{bmatrix}$$

The control variable u represents elevator deflection.

Fig (4) shows the simulation results of the system identification. These figures represent only A_{p11}, A_{p12}, B_{p2}, and B_{p4} histories; similar results can be obtained for other elements of the A_p and B_p matrices. From the numerical results shown in Fig (4), it is clear that the network is able to identify system parameters very well.

4 Optimal Control Application

4.1 Problem Formulation

Let the plant to be controlled be described by the linear equation

$$x_{k+1} = A_k x_k + B_k u_k \quad (32)$$

with $x_k \in R^n$ and $u_k \in R^m$. The associated performance index is the quadratic function

$$J_i = \frac{1}{2} x_N^T S_N x_N + \frac{1}{2}\sum_{k=i}^{N-1} \left(x_k^T Q_k x_k + u_k^T R_k u_k \right) \quad (33)$$

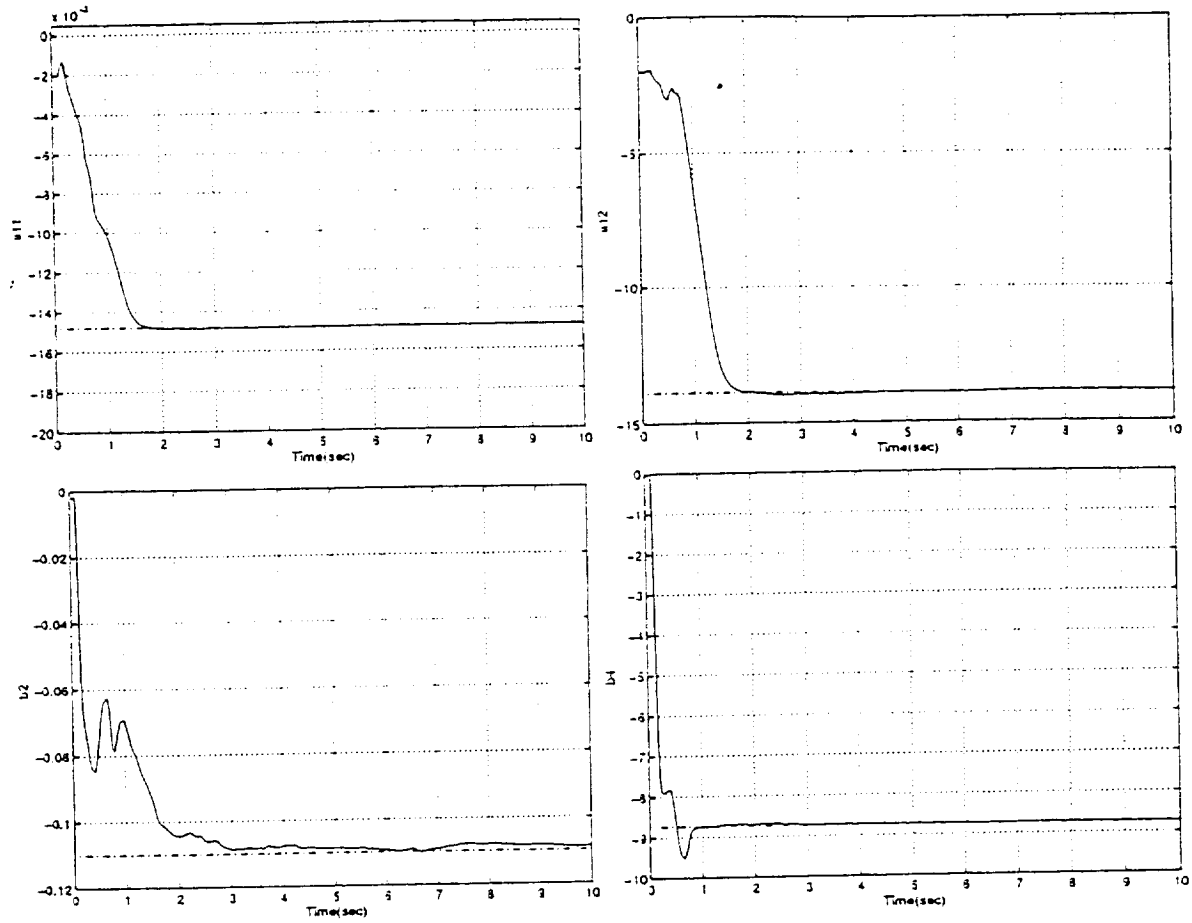

Figure 4: Identification History

defined over the time interval of interest $[i, N]$. Note that both the plant and the cost-weighting matrices can be time-varying. The initial plant state is given as x_i. We assume that Q_k, R_k and S_N are symmetric positive semidefinite matrices, and in addition that $|R_k| \neq 0$ for all k.

The objective is to find the control sequence $u_i, u_{i+1}, \ldots, u_{N-1}$ to minimize J_i.

To solve this linear quadratic regulator (LQR) problem, we begin with the Hamiltonian function

$$H^k = \frac{1}{2}\left(x_k^T Q_k x_k + u_k^T R_k u_k\right) + \lambda_{k+1}^T \left(A_k x_k + B_k u_k\right) \tag{34}$$

Then we can get the state and costate equations

$$x_{k+1} = \frac{\partial H_k}{\partial \lambda_{k+1}} = A_k x_k + B_k u_k \tag{35}$$

$$\lambda_k = \frac{\partial H_k}{\partial x_k} = Q_k x_k + A_k^T \lambda_{k+1} \tag{36}$$

and the stationarity condition

$$0 = \frac{\partial H_k}{\partial u_k} = R_k u_k + B_k^T \lambda_{k+1} \tag{37}$$

This procedure will finally lead to the control,

$$u_k = -K_k x_k, \qquad k < N$$

where the Kalman gain K_k is given by

$$K_k = \left(B_k^T S_{k+1} B_k + R_k\right)^{-1} B_k^T S_{k+1} A_k \tag{38}$$

In terms of the Ricatti variable S_k, now

$$S_k = A_k^T S_{k+1} \left(A_k - B_k K_k\right) + Q_k \tag{39}$$

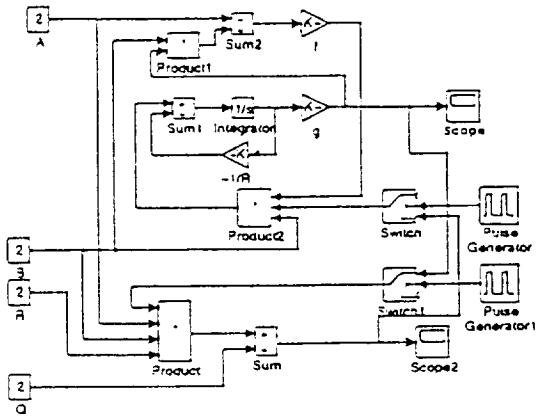

Figure 5: Simulation plot

In the application where the control interval is finite, S_N will be given. Alternatively use Equation (38) and (39), we will get a series of K_k. The gain matrix K_k will generally be time-varying even when the matrices A_k, B_k, Q_k and R_k are all constant. But if the control interval is infinite, the above formulation need to be changed a little.

4.2 Network Solution/Implementation

We briefly discuss the recurrent network solution for optimal gain sequence.

Based on the recursions in Equations (38) and (39), the most commonly encountered operations are scalar and outer product vector multiplications and matrix-vector multiplication. But the crucial operation here is the inverse to get the Kalman gain.

The modified Hopfield networks contain both invariant and variable parameters. Invariant parameters are fixed in the neuron-computing model, while variable parameters can be modified. By comparing Eqs. (38) and (39) with the stable output of the network Eq. (14), if we set $D_1{}^T = B_k^T S_{k+1}$, $D_2 = B_k$, $\frac{G}{K_i K_f} = R_k$, and $b = A_k$, $a = 0$, the network will give us the Kalman sequence. As we know, it is not difficult for the circuits to achieve the multiplication of two signals. However, since D_1 and D_2 are connection conductances, can they be changed by other signals like $B_k^T S_{k+1}$ and B_k?

The answer is a voltage-controlled switch. A voltage-controlled switch can be implemented using a single field-effect or MOS transistor operating in the resistive (ohmic, also called linear) region. So, all the signals are preferred to be voltage signals. The system parameters A_k and B_k are generally the outputs of identification modules which are convenient to be given out as voltages. The optimal control formulation does not limit the A_k, B_k, Q_k and R_k matrices to be constants and the modified Hopfield Hopfield network doesn't limit its capacities either. Time-varying A_k, B_k etc. are easy to be feed into the net as voltage signals to be used in the computations.

4.3 Numerical Example

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 niminizing control is expected to drive the deviations of the longitudinal dynamics in pitch angle θ, pitch rate q, forward velocity u', and andle of attack α to zero.

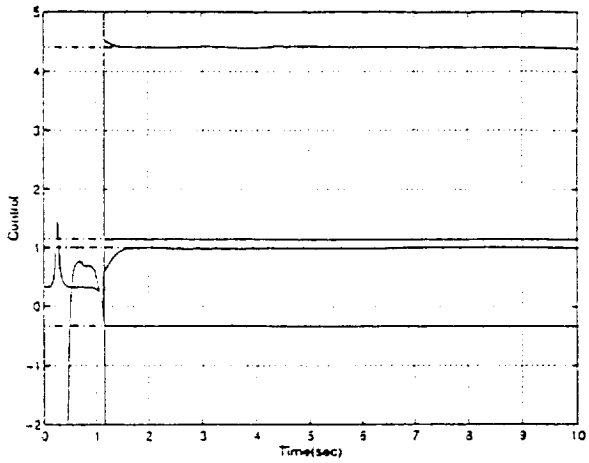

Figure 6: Control History

The system parameters are the same as identification. The performance index has the form

$$J = \int_0^\infty \left(\mathrm{x}^T Q \mathrm{x} + u^T R u\right) dt \qquad (40)$$

where Q, and R are appropriate weighting matrices. We select $R = 91.32$ and

$$Q = \begin{bmatrix} 10.37 & 0 & 0 & 0 \\ 0 & 0.0004 & 0.0016 & 0 \\ 0 & 0.0016 & 7.25 & 0 \\ 0 & 0 & 0 & 14.84 \end{bmatrix}$$

The simulation plot is shown is Fig (5). The controls which are calculated by networks, compared with LQR results are shown in Fig (6). The states trajectories are shown in Fig (7). The controls are applied at 2 seconds.

5 Conclusion

A class of modified Hopfield networks has been presented to solve parameter identification and optimal control problems. The architectures are designed to suit an energy minimization for system identification and a typical optimal control algorithm for system control. Similar to the Hopfield network, the stability of these modified networks is guaranteed. But they provide more degrees of freedom and flexibility to accommodate different applications. A four-dimensional aircraft control problem is identified and optimal control is obtained as illustrations of these approaches. Future work on this topic will investigate the robustness of such network controllers and the use of these methods for other relevant applications.

* * * *

ACKNOWLEDGMENT

This study was partially funded by NSF Grant ECS-9313946, the Missouri Department of Economic Development Center for Advanced Technology Program and by NASA Grant NAG1-1728.

Bibliography

1. Balakrishnan, S.N. and Weil, R.D., "Neurocontrol: A literature Survey", *Mathl. Comput. Modelling*, Vol. 23, No. 1/2 pp. 101-117, 1996.

2. Hunt, N.F., Sbarbaro, D., Zbikowski, R. and Gawthrop, P.J., "Neural Networks for Control System – A Survey," *Automatica*, Vol. 28, No. 6, pp. 1083-1112, 1992.

3. Hopfield, J.J., and D. W. Tank. 1986. "Computing with Neural Circuits: A Model," *Science* 233: 625-633.

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

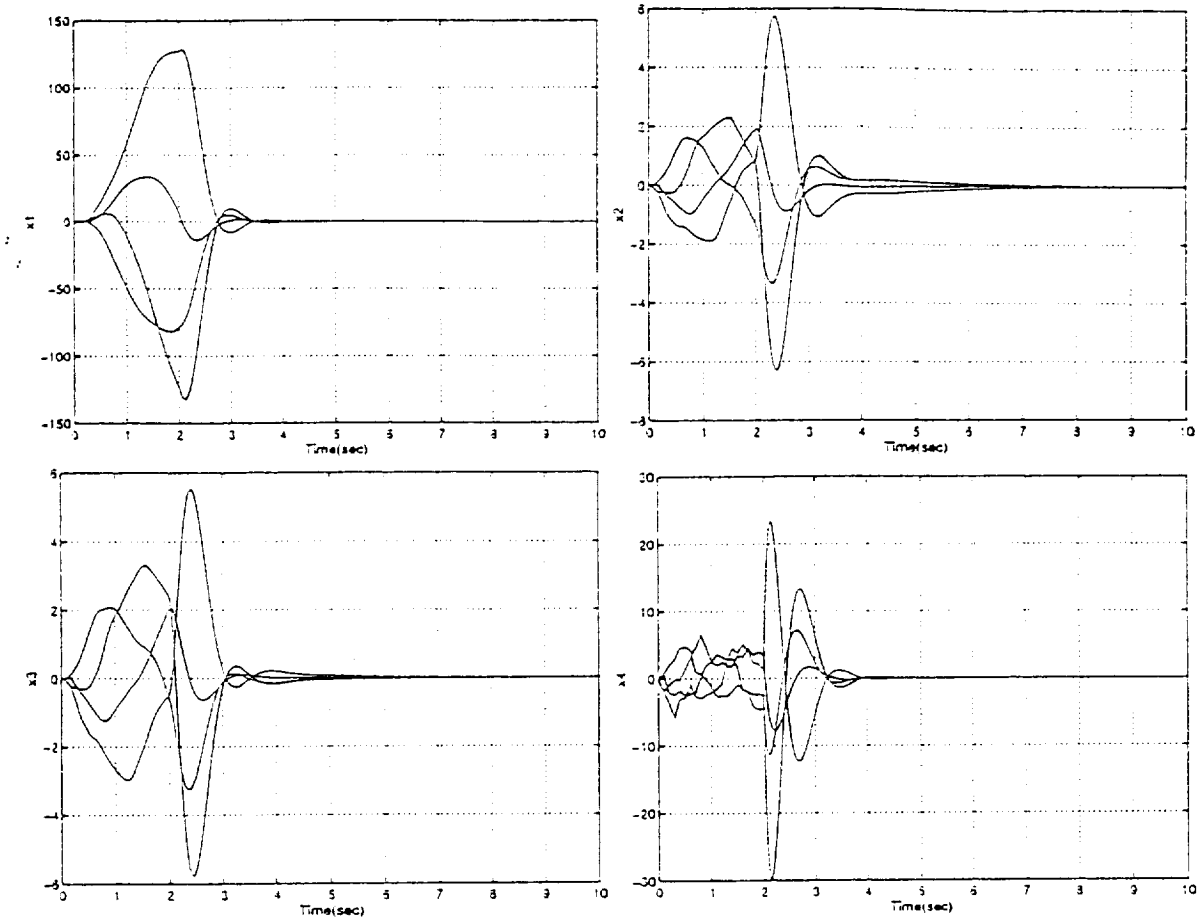

Figure 7: States Trajectories

5. White, D.A. and Sofge, D.A., *Handbook of Intelligent Control - Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold, New York, 1992.

6. Raol, J.R., *Parameter estimation of state space models by recurrent neural networks*, IEE Proc.-Control Theory Appl., Vol. 142, No. 2, pp114118, March 1995

7. Narendra, K.S, and Parthasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Trans. on Neural Networks*, Vol. 1, No. 1, pp. 4-26, March, 1990.

8. DARPA *Neural Network Study*, Fairfax, Virginia: AFCEA Int. Press, 1988

9. Hopfield, J.J., "The Effectiveness of Analogue 'Neural Network' Hardware," *Network* 1: 27-40, 1990

10. Hopfield, J.J., "Neurons with Graded Response Have Collective Computational Properties Like Those of Two State Neurons," *Proc. National Academy of Sciences* 81: 3088-3092, 1984

11. Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. National Academy of Sciences* 79: 2554-2558, 1982

12. *Special Section on Neural Networks for Control Systems*, IEEE Cont. Sys. Mag., Vol. 9, No. 3, pp. 25-59, April 1989.

13. *Special Issue on Neural Networks in Control Systems*, IEEE Cont. Sys. Mag., Vol. 10, No. 3, pp 3-87, April 1990.

14. Kamp, Y., and Hasler, M., *Recursive Neural Networks for Associative Memory*, Chichester, U.K.: John Wiley & Sons, 1990

15. Friedland, B., *Control System Design*, McGraw-Hill Book Company, 1986

16. Simulink, *Dynamic System Simulation Software For the X Window System*, The Math Works Inc., 1993