1-1-2002

State-Constrained Agile Missile Control with Adaptive-Critic-Based Neural Networks

Dongchen Han

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

Recommended Citation

D. Han and S. N. Balakrishnan, "State-Constrained Agile Missile Control with Adaptive-Critic-Based Neural Networks," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 4, pp. 481-489, Institute of Electrical and Electronics Engineers (IEEE), Jan 2002.
The definitive version is available at https://doi.org/10.1109/TCST.2002.1014669

State-Constrained Agile Missile Control With Adaptive-Critic-Based Neural Networks

Dongchen Han and S. N. Balakrishnan

***Abstract*—In this study, we develop an adaptive-critic-based controller to steer an agile missile that has a constraint on the minimum flight Mach number from various initial Mach numbers to a given final Mach number in minimum time while completely reversing its flightpath angle. This class of bounded state space, free final time problems is very difficult to solve due to discontinuities in costates at the constraint boundaries.We use a two-neural-network structure called "adaptive critic" in this study to carry out the optimization process. This structure obtains an optimal controller through solving optimal control-related equations resulting from a Hamiltonian formulation. Detailed derivations of equations and conditions on the constraint boundary are provided. For numerical experiments, we consider vertical plane scenarios. Flight Mach number and the flightpath angle are the states and the aerodynamic angle of attack is treated as the control. Numerical results bring out some attractive features of the adaptive critic approach and show that this formulation works very well in guiding the missile to its final conditions for this state constrained optimization problem from an *envelope* of initial conditions.**

***Index Terms*—Missile guidance and control, neural networks, optimal control.**

I. Introduction

IN ORDER to explore and extend the range of operations of air-to-air missiles, there have been studies in recent years with a completely different concept. It consists of launching the missile as usual from the aircraft and guiding it to intercept a target in the rear hemisphere (see Fig. 1). The best emerging alternative to execute this task is to use the aerodynamics and thrust to turn around the initial flight path angle of zero to a final flight path angle of 180°. (Every scenario can be considered as a subset of this set of extremes in flightpath angle.) In this study, we formulate an optimal controller to achieve this mission in minimum time with a constraint on the minimum flight Mach number. This problem falls under a class called "*free final time state-constrained*" problems in calculus of variations (optimal control) which for an *envelope of initial conditions* is difficult to solve. To our knowledge, there is no publication dealing with this difficult problem other than with using single set of initial conditions.

Manuscript received December 10, 1999; revised June 1, 2001. Manuscript received in final form April 2, 2002. Recommended by Associate Editor S. Banda. This work was supported in part by the U.S. Air Force under Grant F08630-96-1-0001 and the National Science Foundation under Grant ECS-9634127.

The authors are with the Department of Mechanical and Aerospace Engineering and Engineering Mechanics, University of Missouri-Rolla, Rolla, MO 65409-0050 USA.

Publisher Item Identifier S 1063-6536(02)05360-5.

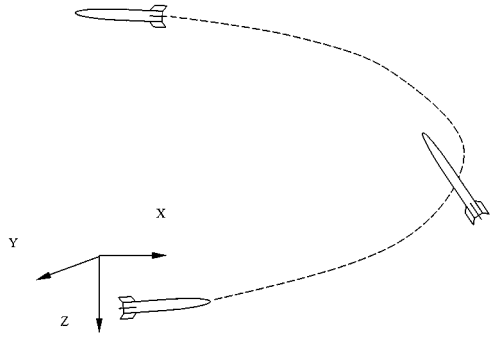

Fig. 1. Agile missile trajectory.

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

Several authors have used neural networks to "optimally" solve nonlinear systems [[2]–[4]]. Almost all these studies fall within four categories: 1) supervised control; 2) direct inverse control; 3) neural adaptive control; and 4) backpropagation through time [7]. These ideas have been used in aerospace applications also. For example, Kim and Calise [4] have proposed a neural-network-based control correction based on Lyapunov theory. A major difference between their approach and this study is that the development of guidance law/control is based on optimal control; hence, it is stabilizing and at the same time minimizing a cost. A fifth and rarely studied class of controller has the most interesting structure. It is called an adaptive critic architecture (Fig. 2) [3], [6]–[9]. Note that in our work, the adaptive critics are used more for effective computational

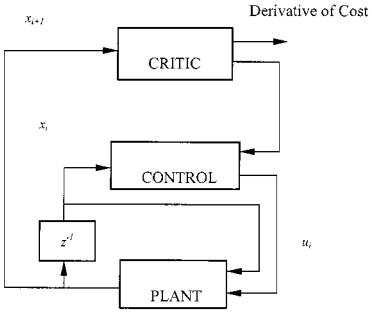

Fig. 2. Schematic of adaptive critic.

uses than for their relationship to reinforcement learning. This approach has a supervisor (critic) that critiques the output of the controller network. The supervisor network, in our case, outputs the Lagrange's multipliers (costates). Each network uses the output of the other network, the propagation equations and the optimality equations in training. When the outputs are mutually consistent, the controller output is optimal. The networks are trained off-line and yet, the resulting control is used as a feedback control. Balakrishnan and Biega [4] have shown the usefulness of this architecture for infinite finite-time linear problems. In this study, we present a feedforward neural framework for the study of linear as well as nonlinear, *finite-time* optimal control problems.

This paper is organized as follows. Adaptive critic development in the context of a fairly general finite time optimal control problem presented in Section II. Hamiltonian corresponding to the state constraints, features of constrained problems, and optimal solutions on the constraint boundary are also discussed. Equations of motion for the missile are given in Section III. Neural-network solutions to this specific minimum time problem are discussed in Section IV. Solutions on the constraint boundary are also discussed. In Section V, it is shown how to use the neurocontroller as a feedback controller. Simulation results and conclusions are presented in Sections VI and VII, respectively.

II. Problem Formulation and Solution Development

A. Cost Function

Through the neural-network methodology presented in this study, we will be able to solve a fairly general class of optimal control problems. The cost function is represented by J, where

$$J = \varphi[x(N)] + \sum_{i=0}^{N-1} L_i[x(i), u(i)]. \tag{1}$$

In (1), $L_i()$ can be a linear or nonlinear function of the states and/or control and $\varphi()$ can be a linear or nonlinear function of terminal states. $L_i()$ is also known as a utility function; i indicates the stage. The underlying system model is given by

$$x(i+1) = f_i(x(i), u(i)) \tag{2}$$

where $f_i()$ can be either linear or nonlinear. The optimal control problem can be formulated in terms of Hamiltonian [1] where the Hamiltonian, H_i, is given by

$$H_i = L_i(x(i), u(i)) + \lambda^T(i+1) f_i(x(i)), u(i). \tag{3}$$

The propagation equations for the Lagrange multiplier, $i = 0, 1, \ldots N-1$, are given by

$$\lambda(i) = \left(\frac{\partial f_1}{\partial x\,(i)}\right)^T \lambda(i+1) + \left(\frac{\partial L_i}{\partial x(i)}\right)^T \tag{4}$$

with boundary condition on λ as

$$\lambda(N) = \left(\frac{\partial \varphi(x(N))}{\partial x(N)}\right)^T. \tag{5}$$

Necessary condition for optimality is

$$\frac{\partial H_i}{\partial u(i)} = 0,\ i = 0, 1, \ldots, N-1. \tag{6}$$

B. Adaptive Critic Solutions

The neural-network solution consists of obtaining a critic network (to output costates) and a controller network to output the control at every stage. This process evolves in two stages.

Synthesis of the Last Network:

1) Note that $\lambda(N) = (\partial\varphi(x(N))/\partial x(N))^T$. For various random values of $x(N)$, λ_N can be calculated.
2) Use the state-propagation (2) and optimality condition in (6) to calculate u_{N-1} for various x_{N-1} by randomly selecting $x(N)$ and the corresponding λ_N from step 1.
3) With u_{N-1} and λ_N, calculate λ_{N-1} for various x_{N-1} by using the costate propagation (4).
4) Train two neural networks: For different values of x_{N-1}, the u_{N-1} network outputs u_{N-1} and the λ_{N-1} network outputs λ_{N-1}. We have optimal control and costates for various values of the state at stage $(N-1)$ now.

It should be noted that we have eliminated the iterative nature of guessing and updating the solutions to the costates in this method through starting at the terminal stage.

Other Networks:

1) Assume different values of the states at stage $(N-2, x_{N-2})$ and use a random network (or initialized with u_{N-1} network) called u_{N-2} network to output u_{N-2}. Use u_{N-2} and x_{N-2} in the state propagation equation to get x_{N-1}. Input x_{N-1} to the λ_{N-1} network to obtain λ_{N-1}. Use x_{N-2} and λ_{N-1} in the optimality condition in (6) to get target u_{N-2}. Use this to correct the u_{N-2} network. Continue this process until the network weights show little changes. This u_{N-2} network yields optimal u_{N-2}.
2) Using random x_{N-2}, output the control u_{N-2} from the u_{N-2} network. Use these x_{N-2} and u_{N-2} to get x_{N-1} and input x_{N-1} to generate λ_{N-1}. Use x_{N-2}, u_{N-2} and λ_{N-1} to obtain optimal λ_{N-2}. Train a λ_{N-2} network with x_{N-2} as input and obtain optimal λ_{N-2} as output.
3) Repeat the last two steps with $k = N-1, N-2, \ldots 0$ until we get u_o.

C. Optimal Solution When State Variable Inequality Constraint is Active

The above method will yield optimal solutions provided there is *no control or state variable constraints*. When there are state variable inequality constraints, these solutions will form optimal solutions to the unconstrained arcs. This is because of the additional conditions imposed by the constraints as the trajectories *enter* and *leave* the constraint boundaries. Optimal solutions on the constrained parts will be obtained next.

State Variable Inequality Constraints: The constrained problem with the state variable inequality constraints is: find the control to minimize the cost function

$$J = \varphi[x(N)] + \sum_{i=0}^{N-1} L_i[x(i),\ u(i)]$$

and the system model

$$x(i+1) = f_i(x(i), u(i))$$

subject to the state constraints

$$S[x(i)] \leq 0 \tag{7}$$

In (7), $S[.]$ is a nonlinear function of states.

If the qth time derivative to the constraint equation is needed in order to obtain an expression that contains the control u explicitly then (7) is called a qth-order state variable inequality constraint. Now the Hamiltonian is

$$H_i = L_i + \lambda^T(i+1) f_i + \mu(i+1)^T S^{(q)}[x(i)] \tag{8}$$

where

$S^{(q)}[x(i)] = 0$ on the constraint boundary $S = 0$;
$\mu(i) = 0$ off the constraint boundary $S < 0$.

$\mu(i)$ is actually an influence function. A necessary condition for $\mu(i)$ is that it should be decreasing on $S[x(i)] = 0$.

Another necessary condition for state inequality constraint problems is that the following tangency constraints must be met both at the entry to and the exit from the constraint boundary:

$$P[x(i)] = \begin{bmatrix} S[x(i)] \\ S^1[x(i)] \\ \cdot \\ \cdot \\ S^{(q-1)}[x(i)] \end{bmatrix} = 0. \tag{9}$$

Equation (9) forms a set of interior boundary conditions. Consequently, the costates λ_i are in general discontinuous at the junctions of constrained and unconstrained arcs. The following relationship must be hold at the entry or the exit points:

$$\lambda^T\left(i^-\right) = \lambda^T\left(i^+\right) + \pi^T \frac{\partial P}{\partial x(i)} \tag{10}$$

i^- signifies just before i and i^+ signifies just after i. Bryson, Denham and Dreyfus [9] have shown that π, the associated Lagrange's multiplier is not unique at one of the junction points. If we pick π arbitrarily at the entry point, then π at the exit point will be automatically determined.

McIntyre and Paiewonsky [10] have given a procedure to calculate π. Assume a constrained problem where the following conditions hold:

1) N is specified (t_f is specified);
2) dS/dt contains the control u;
3) optimal solution contains only one boundary segment;
4) no portion of the boundary segment is an extremal.

Then, at the entry point where the states reach the constraint boundary, the multipliers satisfy

$$\lambda\left(i^-\right) = \lambda\left(i^+\right) + \pi \frac{\partial S}{\partial x(i)} \tag{11}$$

and the integration is continued along the boundary until $\mu(i)$ is zero. Since

$$\mu = \pi > 0 \quad \text{(at the entry point)}$$

and

$$\mu \leq 0 \tag{12}$$

with equality holding only at a finite number of points due to condition 4) above, the quantity μ will approach zero [11], [12]. When the trajectory leaves the boundary, the return to the interior is made with

$$\lambda\left(i^-\right) = \lambda\left(i^+\right) \tag{13}$$

across the corner. Note that with neural networks, we proceed backward. (This will become more clear in the application section.) From this point onwards, we follow the steps outlined in the last section.

Note that $\lambda(i^-)$, $\lambda(i^+)$, $\mu(i)$ and π need to be calculated. We use (4), (6) where H_i defined by (8) to get $\mu(i)$ and $\lambda(i^+)$ since the control is calculated from state propagation equation. $\lambda(i^-)$ and π are calculated from (11) and (12).

III. Minimum Time Problem to Reverse the Heading in a Vertical Plane

The equations of motion in a vertical plane are presented and the minimum time problem is formulated this section. The main goal of this study is to find the control (angle-of-attack) history to minimize the time taken by the missile in reversing its flightpath angle while flying above a minimum mach number. In many engagements, most of the flight is dominated by two-dimensional motion-either in a horizontal or vertical plane. It should be noted that extension of this method to a three-dimensional engagements is straightforward.

A. Equations of Motion of a Missile in a Vertical Plane

The nondimensional equations of motion of a missile (represented as a point mass) in a vertical plane are

$$M' = -S_w M^2 C_D - \sin\gamma + T_w \cos\alpha \tag{14}$$

$$\gamma' = \frac{1}{M}\left[S_w M^2 C_L + T_w \sin\alpha - \cos\gamma\right] \tag{15}$$

where prime denotes differentiation with respect to the nondimensional time, τ. The nondimensional parameters used in

(14)–(15) are: $\tau = g/at; T_w = T/mg; S_w = \rho a^2 S/2mg$ and $M = V/a$.

In these equations, M is the flight Mach number, γ, the flight-path angle, α, the aerodynamic angle of attack, T, the solid rocket thrust, m, the mass of the missile, S, the reference aerodynamic area, V, the speed of the missile, C_L, the lift coefficient, C_D, the drag coefficient, g, the acceleration due to gravity, a, the speed of sound, ρ, the atmospheric density and t is the flight time. Note that C_D and C_L are functions of angle of attack and flight Mach number and a neural network is trained to out put C_D and C_L with angle of attack and flight Mach number as inputs.

B. Minimum Time Optimal Control Problem

The objective of the minimization process is to find the control (angle-of-attack) history to minimize the time taken by the missile to reverse its flightpath angle completely while the Mach number changes from an envelope of initial Mach numbers to a given final Mach number of 0.8.

Mathematically, this problem is stated as to find the control minimizing J, the cost function where

$$J = \int_0^{t_f} dt \tag{16}$$

with the constraints $\gamma(0) = 0°$, $M(0) \equiv$ given, $\gamma(t_f) = 180°$ and $M(t_f) = 0.8$. This constrained optimization problem comes under the class of "free final time" problems in calculus of variations and is difficult to solve. No general solution exists which generates optimal paths for flexible initial conditions.

We seek to provide such solutions using adaptive critic-based neural networks. In order to facilitate the solution using neural networks, the equations of motion are reformulated using the flightpath angle as the independent variable. This process enables us to have a fixed final condition as opposed to the "free final time." It should be observed that when we transform the problem, the independent variable should be monotonically increasing. The transformed dynamic equations are

$$\frac{dM}{d\gamma} = \frac{\left(-S_w M^2 C_D - \sin\gamma + T_w \cos\alpha\right) M}{S_w M^2 C_L - \cos\gamma + T_w \sin\alpha} \tag{17}$$

$$\frac{dt}{d\gamma} = \frac{aM}{g\left(S_w M^2 C_L - \cos\gamma + T_w \sin\alpha\right)} \tag{18}$$

and the transformed cost function is

$$J = \int \left(\frac{aM}{g\left(S_w M^2 C_L \cos\gamma + T_w \sin\alpha\right)}\right) d\gamma \tag{19}$$

with the limits on γ being zero and π rad.

In this study, the final velocity is treated as a hard constraint. This means that the flight path angle and the velocity constraints are met *exactly* at the final point. We express the dynamics and associated optimal control equations in discrete form in order to use them with discrete feedforward neural networks. The system equations in discrete form

$$\begin{aligned} M_{k+1} &= M_k + \frac{\left(-S_w M_k^2 C_{Dk} - \sin\gamma_k + T_{wk}\cos\alpha_k\right) M_k}{S_w M_k^2 C_{Lk} - \cos\gamma_k + T_{wk}\sin\alpha_k} \cdot \delta\gamma_k \\ t_{k+1} &= t_k + \frac{a_{M_k}\delta\gamma_k}{g\left(S_w M_k^2 C_{Lk} - \cos\gamma_k + T_{wk}\sin\alpha_k\right)}. \end{aligned} \tag{20}$$

Note that discretizing γ implies that it is constant during the interval considered. The corresponding Hamiltonian equation is

$$\begin{aligned} H_k &= \frac{a_{M_k}\cdot\delta\gamma_k}{g\left(S_w M_k^2 C_{Lk} - \cos\gamma_k + T_{wk}\sin\alpha_k\right)} \\ &+ \lambda_{k+1}\left(M_k + \frac{\left(-S_w M_k^2 C_{Dk} - \sin\gamma_k + T_{wk}\cos\alpha_k\right) M_k}{S_w M_k^2 C_{Lk} - \cos\gamma_k + T_{wk}\sin\alpha_k} \cdot \delta\gamma_k\right). \end{aligned} \tag{21}$$

For convenience, let us define

$$\text{denk} = S_w M_K C_{Lk} - \cos\gamma_k + T_{wk}\sin\alpha_k.$$

Note that the term denk is in the denominator of (20) and (21)

$$\begin{aligned} \frac{\partial \text{denk}}{\partial \alpha_k} &= S_w M_k^2 \frac{\partial C_{Lk}}{\partial \alpha_k} + T_{wk}\cos\alpha_k \\ \frac{\partial \text{denk}}{\partial M_k} &= 2 S_w M_k C_{Lk} + S_w M_k^2 \frac{\partial C_{Lk}}{\partial M_k}. \end{aligned} \tag{22}$$

Derivatives of the lift and drag coefficients with respect to the angle of attack and the Mach number are obtained from a neural network which stores these coefficients. The costate equation is given in terms of denk as shown in (23) at the bottom of the page. Note that there is no boundary condition on λ since M is given at both ends. The necessary condition for optimality is

$$\frac{\partial H_k}{\partial \alpha_k} = 0. \tag{24}$$

In an expanded form (24) becomes

$$\begin{aligned} &\frac{a}{g}\cdot\frac{\partial \text{denk}}{\partial \alpha_k} + \lambda_{k+1}\cdot\left(S_{wM_k^2}\frac{\partial C_{Dk}}{\partial \alpha_k} + T_{wk}\sin\alpha_k\right)\cdot\text{denk} \\ &+ \lambda_{k+1}\left(-S_w M_k^2 C_{Dk} - \sin\lambda_k + T_{wk}\cos\alpha_k\right) \\ &\cdot\frac{\partial \text{denk}}{\partial \alpha_k} = 0. \end{aligned} \tag{25}$$

$$\begin{aligned} \frac{\partial H_k}{\partial M_k} = \lambda_k &= \frac{a\cdot\delta\gamma_k}{g\cdot\text{denk}} - \frac{aM_k\delta\gamma_k}{g\cdot\text{denk}^2}\cdot\frac{\partial \text{denk}}{\partial M_k} \\ &+ \lambda_{k+1}\cdot\delta\gamma_k\frac{\left(-3S_w M_k^2 C_{Dk} - S_w M_k^3\frac{\partial C_{Dk}}{\partial M_k} - \sin\gamma_k + T_{wk}\cos\alpha_k\right)}{\text{denk}} \\ &+ \lambda_{k+1} + \lambda_{k+1}\cdot\delta\gamma_k\frac{\left(S_w M_k^2 C_{Dk} + \sin\gamma_k - T_{wk}\cos\alpha_k\right)\cdot M_k}{\text{denk}^2}\cdot\frac{\partial \text{denk}}{\partial M_k}. \end{aligned} \tag{23}$$

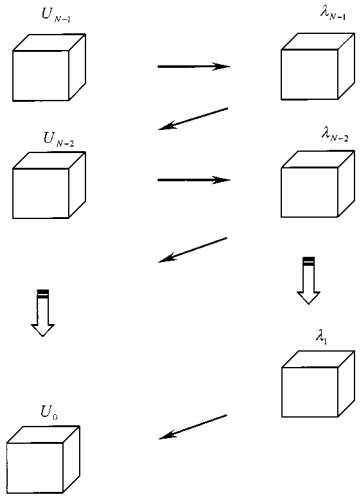

Fig. 3. Schematic of successive adaptive critic controller and critic synthesis.

At every point we solve (25) using the Newton-Raphson method during the network training.

IV. Development of Neural–Network Solutions

The development of the neural networks for this problem proceeds in two stages. We start from the last stage and proceed backward. Note that $\varphi(\cdot)$ is zero in this formulation; however, the final state, M_N is specified.

A. Last Network

1) Final Mach number, M_N, is fixed at 0.8. Set $\Delta\gamma$. For random values of M_{N-1}, calculate α_{N-1} from the state propagation equation.
2) Use optimality condition $H_{\alpha_{N-1}}(M_{N-1}, \lambda_N, \alpha_{N-1}) = 0$ to solve for appropriate λ_N.
3) From the costate propagation equation, calculate λ_{N-1}.
4) Train two neural networks: The α_{N-1} network outputs α_{N-1} for different values of M_{N-1} and the λ_{N-1} network outputs λ_{N-1} for different values of M_{N-1}. We have optimal α_{N-1} and λ_{N-1} now.

B. Other Networks

1) Assume different values of M_{N-2} and use a random neural network (or initialized with α_{N-1} network) called α_{N-2} network to output α_{N-2}. Use M_{N-2} and α_{N-2} to obtain M_{N-1}. Input M_{N-1} to λ_{N-1} network to get λ_{N-1}. Use M_{N-2}, λ_{N-1} in $H_{\alpha_{N-2}} = 0$ to solve for α_{N-2}. Use this α_{N-2} to correct the network. Continue this process until α_{N-2} network converges. This α_{N-2} network yields optimal α_{N-2}.
2) Using random M_{N-2} into network obtains optimal α_{N-2}. Use M_{N-2} and α_{N-2} to obtain M_{N-1} and input to λ_{N-1} network to generate λ_{N-1}. Use M_{N-2}, α_{N-2} and λ_{N-1} in costate equation to obtain optimal λ_{N-2}. Train λ_{N-2} network with M_{N-2} as input. We have λ_{N-2} network that yields optimal λ_{N-2}.
3) Repeat steps 5 and 6 with $k = N-1, N-2, \ldots, 0$, until we get α_0. A schematic of the network development is presented in Fig. 3.

As noted in Section II, the optimal solutions along the constrained path has to be considered separately.

C. Optimal Solution on the Constraint Boundary

Now we apply the procedure discussed in Section II to the missile problem. This problem can be stated as follows: given the dynamics

$$\frac{dM}{d\gamma} = \frac{\left(-S_w M^2 C_D - \sin\gamma + T_w\alpha\right) M}{S_w M^2 C_L - \cos\gamma + T_w \sin\alpha} \tag{26}$$

find a controller to minimize the cost function

$$J = \int_0^{\pi} \frac{aM}{g\left(S_w M^2 C_L \cos\gamma + T_w \sin\alpha\right)} d\gamma \tag{27}$$

subject to the state variable inequality constraint

$$M \geq M^* \tag{28}$$

(here $M^* = 0.18$). This is expressed in terms of S as

$$S = M^* - M \leq 0. \tag{29}$$

The first derivative of S is

$$S^{(1)} = -\frac{dM}{d\gamma} = \frac{-\left(-S_w M^2 C_D - \sin\gamma + T_w \cos\alpha\right) M}{S_w M^2 C_L - \cos\gamma + T_w \sin\alpha}. \tag{30}$$

Note that (30) contains the control α. Consequently, our problem is a state variable inequality constraint problem of order one.

The corresponding Hamiltonian equation is: (in continuous form)

$$H = L + \lambda f + \mu S^{(1)} \tag{31}$$

where

$$f = \frac{dM}{d\gamma} s^{(1)} = -\frac{dM}{d\gamma}. \tag{32}$$

Substituting the expression for f from (32) in (31), we get

$$H = L + (\lambda - \mu) f. \tag{33}$$

To work with neural networks, we write (33) in discrete form as shown in (34) at the bottom of the next page.

Speyer *et al.* [13] have shown that the solution to state variable inequality constraint problem of order one contains only one boundary segment. Furthermore, the final value of the independent variable, γ_f is specified (equal to 180°); it is apparent that the boundary arc is not the optimal solution to the unconstrained problem; so our problem satisfies all four conditions of Section II [11]. Consequently, we can adopt the procedure in Section II [11] for computing the quantities on the boundary.

From [1] and [13] we know that separate computation of unconstrained arc is possible in problems with state variable inequality constraints if they contain only one constrained arc in the middle and the contribution of the constrained arc to the performance index depends only on the entry and exit values of one variable (t or one component of x).

D. Development of Neural —Network Solutions on the Constraint Boundary

The procedure to develop neural-network solution on the unconstrained path is similar to [13]. There are a few points that need to be noted, however.

1) We have developed a neural-network solution corresponding to the unconstrained state variable problem. With state inequality constraints, parts of the solution can still be used. The optimal solution from the boundary segment to the final state is on the optimal trajectories of the unconstrained problem.
2) For the state variable inequality constraint problem, the exit point is the tangency point where one unconstrained state trajectory is tangent to the state constraint line. At this point, $\mu = 0$.
3) At the exit point (the step $k+1$) λ is continuous, but at the entry point λ experiences a jump; the key is to find π; that is, to find μ at the entry point. We solve for μ in the following way. From the state equation we can compute α; since α does not have any discontinuity and the Mach number is constant (minimum) on the boundary, the costate equation can be written as

$$\begin{aligned}\lambda_k =& \frac{\alpha\delta\gamma_k}{g\cdot \text{denk}} - \frac{aM_k\delta\gamma_k}{g\cdot\text{denk}}\cdot\frac{\partial \text{denk}}{\partial M_k} + (\lambda_{k+1}-\mu_{k+1})\,\delta\gamma_k \\ &\cdot\left[\frac{-3S_wM_k^2C_{Lk} - S_wM_k^3\frac{\partial C_{Dk}}{\partial M_k} - \sin\gamma_k + T_{wk}\cos\alpha_k}{\text{denk}}\right] \\ &+ (\lambda_{k+1}-\mu_{k+1}) + (\lambda_{k+1}-\mu_{k+1})\,\delta\gamma\cdot_k \\ &\cdot\frac{\left(S_wM_k^2\cdot C_{Dk} + \sin\gamma_k\cdot T_{wk}\cos\alpha_k\right)\cdot M_k}{\text{denk}^2}\cdot\frac{\partial\text{denk}}{\partial M_k}.\end{aligned} \tag{35}$$

Necessary condition for optimality condition on the constraint boundary is

$$\frac{\partial H_k}{\partial\alpha_k} = 0. \tag{36}$$

This leads to

$$\begin{aligned}&\frac{a}{g}\frac{\partial\text{denk}}{\partial\alpha_k} + (\lambda_{k+1}-\mu_{k+1})\left(S_wM_k^2\frac{\partial C_{Dk}}{\partial\alpha_k} + T_{wk}\sin\alpha_k\right)\text{denk} \\ &+(\lambda_{k+1}-\mu_{k+1})\cdot\left(-S_wM_k^2\cdot C_{Dk} - \sin\alpha_k + T_{wk}\cos\alpha_k\right) \\ &\cdot\frac{\partial\text{denk}}{\partial\alpha_k} = 0.\end{aligned} \tag{37}$$

From (35) and (37), we can compute λ_k and μ_k on the boundary.

4) At the entry to the constraint boundary, we calculate the discontinuous λ, μ and π as described in Section II.

V. Use of Networks in Real-Time as Feedback Conditions

Assume any Mach number M_0 [within the trained range] at the initial point. Use the α_0 neural network to find optimal α and integrate in time until a flightpath angle γ_1 for α_1 network is reached; use the Mach number M_1 at that point to find α_1 from the α_1 neural network and integrate until the flightpath angle γ_2 corresponding to the next controller is reached and so on, until the boundary for the minimum Mach number is reached. On the boundary, we use the state propagation equation to calculate the control. It should be checked at each stage whether the state is on the boundary or not. As soon as the control takes it off the boundary, the neurocontroller is used until the final flightpath angle γ_f is reached. It should be noted that the neurocontrollers are obtained for the entire range (0 to 180°) of flightpath angles since the entry point to the constraint boundary is different for each initial velocity.

Note that the forward *integration* is done *in terms of time*. As a result, even though the network synthesis is done *off-line*, the control is *a feedback process* based on current states.

VI. Numerical Results

In this section, we present the numerical results from representative simulations. Tables of aerodynamic data of C_L and C_D variations with Mach numbers and angle of attack were provided by the Air Force. Selection of the neural-network type and the architecture are primarily done with intuition only. All the neural networks in this study are feedforward networks. We picked a feedforward network in order to facilitate numerical derivatives of the outputs with respect to the inputs. Each network has a three-layered structure with the first layer having a tangent sigmoidal activation function, the second layer having a logarithmic sigmoidal activation function and the third layer having a unit gain. Each layer consists of nine neurons. Our choices were proved to be adequate from the results. We *did not* try to optimize the structure of networks in this study. A Levenberg–Marquardt method is used to train the networks. Any other training method would have been fine too.

The state inequality constraint is chosen such that the flight Mach number, $M \geq 0.18$. We used 21 controller networks corresponding to the unconstrained segment from the final point to the boundary and another 30 networks to represent the solutions from initial state to the entry point and entry to the exit point. The number of networks was dictated by the convergence

$$\begin{aligned}H_k =& \left(\frac{dt}{d\gamma}\right)_k\delta\gamma_k + (\lambda_{k+1}-\mu_{k+1})\cdot M_{k+1} \\ =& \frac{aM_k\cdot\delta\gamma_k}{g\left(S_wM_k^2C_{Lk} - \cos\gamma_k + T_{wk}\sin\alpha_k\right)} \\ &+(\lambda_{k+1}-\mu_{k+1})\left[M_k + \frac{\left(-S_wM_kC_{Dk} - \sin\gamma_k + T_w\cos\alpha_k\right)}{S_wM_k^2C_{Lk} - \cos\gamma_k + T_{wk}\sin\alpha_k}\cdot M_k\cdot\delta\gamma_k\right].\end{aligned} \tag{34}$$

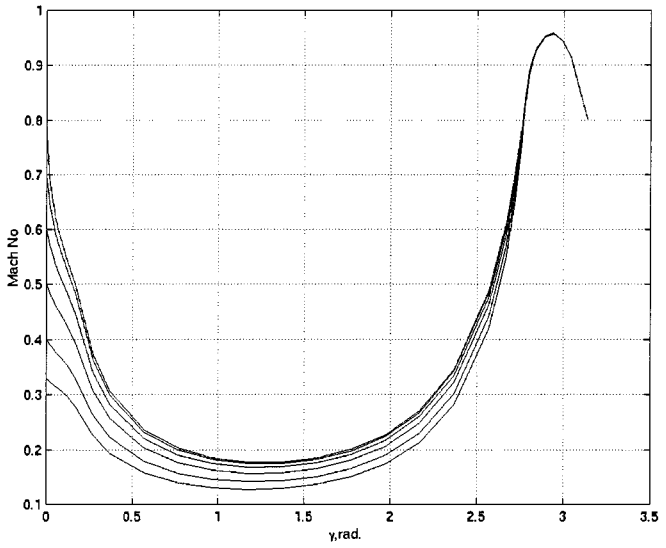

Fig. 4. Mach number versus flightpath angle, γ (Unconstrained, M_0 varies from 0.33 to 0.8 and $M_f = 0.8$).

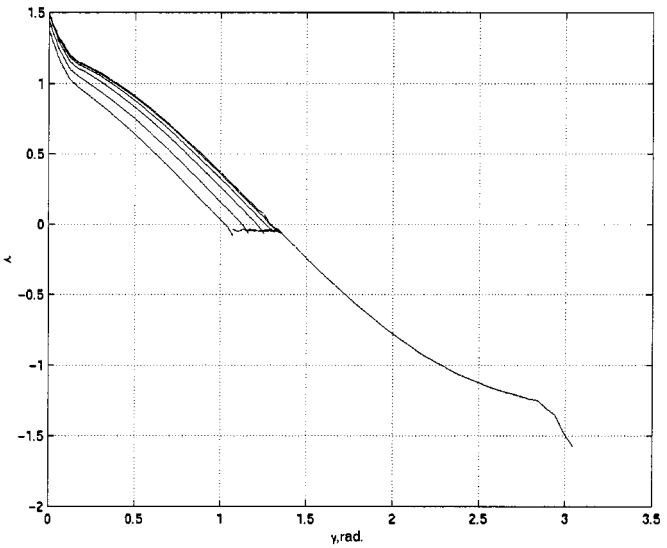

Fig. 6. Costate, λ versus flightpath angle, γ ($M_0 = 0.33 \sim 0.8$, $M_f = 0.8$, $M \geq 0.18$).

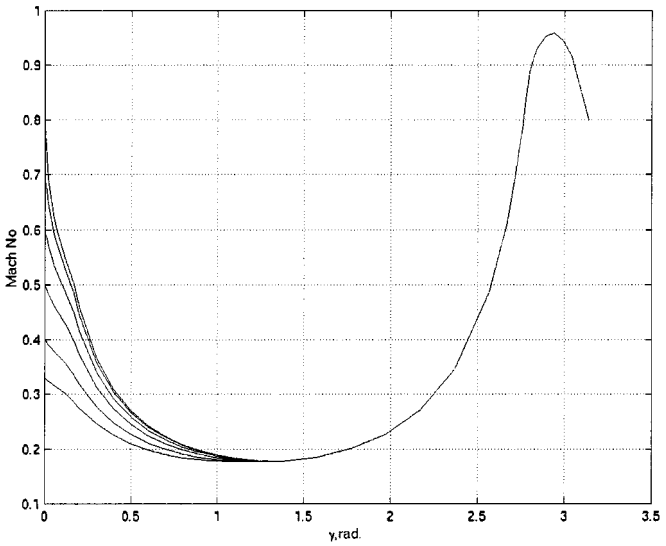

Fig. 5. Mach number versus flightpath angle, γ ($M_0 = 0.33 \sim 0.8$, $M_f = 0.8$, $M \geq 0.18$).

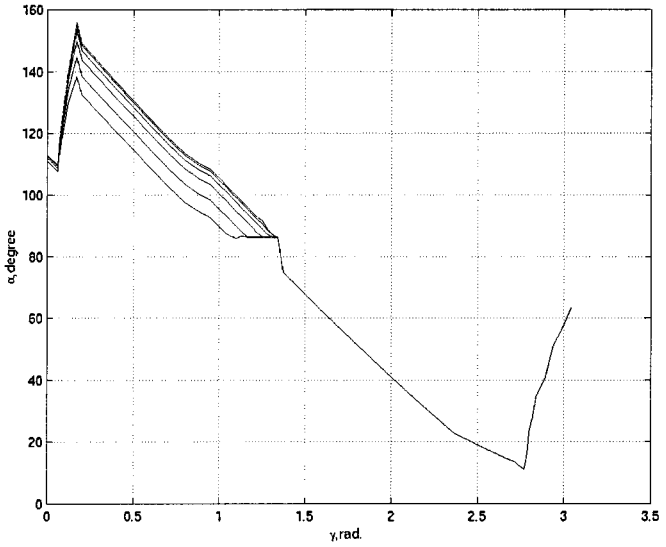

Fig. 7. Angle of attack, α versus flightpath angle, γ ($M_0 = 0.33 \sim 0.8$, $M_f = 0.8$, $M \geq 0.18$).

characteristics of the networks; If the intervals were big ($\delta\gamma_k$ was big), then the networks did not converge for the range of Mach number used as inputs. A higher number of networks for a problem does not translate to more calculations for control at a point. It simply means that we have to store more weights; even if this problem is formulated in three dimensions, this will lead only to a few thousand weights. One way to reduce the number of networks or the number of weights is to treat the flight path angle as an input to the networks and reduce the number of networks from 51 to one.

In [13], the neural-network solutions for the unconstrained problem are compared with a shooting method solution to show that the neural solutions are indeed *optimal*. Mach number histories with flightpath angle for the unconstrained problem are provided in Fig. 4 [13]; For the state constrained problem, these histories are provided in Fig. 5. The costate histories and the angle of attack histories are presented in Figs. 6 and 7 as functions of γ. The nature of the solutions is seen better in Figs. 8–10, which show Mach number, angle of attack and the costate histories versus time, respectively. Note that the minimum Mach number constraint is satisfied *exactly*. Furthermore, the solutions *after* the state constraint is reached are the same for all cases (Fig. 5). This can be explained as follows: Observe that corresponding to the constraint $M \geq M^*$, the trajectory should include two interior segments and one boundary segment. According to [1] and [13], we will have n different trajectories corresponding to n different initial Mach numbers in the interior segment from initial state until entering the boundary arc; they will enter the constraint at n different flightpath angles or γ values. However, there will be only one

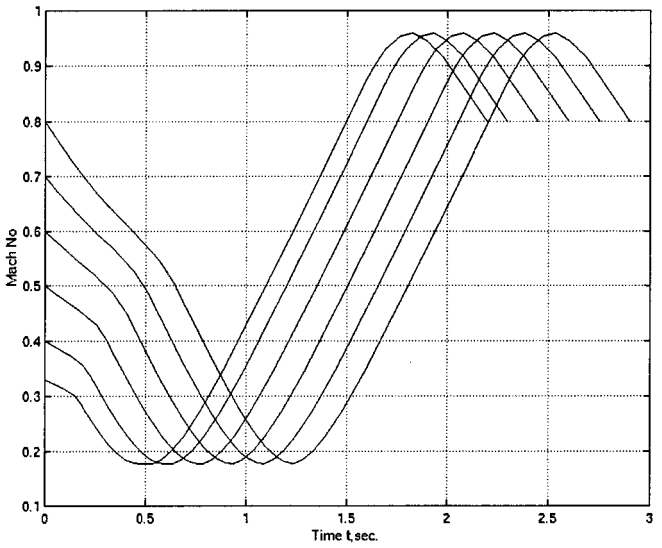

Fig. 8. Mach number versus time, t ($M_0 = 0.33 \sim 0.8$, $M_f = 0.8$, $M \geq 0.18$).

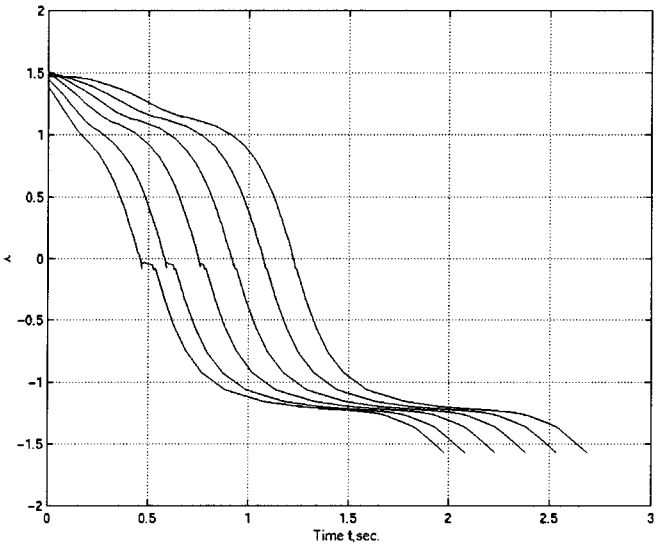

Fig. 9. Costate, λ versus time, t ($M_0 = 0.33 \sim 0.8, M_f = 0.8, M \geq 0.18$).

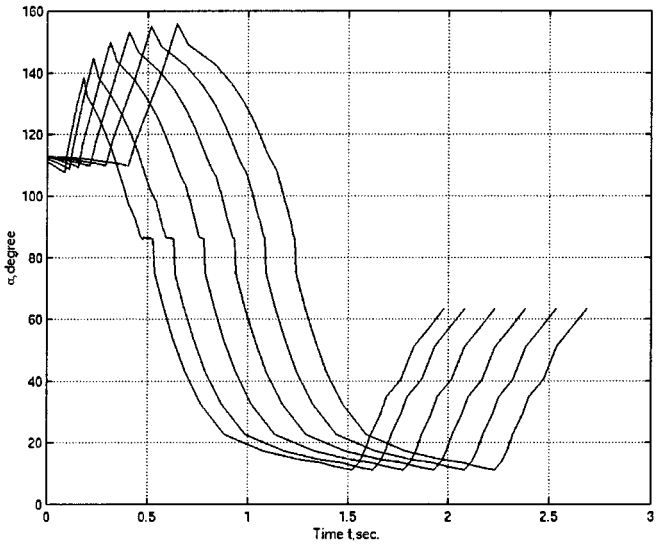

Fig. 10. Angle of attack versus time, t ($M_0 = 0.33 \sim 0.8$, $M_f = 0.8$, $M \geq 0.18$).

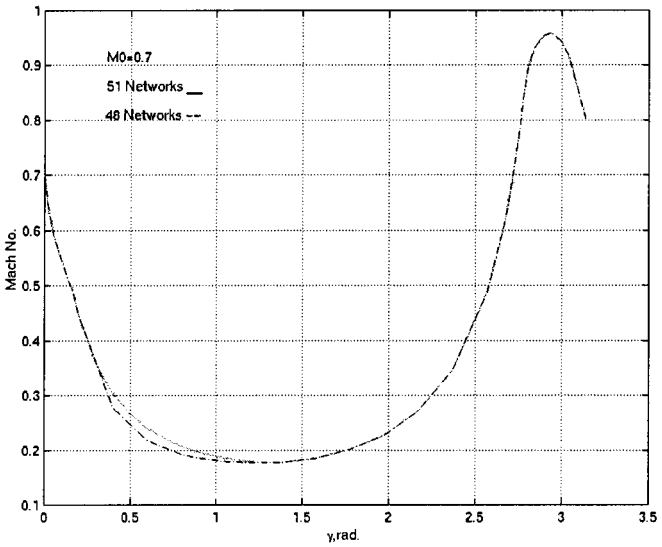

Fig. 11. Mach number versus flightpath angle, γ.

optimal trajectory from boundary segment to the final state since there is only one initial state (minimum Mach number) afterwards and only one final state (final Mach number). It should be noted that each of these trajectories is different in its time history (Figs. 8–10).

Note that the minimum flight time is different with different initial Mach numbers. The flight time varies inversely with the initial Mach number. This is to be expected because for lower Mach number, less energy is needed to reverse the heading.

The advantage of using the adaptive critic approach is clear from these solutions. For each starting Mach number a minimum time trajectory has been obtained with a bound on minimum Mach number. However, the same cascade of neurocontrollers is used to solve and generate optimal control for *this envelope of initial conditions*. We carried out further numerical experiments to test the robustness of controllers; we removed a few controllers during the phase of flight (which means that some of the controls are held for longer periods) before reaching the state constraint and carried out the simulations. Mach number histories and the angle of attack histories for a 51-network controller and a 48-network controller are presented in Figs. 11 and 12. Even though the trajectory is more suboptimal, the constraints and the end conditions are met. In order to study the effects of parameter variations, we changed the nondimensional thrust level from 27 to 37 during the period when the flight path angle changed from 0.55–1.17 radians which is a relatively long interval; the results from this experiment shown in Fig. 13 demonstrate that the network controllers can still steer the missile to a desired final state accurately.

VII. Conclusion

An adaptive critic-based neural network solution for a "bounded state space, free final time" problem was proposed. Results from an agile missile application which is nonlinear in dynamics and control show the potential of the adaptive

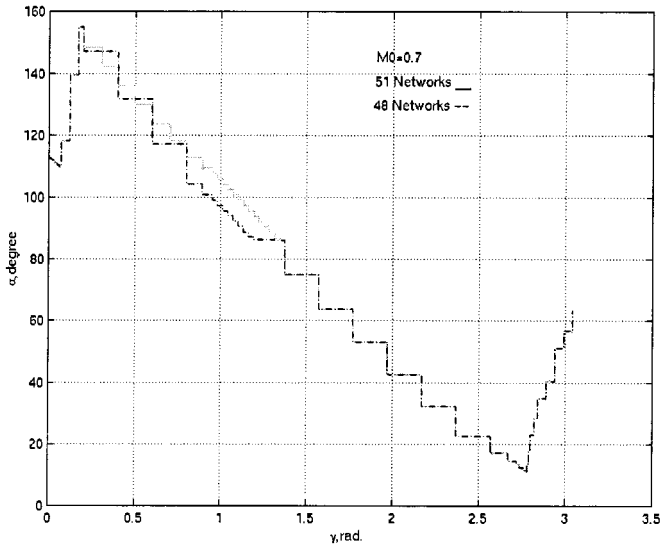

Fig. 12. Angle of attack, α versus flightpath angle, γ for robustness test ($M_0 = 0.7$, $M_f = 0.8$, $M \geq 0.18$).

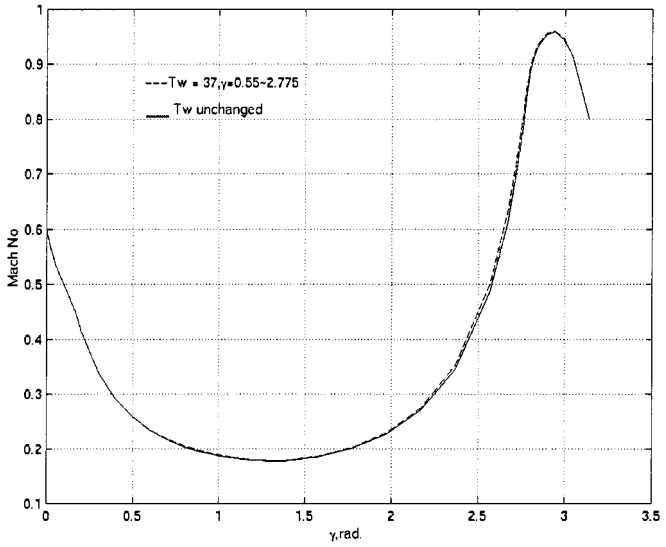

Fig. 13. Mach number versus flightpath angle, α for robustness test with thrust variation ($M_0 = 0.6$, $M_f = 0.8$, $M \geq 0.18$).

critic approach to solve complex optimal guidance/control problems. An added advantage in using these neurocontrollers is that they provide minimum time solutions even when we change the initial flight path angle from zero to any non zero (positive) value. To our knowledge, there is no one tool (other than dynamic programming) which provides such solutions for flexible initial conditions.

REFERENCES

[1] A. E. Bryson and Y. Ho, *Applied Optimal Control*. Bristol, PA: Hemisphere, 1975, pp. 128–211.
[2] K. J. Hunt, "Neural networks for controller systems, a survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.
[3] D. A. White and D. Sofge, *Handbook of Intelligent Control*. New York: Van Nostrand Reinhold, 1992, ch. 3, 5, 8, 12, 13.
[4] S. N. Balakrishnan and V. Biega, "Adaptive critic based neural networks for aircraft optimal control," *AIAA J.*, vol. 19, no. 4, pp. 893–898, Aug. 1996.
[5] B. S. Kim and A. J. Calise, "Nonlinear flight control using neural networks," *AIAA J. Guidance, Contr., Dyn.*, vol. 20, no. 1, pp. 26–33, 1997.
[6] C. W. Anderson, "Learning to control an inverted pendulum using neural networks," *IEEE Contr. Syst. Mag.*, vol. 9, pp. 31–37, Apr. 1898.
[7] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 834–846, 1983.
[8] P. Werbos, "Neurocontrol and supervised Learning; an overview and evaluation," in *Handbook of Intelligent Control*. New York: Van Nostrand Reinhold, 1992.
[9] A. E. Bryson, W. F. Denham, and S. E. Dreyfus, "Optimal programming problems with inequality constraints I: Necessary condition for external solution," *AIAA J.*, no. 11, pp. 2544–2550, 1963.
[10] J. McIntyre and B. Paiewonsky, "On optimal control with bounded state variables," in *Advances in Control Systems*, C. T. Leondes, Ed. New York: Academic, 1967, vol. 5.
[11] E. Kreindler, "Additional necessary conditions for optimal control with state-variable inequality constraints," *J. Optimization Theory Applicat.*, vol. 38, no. 2, October 1982.
[12] D. Han and S. N. Balakrishnan, "Adaptive critic based neural networks for agile missile control," in *Proc. AIAA Guidance, Navigation, Contr. Conf. Exhibit*, Boston, MA, Aug. 10–12, 1998, pp. 1803–1812.
[13] J. L. Speyer, R. K. Mehra, and A. E. Bryson, "The separate computation of arcs for optimal flight paths with state variable inequality constraints," in *Advanced Problems and Methods for Space Flight Optimization*. New York: Pergamon, 1969, pp. 53–68.

Dongchen Han was born on October 20, 1964, in Hebei, China. He finished his primary and secondary education in Beijing, China. He received the Bachelor of Science degree in aerospace engineering from Beijing University of Aeronautics and Astronautics in July 1985 and the Master of Science degree in aerospace engineering from Beijing Electrical System Institute of China National Space Administration in March 1988. He received the Ph.D. degree in aerospace engineering at the University of Missouri-Rolla, Rolla, in August 2001.

From March 1988 to July 1996, he worked as Assistant Engineer, Engineer, and Senior Engineer in Beijing Electrical System Institute.

S. N. Balakrishnan received the Ph.D. degree in aerospace engineering the from the University of Texas at Austin, Austin.

Since 1985, he has been with University of Missouri-Rolla, Rolla. His nonteaching experience includes work as a Lead Engineer in the Space Shuttle program, Fellow, Center for Space Research at the University of Texas at Austin, Summer Faculty Fellow at the Air Force Research Laboratory, Eglin, FL, and Engineer, Indian Space Program. His interests include System Theory and Applications. His current research uses neural networks and classical methods in the identification and robust control of missiles, airplanes, rockets, and mechanical and other "interesting" systems. His work has been sponsored by the National Science Foundation, the Air Force, the Naval Surface Warfare Center, the Army Space and Missile Defense Command and NASA.

Dr. Balakrishnan is a Member of Sigma Gamma Tau, is an Associate Fellow of AIAA and serves as a Director, American Automatic Control Council. He has also been a Program Chair, AIAA Guidance, Navigation and Control Conference and a Vice Chair, American Control Conference.