JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 27, No. 5, September–October 2004

Online Adaptive Critic Flight Control

Silvia Ferrari
Duke University, Durham, North Carolina 27707
and
Robert F. Stengel
Princeton University, Princeton, New Jersey 08544

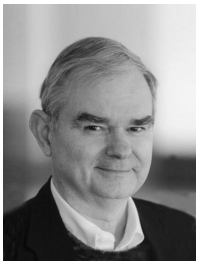
A nonlinear control system comprising a network of networks is taught by the use of a two-phase learning procedure realized through novel training techniques and an adaptive critic design. The neural network controller is trained algebraically, offline, by the observation that its gradients must equal corresponding linear gain matrices at chosen operating points. Online learning by a dual heuristic adaptive critic architecture optimizes performance incrementally over time by accounting for plant dynamics and nonlinear effects that are revealed during large, coupled motions. The method is implemented to control the six-degree-of-freedom simulation of a business jet aircraft over its full operating envelope. The result is a controller that improves its performance while unexpected conditions, such as unmodeled dynamics, parameter variations, and control failures, are experienced for the first time.

I. Introduction

THE problem of optimizing a desired metric over time lies at the basis of many robust and fault-tolerant control and identification schemes. Dynamic programming (DP) uses the principle of optimality to find an optimal strategy of action in a nonlinear environment.[1] Classical DP methods discretize the state space and make a direct comparison of the cost associated with all feasible trajectories that satisfy the principle of optimality, guaranteeing the solution of the optimal control problem.[2] However, these approaches lead to a number of computations that grows exponentially with the number of state variables ("curse of dimensionality").[1] Adaptive critic designs constitute a class of approximate dynamic programming (ADP) methods that uses incremental optimization, combined with parametric structures that approximate the optimal cost and control, to reduce the required computations.[3] At any moment in time, they optimize a short-term cost metric that ensures incremental optimization of the cost over all future times. A critic network is used to evaluate the performance of the parametric structure that approximates the optimal control law, also referred to as an action network. In this paper, neural networks are the parametric structures of choice because they easily handle large-dimensional input and output spaces and can learn in batch or incremental mode.

Adaptive critic designs can be used to solve nonlinear optimal control problems, without posing restrictions on the form of the dynamic equation or the controller a priori. By approximating the DP solution forward in time, they can learn the optimal control law both off and online. When plant dynamics and uncertainties are captured by available models or satisfy appropriate assumptions, the appropriate control law and its performance guarantees can be obtained offline. If significant dynamic and environmental effects arise that are not anticipated and accounted for a priori, the control system performance can deteriorate and, possibly, compromise safety. Then, a controller that optimizes its strategy online, subject to these effects, can improve performance and prevent hazardous conditions in real time. Although ADP methods, including adaptive critics, have been shown to converge to the optimal policy over time,[3] in practice, it

Silvia Ferrari is Assistant Professor of Mechanical Engineering and Materials Science at Duke University, where she directs the Laboratory for Intelligent Systems and Controls (LISC). Currently, her principal research interests are robust adaptive control of aircraft, learning and approximate dynamic programming, and management of heterogeneous sensor networks. She received the B.S. degree from Embry-Riddle Aeronautical University and the M.A. and Ph.D. degrees from Princeton University. She is a member of the Institute of Electrical and Electronics Engineers and of the American Institute of Aeronautics and Astronautics. She received the Zonta International Amelia Earhart Fellowship Award (2000 and 2001), the AAS Donald K. "Deke" Slayton Memorial Scholarship (2001), the ASME Graduate Teaching Fellowship (2001), and the AIAA Guidance, Navigation, and Control Graduate Award (1999).



Robert Stengel is Professor and former Associate Dean of Engineering and Applied Science at Princeton University, where he directs the program in robotics and intelligent systems. Prior to his Princeton appointment, he was with The Analytic Sciences Corporation, Charles Stark Draper Laboratory, USAF, and NASA. He was a principal designer of the Apollo lunar module manual control logic, and he contributed to space shuttle control system design. He received the S.B. degree from Massachusetts Institute of Technology and M.S.E., M.A., and Ph.D. degrees from Princeton University. He is a Fellow of the Institute of Electrical and Electronics Engineers and of the American Institute of Aeronautics and Astronautics. He received the AACC John R. Ragazzini Education Award (2002), AIAA Mechanics and Control of Flight Award (2000), and the FAA's first annual Excellence in Aviation Award (1997). He wrote *Optimal Control and Estimation* (Dover, 1994), *Flight Dynamics* (in press), and numerous technical papers.

Received 28 October 2002; revision received 12 November 2003; accepted for publication 30 December 2003.

has proven difficult to obtain convergence quickly enough to affect performance in real time.[4] Also, online learning may deteriorate knowledge assimilated earlier by the control system about other operating regions, affecting global stability.

Adaptive critic controllers have been successfully trained offline for two-axle vehicle steering and speed control,[5] agile missile interception,[6] aircraft autolanding and control,[7–9] and turbogenerator control.[4] Although several architectures have been proposed to accelerate convergence to the optimal solution (as in Refs. 10–12, dual heuristic programming (DHP) has been shown to learn quickly and to alleviate persistence of excitation problems by computing the correlation between the cost and the individual state elements.[4,10,13] In this paper, a DHP architecture is trained online to control the nonlinear simulation of a business jet aircraft over its full operating envelope, improving performance during unexpected conditions such as unmodeled dynamics, parameter variations, and control failures.

The nonlinear control system, comprising an action and a critic neural network, is trained by the use of a two-phase procedure. During the first offline training phase, the networks' size and weights are determined from earlier control knowledge, that is, multivariable control theory, meeting satisfactory safety, and performance baselines a priori. During the second online phase, the networks are updated incrementally to improve control response based on the actual state of the plant, accounting for differences between actual and assumed dynamic models. Extensive numerical experiments show that earlier knowledge is always preserved during online learning and that the neural networks adapt only to improve on a priori performance baselines.

II. Foundations

In this paper, the adaptive critic architecture approximates the solution of an infinite horizon optimal control problem by means of neural networks, subject to the real-time dynamics of a continuous plant or simulation. The adaptive critic controller adapts online, with the plant operating over the entire range of the full state and command input $\{\boldsymbol{x}(\boldsymbol{y}_c), \boldsymbol{y}_c\}$, or some suitably dense set in the space denoted by R; thus, it is said to be global. The state of the plant $\boldsymbol{x}$ and the command input $\boldsymbol{y}_c$ both are fed to the controller online and are unknown before operation. It is assumed that linearized time-invariant plant models are known a priori for a subset of operating points $P \subset R$. Corresponding linear control data are used to train and test the action and critic neural networks offline. In a flight-control problem, such as the one presented in Sec. V, P may consist of the steady-level flight envelope of the aircraft, and R is the envelope of all possible maneuvers. The same networks are then modified incrementally online during rapidly changing, large-angle maneuvers through a DHP architecture.[10]

During the first phase, the action and critic neural networks are trained in a batch mode. The initial controller, which is nonlinear but is similar in concept to a gain-scheduled controller, can be considered to be global over P. During the online phase, its knowledge of the operating space is expanded as new regions are explored in R. The online optimization is local because the networks are updated with every observed value of the state. However, if improved performance locally does not deteriorate it elsewhere in R, the online phase amounts to an expansion of the controller's region of optimality beyond P. The offline phase is based on linear multivariable control, and the online phase is based on approximate dynamic programming.

A. Problem Statement

Consider the deterministic minimization of a scalar integral function of the $n \times 1$ plant state $\boldsymbol{x}$ and of the $m \times 1$ control $\boldsymbol{u}$ and a scalar terminal cost:

$$J = \phi[\boldsymbol{x}(t_f)] + \int_{t_0}^{t_f} L[\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)]\,\mathrm{d}\tau \tag{1}$$

The objective is to determine the control law that causes this cost function to be stationary, subject to the nonlinear dynamics of the plant:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), \boldsymbol{u}(t)] \tag{2}$$

$\boldsymbol{x}(t_0)$ given. Plant motions and controls are sensed in the $e \times 1$ output vector $\boldsymbol{y}$,

$$\boldsymbol{y}(t) = \boldsymbol{h}[\boldsymbol{x}(t), \boldsymbol{u}(t)] \tag{3}$$

Here, it is assumed that perfect measurements are available and that the output views all elements of the state, that is, $\boldsymbol{y} = \boldsymbol{x}$. If these assumptions are not met, the use of an optimal estimator also is required. The mission goals are expressed by the $e_c \times 1$ command input $\boldsymbol{y}_c$, which can be viewed as some desirable combination of state and control elements with $e_c \leq m$.

The action network models the control law, which can be assumed to be a function solely of the state, without loss of generality. It can be written as the sum of a nominal and a perturbed effect,

$$\boldsymbol{u}^*[\boldsymbol{x}^*(t)] = \boldsymbol{u}_0^*\left[\boldsymbol{x}_0^*(t)\right] + \Delta\boldsymbol{u}^*\left[\boldsymbol{x}_0^*(t), \Delta\boldsymbol{x}^*(t)\right] \tag{4}$$

where $\boldsymbol{x}^*(t) = \boldsymbol{x}_0^*(t) + \Delta\boldsymbol{x}^*(t)$, and $(\bullet)^*$ denotes the optimal solution. When the control law depends on parameters and command inputs as well as the state,[14] an augmented state can be defined to include these additional elements, as described in later sections. At any moment in time, $t_0 \leq t \leq t_f$, the minimized value function or cost-to-go $V^*(t)$, corresponding to Eq. (1), can be expressed as

$$V^*[\boldsymbol{x}^*(t)] = \min_{u(t)} \left\{ \phi\left[\boldsymbol{x}^*(t_f)\right] - \int_{t_f}^{t} L[\boldsymbol{x}^*(\tau), \boldsymbol{u}(\tau)]\,\mathrm{d}\tau \right\} \tag{5}$$

The critic network evaluates the action network performance by approximating the derivative of the corresponding cost-to-go with respect to the state:

$$\boldsymbol{\lambda}^*[\boldsymbol{x}^*(t)] \equiv \frac{\partial V^*[\boldsymbol{x}^*(t)]}{\partial \boldsymbol{x}^*(t)} \tag{6}$$

This indirect measure of performance corresponds to the costate vector in the Hamilton–Jacobi–Bellman equation (see Ref. 11) and is used in the optimality condition, derived in the following section, to obtain the explicit measure $\partial V^*[\boldsymbol{x}^*(t)]/\partial \boldsymbol{u}^*(t)$.

Single-hidden-layer sigmoidal neural networks are chosen to model the action and critic functionals. They have input $\boldsymbol{p}(t) = [\boldsymbol{x}(t)^T \;\; \boldsymbol{a}(t)^T]^T$, where $\boldsymbol{a}$ is a scheduling vector of auxiliary inputs that informs the neural networks of the dynamically significant variables in the system. The network adjustable parameters consist of the input weights $\boldsymbol{W}$, output weights $\boldsymbol{V}$, and input and output biases $\boldsymbol{d}$ and $\boldsymbol{b}$. The output of the network is computed as the nonlinear transformation of the weighted sum of the input and the input bias:

$$\boldsymbol{z}[\boldsymbol{p}(t)] = \boldsymbol{V}^T \boldsymbol{\sigma}[\boldsymbol{W}\boldsymbol{p}(t) + \boldsymbol{d}] + \boldsymbol{b} \tag{7}$$

where $\boldsymbol{\sigma}[\bullet]$ is a vector-valued function composed of individual sigmoidal functions of the form $\sigma(n) \equiv (e^n - 1)/(e^n + 1)$. This architecture can approximate any nonlinear function on a compact space arbitrarily well.[15]

B. Offline Training Phase

The goal of the offline training phase is to incorporate earlier control knowledge in the neural network control system. Here, the earlier knowledge consists of well-known gain-scheduled linear controllers.[16] The training procedure is based on the observation that, to match the performance of a gain-scheduled controller, the gradients of the nonlinear neural network controller must equal the linear gain matrices at selected operating points P, indexed by $j = 1, 2, \ldots, p$. A basic assumption is that linearized models of the plant can be obtained from Eq. (2) for the subset P under the assumption of small perturbations about corresponding equilibria and by neglect of time-varying effects:

$$\Delta\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\Delta\boldsymbol{x}(t) + \boldsymbol{G}\Delta\boldsymbol{u}(t) \tag{8}$$

$\Delta \boldsymbol{x}(t_0)$ given. The optimization goals are expressed as a quadratic function of the state and control

$$J = \frac{1}{2}\int_0^{t_f} [\Delta \boldsymbol{x}^T(\tau)\boldsymbol{Q}\Delta \boldsymbol{x}(\tau) + 2\Delta \boldsymbol{x}^T(\tau)\boldsymbol{M}\Delta \boldsymbol{u}(\tau) + \Delta \boldsymbol{u}^T(\tau)\boldsymbol{R}\Delta \boldsymbol{u}(\tau)]\,\mathrm{d}\tau \tag{9}$$

When the plant is subject to continuing disturbance inputs and t_f becomes infinite in the limit, the value of J may still be bounded by definition of an average cost,

$$J_A = \lim_{t_f \to \infty} (J/t_f) \tag{10}$$

that has the same optimality conditions as J (Ref. 14). As t_f approaches infinity, it is reasonable to let the terminal cost, $\phi[\boldsymbol{x}(t_f)]$, equal zero. Furthermore, it can be shown[17] that the value function,

$$V^*[\Delta \boldsymbol{x}^*(t)] = \tfrac{1}{2}\Delta \boldsymbol{x}^{*T}(t)\boldsymbol{P}(t)\Delta \boldsymbol{x}^*(t) \tag{11}$$

is optimal for Eqs. (8) and (9), and that $\boldsymbol{P}(t)$ approaches its steady-state value $\boldsymbol{P}$. The following closed-form linear-optimal control law can be derived[14]:

$$\Delta \boldsymbol{u}^*(t) = -\boldsymbol{R}^{-1}[\boldsymbol{G}^T\boldsymbol{P} + \boldsymbol{M}^T]\Delta \boldsymbol{x}^*(t) = -\boldsymbol{C}\Delta \boldsymbol{x}^*(t) \tag{12}$$

Linear time-invariant control laws that satisfy desired engineering criteria[18] can be designed for P to provide a set of locally optimal gains and Riccati matrices $\{\boldsymbol{C}_j, \boldsymbol{P}_j\}_{j=1,\ldots,p}$. The gradient of the action network at the jth operating point, which has value in training the network offline, is found by differentiation of Eq. (4) with respect to $\boldsymbol{x}^*(t)$ by the use of the result in Eq. (12):

$$\left.\frac{\partial \boldsymbol{u}^*[\boldsymbol{x}^*(t)]}{\partial \boldsymbol{x}^*(t)}\right|_{x_0^*, a_j} = \left.\frac{\partial \Delta \boldsymbol{u}^*(t)}{\partial \Delta \boldsymbol{x}^*(t)}\right|_{\Delta x^* = 0, a_j} = -\boldsymbol{C}_j, \qquad \forall \quad j \tag{13}$$

$\boldsymbol{C}_j$ is known from the linear quadratic (LQ) optimal gain matrices, and $\boldsymbol{a}_j$ is the scheduling vector evaluated at the jth operating condition. In infinite horizon problems, the structure of the value function is independent of time; therefore, a single time-invariant critic network can be used to approximate $\boldsymbol{\lambda}^*[\boldsymbol{x}^*(t)]$, or simply $\boldsymbol{\lambda}^*(t)$ [Eq. (6)]. The LQ optimal value function, Eq. (11), can be differentiated twice with respect to the state to seek the following derivative,

$$\left.\frac{\partial \boldsymbol{\lambda}^*[\boldsymbol{x}^*(t)]}{\partial \boldsymbol{x}^*(t)}\right|_{x_0^*, a_j} = \left.\frac{\partial^2 V^*[\Delta \boldsymbol{x}^*(t)]}{\partial \Delta \boldsymbol{x}^*(t)^2}\right|_{\Delta x^* = 0, a_j} = \boldsymbol{P}_j, \qquad \forall \quad j \tag{14}$$

where, $\boldsymbol{P}_j$ is known and is used to train the critic offline.

Under the stated assumptions, the gradient $\partial \boldsymbol{z}[\boldsymbol{p}(t)]/\partial \boldsymbol{x}(t)$ is known for both the critic and the action network. In addition, the following input/output condition applies:

$$\boldsymbol{z}[\boldsymbol{x}(t), \boldsymbol{a}(t)]_{x_0^*, a_j} = \boldsymbol{0}, \qquad \forall \quad j \tag{15}$$

The size and parameters of the nonlinear neural networks are determined in one step by solution of sets of linear equations, such that the requirements in Eqs. (13–15) are matched exactly over P, as explained in Ref. 19. This phase provides an excellent starting point for the online phase, retaining the characteristics of the linear designs for small perturbations.

C. Online Training by a DHP Adaptive Critic

The online logic is implemented in discrete time through a DHP incremental optimization scheme that is based on the recurrence relation of dynamic programming. During each time interval $\Delta t = t_{k+1} - t_k$, the action and critic networks are adapted to approximate more closely the optimal control law and value function derivatives, respectively. Adaptation criteria are derived from the recurrence relation by discretization of the infinite horizon optimal control problem.[2] The recurrence relation[3] is used to predict the value function over time

$$V[\boldsymbol{x}(t_k)] = L[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k)] + V[\boldsymbol{x}(t_{k+1})] \tag{16}$$

The control $\boldsymbol{u}(t_k)$ is defined as the function of $\boldsymbol{x}(t_k)$ that minimizes the right-hand side of Eq. (16) for any $\boldsymbol{x}(t_k)$. Howard shows[3] that when the function $V[\boldsymbol{x}(t_k)]$ is calculated from Eq. (16), and the control law is adjusted to minimize the right-hand side of Eq. (16), the control law improves at every iteration (as in Ref. 20).

At time t_k, the value function is stationary provided the optimality condition is satisfied

$$\frac{\partial V[\boldsymbol{x}(t_k)]}{\partial \boldsymbol{u}(t_k)} = \frac{\partial L[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k)]}{\partial \boldsymbol{u}(t_k)} + \boldsymbol{\lambda}[\boldsymbol{x}(t_{k+1})]\frac{\partial \boldsymbol{x}(t_{k+1})}{\partial \boldsymbol{u}(t_k)} = 0 \tag{17}$$

A recurrence relation for the critic is obtained by differentiation of Eq. (16) with respect to the state:

$$\boldsymbol{\lambda}[\boldsymbol{x}(t_k)] \equiv \frac{\partial V[\boldsymbol{x}(t_k)]}{\partial \boldsymbol{x}(t_k)} = \frac{\partial L[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k)]}{\partial \boldsymbol{x}(t_k)} + \frac{\partial L[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k)]}{\partial \boldsymbol{u}(t_k)}\frac{\partial \boldsymbol{u}[\boldsymbol{x}(t_k)]}{\partial \boldsymbol{x}(t_k)} + \boldsymbol{\lambda}[\boldsymbol{x}(t_{k+1})]\frac{\partial \boldsymbol{x}(t_{k+1})}{\partial \boldsymbol{x}(t_k)} + \boldsymbol{\lambda}[\boldsymbol{x}(t_{k+1})]\frac{\partial \boldsymbol{x}(t_{k+1})}{\partial \boldsymbol{u}(t_k)}\frac{\partial \boldsymbol{u}[\boldsymbol{x}(t_k)]}{\partial \boldsymbol{x}(t_k)} \tag{18}$$

The DHP critic approximates $\boldsymbol{\lambda}[\boldsymbol{x}(t)]$; thus, it can be used to compute $\boldsymbol{\lambda}[\boldsymbol{x}(t_{k+1})]$ in Eqs. (17) and (18), once the predicted state $\boldsymbol{x}(t_{k+1})$ is obtained from the model of the plant [Eq. (2)].

III. Online Phase Implementation

The DHP criteria are implemented to adapt the action and critic networks, as shown by the flow charts in Figs. 1 and 2. The neural network weights are updated to minimize the mean-squared error between a desired output or target, denoted by $(\bullet)_D$, and the network's output, $\boldsymbol{z}[\boldsymbol{p}(t_k)]$, for a known input $\boldsymbol{p}(t_k)$. Equations (17) and (18) are used to generate the action and the critic targets, $\boldsymbol{u}_D(t_k)$ and $\boldsymbol{\lambda}_D(t_k)$, respectively. During the first time interval $(t_1 - t_0)$, the weights obtained offline are used before the online update. Later, the weights obtained online during $(t_k - t_{k-1})$ are used as earlier weights during $(t_{k+1} - t_k)$.

A. Action and Critic Network Target Generation

The action network target, $\boldsymbol{u}_D(t_k)$, is obtained by solution of the optimality condition, that is, the set of nonlinear equations in Eq. (17). A guess to the solution, $\boldsymbol{u}_D(t_k)^G$, is provided by the action network. Subsequently, it is perturbed by an established algorithm, for example, Newton–Raphson, until the stopping condition is met. Based on the prediction of $\boldsymbol{x}(t_{k+1})$, $\boldsymbol{\lambda}(t_{k+1})$ is computed by the critic network, as shown in Fig. 1. Once the action network has been updated, the critic's desired output is computed from Eq. (18) based on the exact values of $\boldsymbol{u}(t_k)$ and $\partial \boldsymbol{u}(t_k)/\partial \boldsymbol{x}(t_k)$. The derivatives $\partial L[\bullet]/\partial \boldsymbol{x}(t_k)$ and $\partial L[\bullet]/\partial \boldsymbol{u}(t_k)$ are computed analytically from $L[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k)]$. The transition matrices, $\partial \boldsymbol{x}(t_{k+1})/\partial \boldsymbol{u}(t_k)$ and $\partial \boldsymbol{x}(t_{k+1})/\partial \boldsymbol{x}(t_k)$, are obtained numerically from Eq. (2) because the model is not entirely analytical and utilizes tabulated data.

B. Online Training Algorithm

The online training algorithm minimizes an error functin E with respect to $\boldsymbol{w}$, which is a vector of ordered weights w_ℓ, indexed by $\ell = 1, 2, \ldots$:

$$E(\boldsymbol{w}) \equiv \tfrac{1}{2}\|\boldsymbol{z}_D - \boldsymbol{z}(\boldsymbol{w})\|^2 \tag{19}$$

Because the networks are updated every time a value of $\boldsymbol{x}$ is observed, this error minimization is kept local. Based on the idea of backpropagation[21] at each epoch i, the online training algorithm modifies each weight $w_\ell^{(i)}$ by a small increment $\Delta w_\ell^{(i)}$, based on the derivative $\partial E(\boldsymbol{w})/\partial w_\ell$, such that

$$w_\ell^{(i+1)} = w_\ell^{(i)} + \Delta w_\ell^{(i)} \tag{20}$$

To be viable in applications, the online phase must be reliable as well as effective. It must perform at least as well as the offline

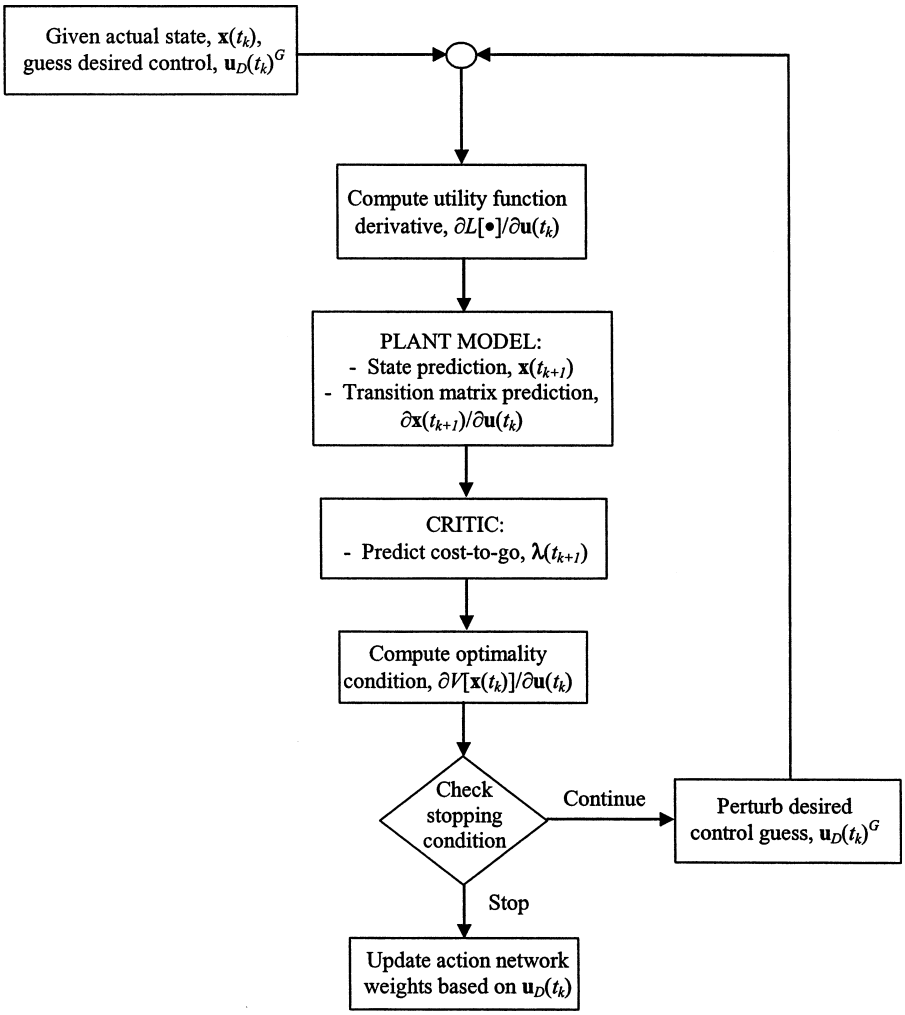


Fig. 1 DHP action network adaptation during $\Delta t = t_{k+1} - t_k$.

design, and it must improve performance quickly to impact a task, for example, an aircraft maneuver, while it is still taking place. A necessary condition for reliability is that the network update algorithm use earlier network weights to initiate the minimization of the error function E. Then, to be effective, the algorithm must decrease the network error significantly at the onset of training, that is, in a few epochs, without disregarding the earlier weights. Because of the high-dimensional nature of many applications, the neural networks implemented typically are very large and have parameters characterized by different orders of magnitude, causing the derivatives to have highly dissimilar sizes.

A modified resilient backpropagation algorithm (RPROP) is used to meet the desired objectives. Like the original RPROP algorithm (presented in Ref. 22), it considers the temporal behavior of the gradients' signs; therefore, it has low memory requirements and no dependence on the size of the derivatives. The individual size of each weight's increment, denoted by Δ_ℓ, is increased by a factor η^+ when the algorithm is converging to a minimum and the derivative is not changing sign, whereas it is decreased by a factor η^- when the algorithm is jumping over a local minimum and the derivative is changing sign. This process accelerates convergence in shallow regions and slows the search when local minima are missed. Once all Δ_ℓ are adjusted, each weight is modified in the direction of gradient descent. When the error derivative changes sign, indicating that a minimum was missed, the weight $w_\ell^{(i+1)}$ is brought back to its earlier value $w_\ell^{(i-1)}$ by a backtracking epoch.[22]

Backtracking is a key algorithmic feature that allows the search to remain local. Another crucial element is the initial increment value $\Delta_\ell^{(0)}$. The assignment of all initial increments equal to the same constant value, for example, 0.1, for weights of dissimilar size, as suggested in Ref. 22, is equivalent to forgetting the starting network weights. For this reason, the MATLAB® 5.3 implementation of Ref. 22 sends the training error to very high values before it converges to satisfactory weights. The modified RPROP takes advantage of earlier weights, that is, obtained during the preceding time step, choosing initial increments that are commensurate with a fraction f_w of the earlier weights and perturbing them by f_0 to account for zero weights:

$$\Delta_\ell^{(0)} = f_w |w_\ell| + f_0 \tag{21}$$

This weight-update routine is used for the action and the critic networks by letting $z_D = \boldsymbol{u}_D(t_k)$ in the action update and $z_D = \boldsymbol{\lambda}_D(t_k)$ in the critic update. The numerical studies in Sec. V show that, with these modifications, the network error always decreases at the onset of training, without first undergoing a significant change.

IV. Adaptive Critic Proportional–Integral Neural Network Control Design

The nonlinear control structure is obtained by simulation of that of an existing multivariable linear controller, by substitution of the linear gains by nonlinear neural networks as suggested in Refs. 19 and 23, to incorporate earlier control knowledge. In addition, a critic neural network is included to implement the DHP online phase (Sec. II.C). The design assumptions are presented in Sec. II, and the method is illustrated for a proportional–integral (PI) controller.[14] The feedback gain matrix $\boldsymbol{C}_B$, the forward gain matrix $\boldsymbol{C}_F$, and the

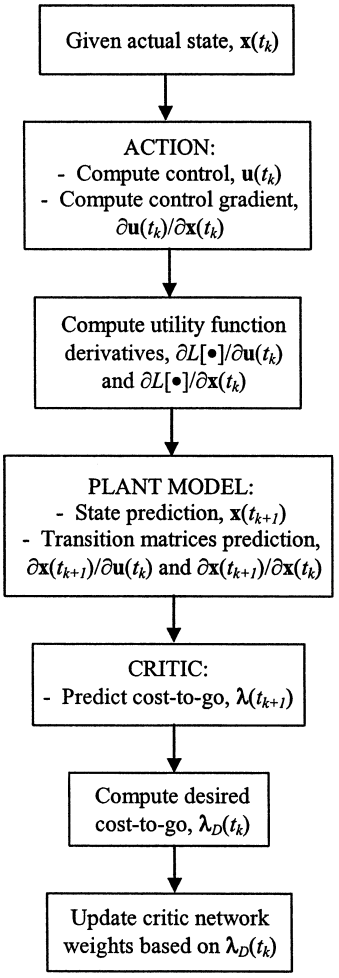


Fig. 2 DHP critic network adaptation during $\Delta t = t_{k+1} - t_k$.

command-integral gain matrix $\boldsymbol{C}_I$, are computed to minimize the following cost function:

$$J = \lim_{t_f \to \infty} \int_0^{t_f} L[\boldsymbol{x}_a(\tau), \tilde{\boldsymbol{u}}(\tau)]\,\mathrm{d}\tau = \lim_{t_f \to \infty} \int_0^{t_f} \frac{1}{2}\left[\boldsymbol{x}_a^T(\tau)\boldsymbol{Q}\boldsymbol{x}_a(\tau) + 2\boldsymbol{x}_a^T(\tau)\boldsymbol{M}\tilde{\boldsymbol{u}}(\tau) + \tilde{\boldsymbol{u}}^T(\tau)\boldsymbol{R}\tilde{\boldsymbol{u}}(\tau)\right]\mathrm{d}\tau \tag{22}$$

where $\boldsymbol{x}_a$ represents an augmented state that includes the state deviation $\tilde{\boldsymbol{x}}$ and the output error's time integral $\boldsymbol{\xi}$, that is, $\boldsymbol{x}_a \equiv [\tilde{\boldsymbol{x}}^T \;\; \boldsymbol{\xi}^T]^T$, where $\tilde{\boldsymbol{x}} \equiv \boldsymbol{x} - \boldsymbol{x}_c$. The output error $\tilde{\boldsymbol{y}}$ and the control deviation $\tilde{\boldsymbol{u}}$ are similarly defined. The set point $(\boldsymbol{x}_c, \boldsymbol{u}_c)$ is a function of the command input $\boldsymbol{y}_c$ (Ref. 14). The LQ law [Eq. (12)] provides for the optimal control in terms of the newly defined deviations:

$$\tilde{\boldsymbol{u}}(t) = -\boldsymbol{C}_a\boldsymbol{x}_a(t) = -\boldsymbol{C}_B\tilde{\boldsymbol{x}}(t) - \boldsymbol{C}_I\boldsymbol{\xi}(t) \tag{23}$$

The gains and the matrix $\boldsymbol{P}_a$ are obtained by solution of a matrix Riccati equation (see Ref. 14) formulated in terms of $\boldsymbol{x}_a$ and $\tilde{\boldsymbol{u}}$. The weighting matrices $\boldsymbol{Q}$, $\boldsymbol{M}$, and $\boldsymbol{R}$, are designed with implicit model following, based on an ideal model that satisfies established design criteria.[24,25]

In the nonlinear control structure, each linear gain matrix is replaced by a nonlinear control network, $\boldsymbol{NN}_B$ for $\boldsymbol{C}_B$, $\boldsymbol{NN}_F$ for $\boldsymbol{C}_F$, and $\boldsymbol{NN}_I$ for $\boldsymbol{C}_I$, as shown in Fig. 3. In addition to the scheduling vector $\boldsymbol{a}$, the networks $\boldsymbol{NN}_B$, $\boldsymbol{NN}_F$, and $\boldsymbol{NN}_I$ are provided with $\tilde{\boldsymbol{x}}$, $\boldsymbol{y}_c$, and $\boldsymbol{\xi}$, respectively. Each network contributes to the total control,

$$\begin{aligned}\boldsymbol{u}(t) &= \boldsymbol{u}_c(t) + \Delta\boldsymbol{u}_B(t) + \Delta\boldsymbol{u}_I(t)\\ &= \boldsymbol{NN}_F[\boldsymbol{y}_c(t), \boldsymbol{a}(t)] + \boldsymbol{NN}_B[\tilde{\boldsymbol{x}}(t), \boldsymbol{a}(t)] + \boldsymbol{NN}_I[\boldsymbol{\xi}(t), \boldsymbol{a}(t)]\end{aligned} \tag{24}$$

where $\tilde{\boldsymbol{u}} = \Delta\boldsymbol{u}_B + \Delta\boldsymbol{u}_I$ is the control to be optimized.

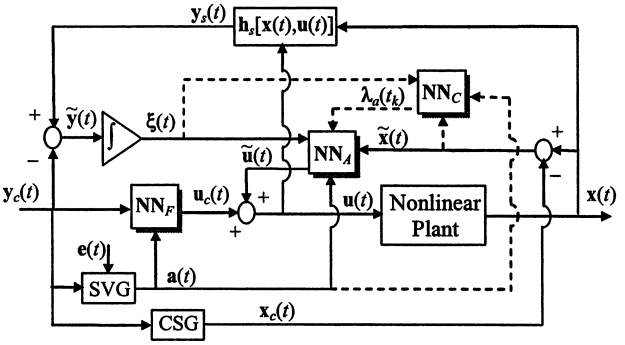


Fig. 3 Action critic neural network controller.

The action network $\boldsymbol{NN}_A$, approximates the minimizing control law:

$$\tilde{\boldsymbol{u}}(t) = \boldsymbol{NN}_A[\boldsymbol{x}_a(t), \boldsymbol{a}(t)] \tag{25}$$

Given the same inputs, the critic network $\boldsymbol{NN}_C$, computes the derivative of the value function $V[\boldsymbol{x}_a(t)]$ corresponding to Eq. (22):

$$\boldsymbol{\lambda}_a(t) \equiv \frac{\partial V[\boldsymbol{x}_a(t)]}{\partial \boldsymbol{x}_a(t)} = \boldsymbol{NN}_C[\boldsymbol{x}_a(t), \boldsymbol{a}(t)] \tag{26}$$

The scheduling variable generator (SVG) contains algebraic equations that produce $\boldsymbol{a}$ based on $\boldsymbol{y}_c$ and an exogenous vector $\boldsymbol{e}$ of measured variables. The command state generator (CSG) provides secondary elements of the state that are compatible with $\boldsymbol{y}_c$. Both blocks are obtained from the governing equations (2), according to well-established techniques.[25] The action and critic networks are trained offline by the use of $\{\boldsymbol{C}_{B_j}, \boldsymbol{C}_{I_j}, \boldsymbol{P}_{a_j}\}_j$, $j \in P$ (see Ref. 19). Equations (17) and (18) are reformulated in terms of $\boldsymbol{x}_a$ and $\tilde{\boldsymbol{u}}$, to train the action and the critic networks on line.

V. Flight Control Simulation and Results

The adaptive controller is implemented for the control of a six-degree-of-freedom simulation of a business jet aircraft.[26–29] The simulated aircraft explores its full flight envelope, $R = \{V, H, \gamma, \mu, \beta\}$, that is, the set of all possible combinations of the enclosed variables.

The control design is based on the state vector, $\boldsymbol{x} = [V \;\; \gamma \;\; q \;\; \theta \;\; r \;\; \beta \;\; p \;\; \mu]^T$, comprising airspeed V (meters per second), path angle γ (radians), pitch rate q (radians), pitch angle θ (radians), yaw rate r (radian per second), sideslip angle β (radians), roll rate p (radian per second), and bank angle μ (radians). The independent controls being generated are throttle δT (percent), stabilator δS (radians), aileron δA (radians), and rudder δR (radians), that is, $\boldsymbol{u} = [\delta T \;\; \delta S \;\; \delta A \;\; \delta R]^T$. The command input, $\boldsymbol{y}_c = [V_c \;\; \gamma_c \;\; \mu_c \;\; \beta_c]^T$, contains the state elements that, given the altitude H (meters), uniquely specify a longitudinal–lateral-directional steady maneuver, for example, a coordinated turn, postulating $\dot{\phi}_c = \dot{\theta}_c = 0$ with ϕ as the Euler roll angle. Because three-dimensional maneuvering flight is considered, the SVG and CSG blocks in Fig. 3 are designed with nonlongitudinal, non-level flight angular and kinematic relations and spherical trigonometry.[27]

The forward neural network $\boldsymbol{NN}_F$ is trained offline to approximate the aircraft trim map,[30] and it is held fixed online to compute the control settings corresponding to $\boldsymbol{y}_c$, based on the aircraft model, Eq. (2). The action and critic networks, $\boldsymbol{NN}_A$ and $\boldsymbol{NN}_C$, are trained algebraically offline by the use of the linear controllers obtained for a set P of 34 steady-level flight conditions.[19,25] During the online phase, the action and critic networks are adapted to improve on their performance in R. The adaptation time interval Δt is chosen equal to or greater than the Runge–Kutta (RK)-integration time step. During every 0.1-s time interval, the critic and action networks are updated by the modified RPROP algorithm based on the respective targets, $\tilde{\boldsymbol{u}}_D$ and $(\boldsymbol{\lambda}_a)_D$. The update parameters defined by the user are $\eta^+ = 1.2$, $\eta^- = 0.5$[22], $f_w \sim \mathcal{O}(10^{-5})$, and $f_0 \ll 1$.

The performance of the modified RPROP algorithm is compared to that of the MATLAB® 5.3 RPROP-based learning function

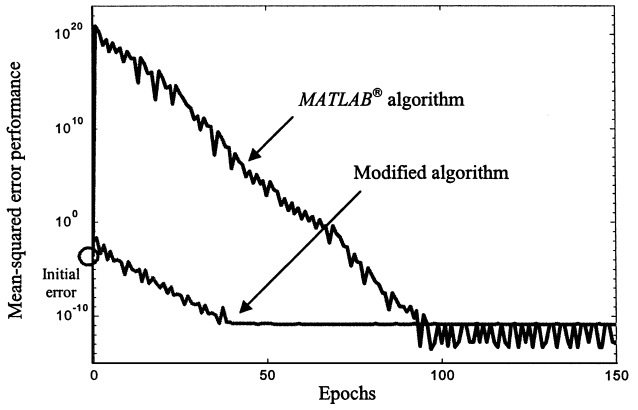


Fig. 4 Performance comparison between the MATLAB® resilient backpropagation algorithm and its modified version, for the action network training at $t = 0.2$ s.

"trainrp" in Fig. 4, where the action network update at $t_k = 0.2$ s is shown over 150 epochs. These results are representative of several simulations involving the update of action and critic neural networks at different time steps. Typically, the modified algorithm begins decreasing the error by the third epoch and approaches the same performance as the MATLAB function in one-half the number of epochs. Thus, only a few epochs are needed to decrease the network error significantly during one time step. Further studies confirm that the modified RPROP better preserves initial weights and avoids overfitting.[25]

The termination rule stops the online training of the action and the critic network after the mean-squared measure of the network error, $[z_D - z(\boldsymbol{w})]$, has decreased by 10% and at least three epochs have elapsed. When more than three epochs are needed to decrease the network error by this amount, the terminating value of Δ_ℓ is saved and used as $\Delta_\ell^{(0)}$ for the next time interval, for $\forall\, \ell$. Typically, the modified RPROP algorithm runs for several epochs during the first two or three time intervals, for example, 0.3 s, because it needs to adjust the increment size (Sec. III.B.); during later time intervals, three epochs are sufficient to decrease the network error by 10%.

In the numerical studies, the controller learns from the simulation's nonlinear and coupling effects that were missed by the linearizations and adapts to unforeseen failures and parameter variations. The adaptation's progress is monitored by recording the optimality condition [Eq. (17)] and the mean-squared action- and critic-network errors at every time step. The state response to step command inputs is also used to assess the controller's overall performance. Extensive numerical experiments show that the adaptation improves performance quickly enough to impact a new maneuver while it is still taking place. Furthermore, while adapting to new non-steady flight conditions, the adaptive controller preserves, and only improves on, any knowledge already assimilated in R.

A. Case 1: Adaptive Control During a Coupled Maneuver

The aircraft response is considered during a large-angle asymmetric maneuver, for which the longitudinal–lateral–directional coupling effects neglected by the offline designs are significant. Initially, the aircraft is flying steady and level, at a nominal airspeed V_0 of 95 m/s and an altitude H_0 of 2000 m, where $(V_0, H_0) \not\subset P$. At time $t = 0$, a step command consisting of a 5-deg climb angle and 30-deg roll angle is initiated, as would be required to perform a climbing steady turn. (The other commands remain equal to their nominal value.) The response of the aircraft subject to the adaptive controller is plotted with a solid line in Fig. 5, together with that of the aircraft subject to the controller with parameters held fixed, represented by a dashed line. The adaptation reduces the amplitude of the velocity, path angle, and sideslip oscillations and displays the desired transient characteristics specified through implicit model following.[24,25] The adaptive controller's time history is compared to that of the fixed controller in Fig. 6, showing a minor difference in control usage. Following this maneuver, additional tests show

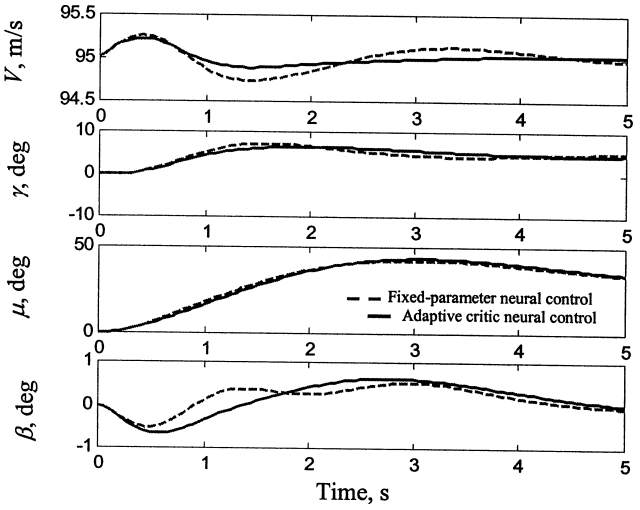


Fig. 5 Comparison between the adaptive critic neural network controller and the neural network controller with parameters held fixed at $(V_0, H_0) = (95$ m/s, 2 km), subject to 5-deg climb angle and 30-deg roll angle step command.

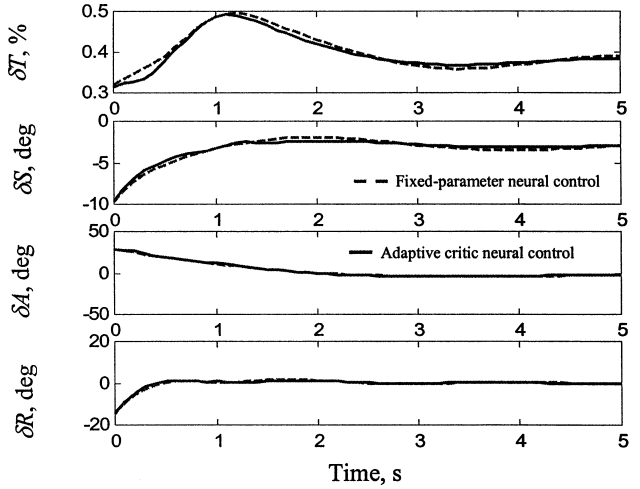


Fig. 6 Comparison between the online trained adaptive critic neural network control history and the fixed-parameter neural network control history subject to 5-deg climb angle and 30-deg roll angle step command at $(V_0, H_0) = (95$ m/s, 2 km).

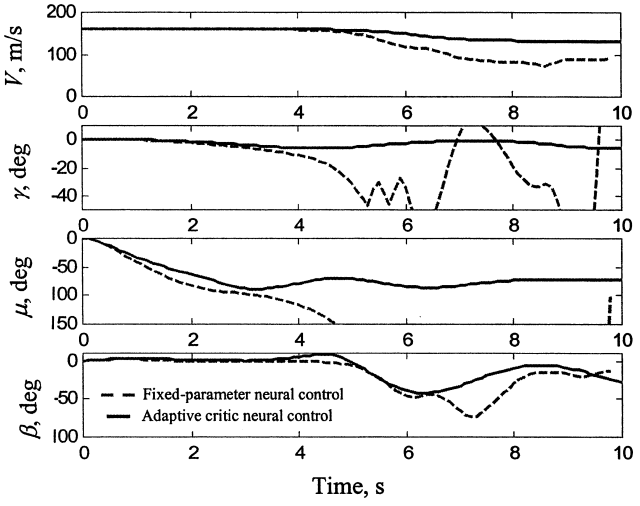


Fig. 7 Comparison between the online trained adaptive critic neural network controller and the fixed-parameter neural network controller subject to −70-deg roll angle step command at $(V_0, H_0) = (160$ m/s, 7 km).

that the weights of the action and critic networks have undergone a small change and that their gradients over P have remained very close to the corresponding linear gains learned offline, with a total mean-squared difference of $\mathcal{O}(10^{-4})$. This indicates that, although it has improved performance over this new region of the state space (Fig. 5) where $\mu \neq 0$, the adaptive controller has preserved earlier knowledge of the optimal controllers over the steady-level flight envelope P.

B. Case 2: Adaptive Control During a Large-Angle Maneuver

The adaptive controller is implemented on a large-angle maneuver for which the nonlinear and coupling effects are so significant that they would otherwise lead to closed-loop instability. To demonstrate this capability, a −70-deg turn is commanded while the aircraft is flying steady and level at the nominal airspeed and altitude of 160 m/s and 7000 m. At this angle, the aircraft cannot produce sufficient lift to maintain altitude, and the coupled dynamics become so significant as to compromise any decoupled control design; also, it becomes more difficult to coordinate the turn. Although not a normal maneuver, these conditions could come about in an emergency situation.

With control parameters held fixed, represented by a dashed line in Fig. 7, the aircraft departs from controlled flight. The roll and climb angles increase beyond acceptable limits, and the aircraft enters a stall. At this point, the simulation is not a faithful representation of the aircraft dynamics because poststall aerodynamic effects are not modeled. Still, the uncoupled control design causes the aircraft to gyrate wildly, and it is not capable of recovering from this maneuver. Figure 7 also shows the response of the adaptive controller (solid line) for the same flight conditions. In this case, the control system learns from the nonlinear aircraft dynamics and adjusts the network weights on line, preventing loss of stability.

Under challenging circumstances, the tendency is for the system to demand unreasonable control usage. Soft control bounds are easily accounted for in the adaptive critic architecture by allowance of the weighting matrix $\boldsymbol{R}$ to vary exponentially with respect to control inputs that exceed physical limitations.[25] The throttle is bounded between 0 and 100%. The stabilator, aileron, and rudder deflections cannot exceed ±0.6 rad. The time histories of the fixed and adaptive controllers that produce the aircraft response in Fig. 7 are plotted in Fig. 8. The adaptive controller improves performance considerably over time and learns the control bounds online. The result is that the adaptive controller can sustain the desired banked turn, whereas the fixed controller, based on uncoupled linear designs, leads to a hazardous maneuver.

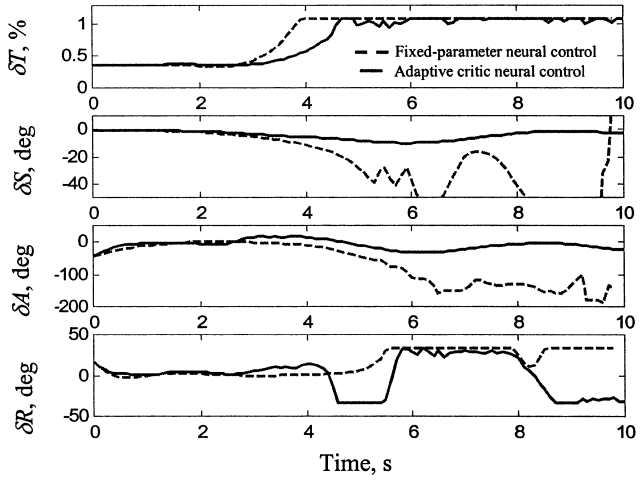


Fig. 8 Comparison between the adaptive critic neural network control history and the fixed-parameter neural network control history subject to −70-deg roll angle step command at $(V_0, H_0) = (160$ m/s, 7 km).

C. Case 3: Adaptive Control During Multiple Control Failures

The capability of the adaptive control system to handle a near-emergency situation is considered by the simulation of control failures during an approach to landing. The aircraft, initially flying at $(V_0, H_0) = (100$ m/s, 3000 m), begins its final approach by decreasing its velocity and performing a descending turn with −6-deg climb angle and 50-deg roll angle for 10 s. During this time, multiple control failures occur, impairing control of the aircraft. Both engines produce no thrust; the rudder and stabilator are stuck at 0 deg for $5 \text{ s} \leq t_k \leq 10$ s, and the rudder is stuck at −34 deg for $0 \text{ s} \leq t_k \leq 5$ s. The airplane enters a steep dive with a large roll angle

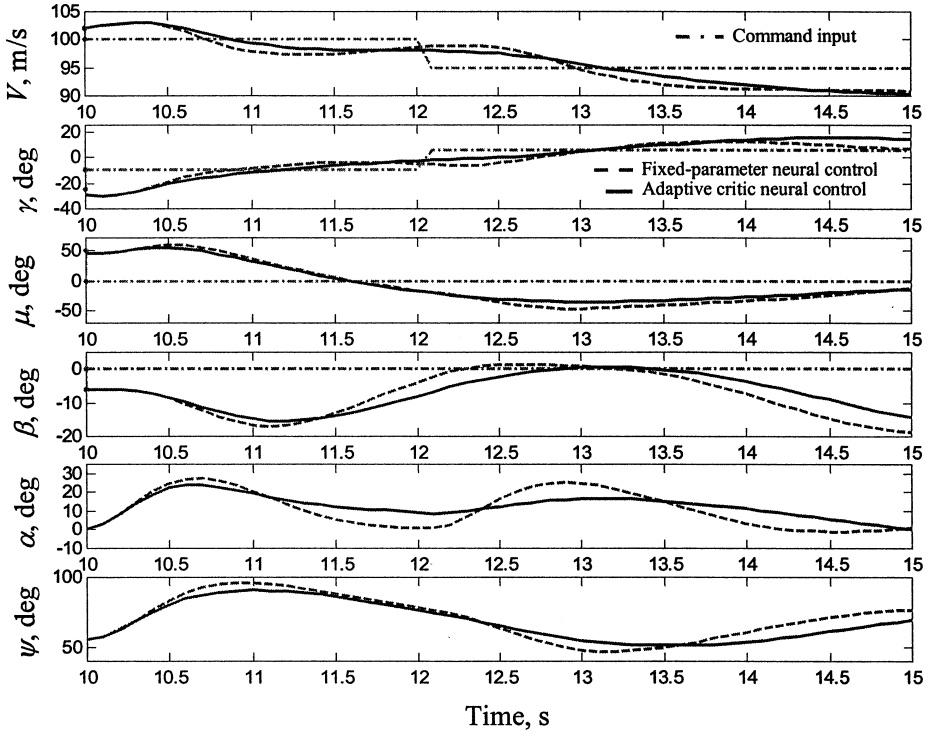


Fig. 9 Comparison between the adaptive and the fixed-parameter neural controllers in the presence of multiple control failures.

and fast accelerations. This critical situation is simulated to compare the adaptive and the fixed control systems during a recovery maneuver with reduced but sufficient control authority. Reduced control power prevents precise command-input tracking over short time periods. Therefore, improved performance is demonstrated by reduced oscillations and a smaller accumulated cost [Eq. (1)].

Assume that, 10 s after the initial failures, the simulated pilot or guidance logic becomes aware of the failures and initiates a wings-level climb to avoid obstacles on the ground. In the meantime, the stabilator has become fully operational, and the available throttle is increased to 50% (as by the restoration of full thrust to a single engine); the rudder is stuck at −15 deg. First, the wings are brought back to level by a 0-deg roll angle command for 2 s. Then, an airspeed of 95 m/s and a 5-deg path angle are commanded to climb for 3 s. The response of the adaptive controller is compared to that of the controller with fixed parameters, with the integrator reset state to zero in both cases to avoid the phenomenon known as integrator windup.[31]

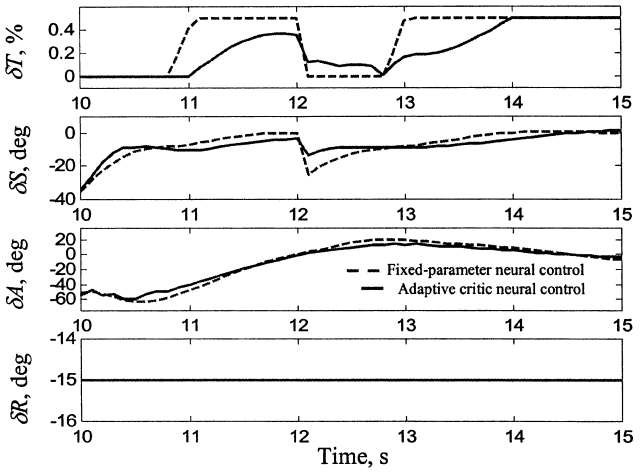


Fig. 10 Adaptive and fixed-parameter neural control histories with 50%-available thrust and the rudder stuck at −15 deg.

Figure 9 shows that the adaptation improves the command-input response, at times by more than 30%, even though these conditions are being experienced for the first time. All relevant state histories, including total airspeed, are improved on by the adaptive critic architecture. Despite less throttle usage, the velocity and path angle are followed more closely because the adaptive-controlled aircraft experiences smaller angles of attack and sideslip and, hence, less drag. The adaptation also diminishes the amplitude of the roll and heading-angle oscillations. The adaptive and fixed-control time histories are shown in Fig. 10. Because of the limited (50%) available thrust, the throttle-input profile is significantly modified, and its usage is more evenly distributed over the time interval. With the rudder stuck at −15 deg, the lateral–directional response is improved by adaptation of the aileron control input.

D. Case 4: Adaptive Control in the Presence of Parameter Variations

The adaptive controller is tested for a case in which the parameters of the simulated aircraft have changed with respect to the model [Eq. (2)] used for offline training. All control effectors are assumed to be unfailed. The pitch-rate and angle-of-attack-rate effects of the controls are decreased by 50%, and the static and directional stability coefficients are reduced by 20 and 30%, respectively. With the original aircraft parameters unchanged, that is, perfect modeling, the response of the fixed controller subject to a small-angle command input can be considered to be optimal in the neighborhood of a design point with $(V_0, H_0) \subset P$. Because of modified aerodynamic effects, the actual dynamic characteristics differ from those accounted for by the linear design. Therefore, the performance of the fixed controller is degraded with respect to its original baseline, as shown in Figs. 11 and 12.

Although the DHP adaptive critic architecture employs an imperfect model, it can learn about the new dynamics through its knowledge of the actual state. The simulated aircraft with modified parameters undergoes the small-angle maneuver shown in Fig. 11, near the design point with $(V_0, H_0) = (200$ m/s, 11,000 m). After experiencing a command input of 2-deg path angle, 5-deg roll, and 3-deg sideslip for 5 s, the adaptive controller's performance begins to approach the original baseline. Figure 12 shows that, over time, the adaptation reduces the control usage with respect to the fixed

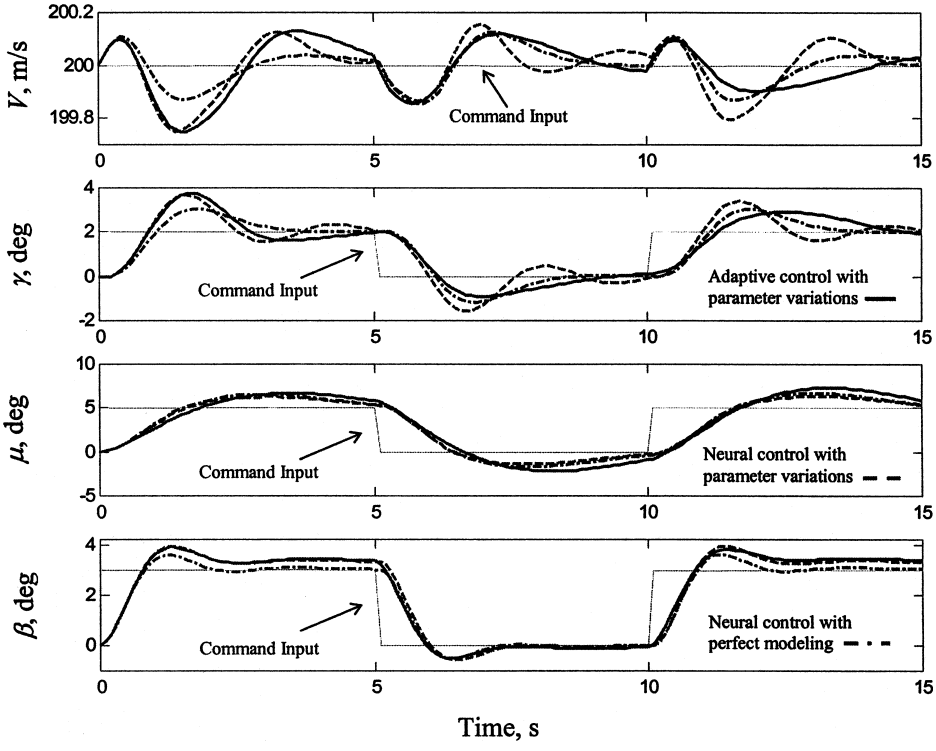


Fig. 11 Comparison between the adaptive neural controller and the fixed-parameter neural controller in the presence of aircraft parameter variations at $(V_0, H_0) = (200$ m/s, 11 km).

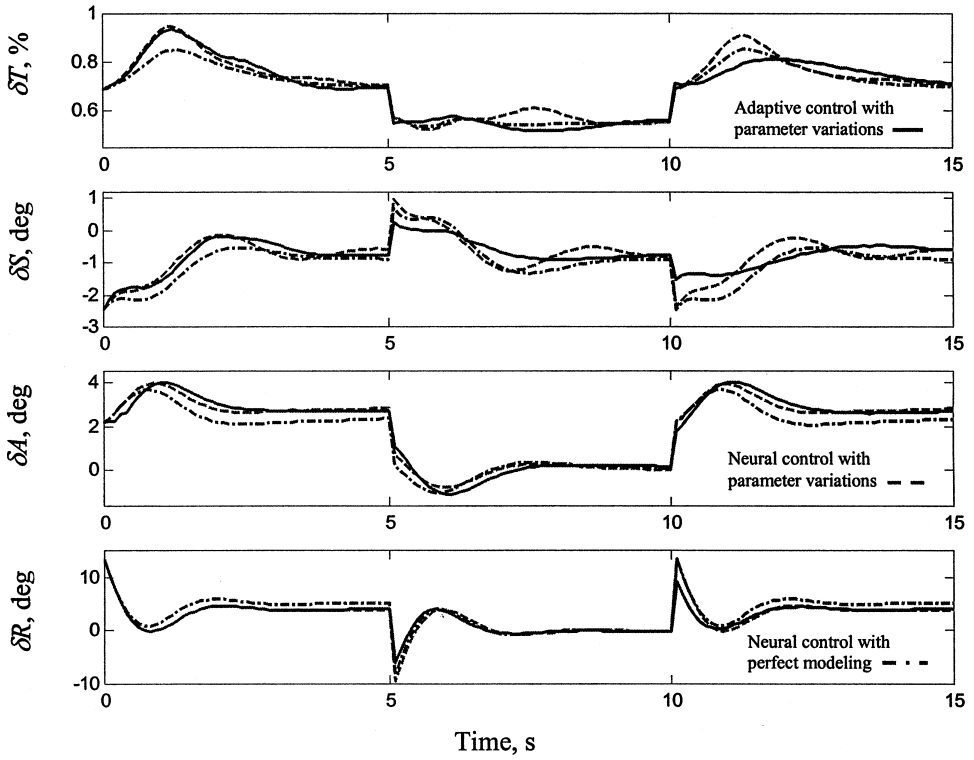


Fig. 12 Control history of the adaptive neural controller and the fixed-parameter neural controller in the presence of aircraft parameter variations at (V_0, H_0) = (200 m/s, 11 km).

controller subject to the modified parameters and even subject to the original aircraft parameters, that is, with perfect modeling. The action and critic networks learn how to minimize the cost-to-go online, in the presence of coupling and nonlinear effects, control failures, and parameter variations that were unaccounted for a priori, without unlearning previous information.

VI. Conclusions

Advances in neural network training techniques and adaptive critic methods are presented and incorporated in a novel approach to neural network control design. The nonlinear control system is trained in two phases. An offline phase provides for reliability, and an online phase improves performance subject to actual plant dynamics. Both phases are founded on optimal control theory and are realized with significant computational savings through a novel algebraic training approach and a modified online training algorithm. The adaptive controller is successfully implemented on a full-scale aircraft simulation. Both the action and critic neural networks learn newly available information online, while retaining their baseline performance. The result is a flight control system that improves performance with respect to its offline specifications when subject to unexpected conditions such as unmodeled dynamics, control failures, and parameter variations.

Acknowledgment

This research has been supported by the Federal Aviation Administration and NASA under FAA Grant 95-G-0011.

References

[1]Bellman, R. E., *Dynamic Programming*, Princeton Univ. Press, Princeton, NJ, 1957.

[2]Kirk, D. E., *Optimal Control Theory: An Introduction*, Prentice–Hall, Englewood Cliffs, NJ, 1970, Chaps. 1–3.

[3]Howard, R., *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, MA, 1960, pp. 42, 43.

[4]Venayagamoorthy, G. K., Harley, R. G., and Wunsch, D. C., "Comparison of Heuristic Dynamic Programming and Dual Heuristic Programming Adaptive Critics for Neurocontrol of a Turbogenerator," *IEEE Transactions on Neural Networks*, Vol. 13, No. 3, 2002, pp. 764–773.

[5]Lendaris, G. G., Schultz, L., and Shannon, T. T., "Adaptive Critic Design for Intelligent Steering and Speed Control of a 2-Axle Vehicle," *Proceedings of the International Joint Conference on Neural Networks*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2000.

[6]Han, D., and Balakrishnan, S. N., "Adaptive Critic Based Neural Networks for Control-Constrained Agile Missile Control," *Proceedings of the American Control Conference*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1999, pp. 2600–2604.

[7]Saini, G., and Balakrishnan, S. N., "Adaptive Critic Based Neurocontroller for Autolanding of Aircraft," *Proceedings of the American Control Conference*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1997, pp. 1081–1085.

[8]Balakrishnan, S. N., and Biega, V., "Adaptive-Critic-Based Neural Networks for Aircraft Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 4, 1996, pp. 893–898.

[9]KrishnaKumar, K., and Neidhoefer, J., "Immunized Adaptive Critics for Level-2 Intelligent Control," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, IEEE Publications, Piscataway, NJ, 1997, pp. 856–861.

[10]Werbos, P. J., "Approximate Dynamic Programming for Real-Time Control and Neural Modeling," *Handbood of Intelligent Control*, edited by D. A. White and D. Sofge, Van Nostrand Reinhold, New York, 1992, pp. 493–526.

[11]Prokorov, D. V., and Wunsch, D. C., "Adaptive Critic Designs," *IEEE Transactions on Neural Networks*, Vol. 8, No. 5, 1997, pp. 997–1007.

[12]Si, J., and Wang, Y. -T., "On-Line Learning Control by Association and Reinforcement," *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, 2001, pp. 264–276.

[13]Lendaris, G. G., and Shannon, T., "Application Considerations for the DHP Methodology," *Proceedings of the International Joint Conference on Neural Networks*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1998.

[14]Stengel, R. F., *Optimal Control and Estimation*, Dover, New York, 1994.

[15]Barron, A. R., "Universal Approximation Bounds for Superposition of a Sigmoidal Function," *IEEE Transactions on Information Theory*, Vol. 39, No. 3, 1993, pp. 930–945.

[16]Shamma, J. S., and Athans, M., "Guaranteed Properties of Gain Scheduled Control for Linear Parameter-Varying Plants," *Automatica*, Vol. 27, No. 3, 1991, pp. 55–56.

[17]Åström, K. J., and Stewart, G. W., "Solution of the Matrix Equation $AX + XB = C$," *Communications of the ACM*, Vol. 15, Feb. 1972, pp. 820–826.

[18]Stengel, R. F., and Marrison, C., "Design of Robust Control Systems for Hypersonic Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 1, 1997, pp. 58–63.
[19]Ferrari, S., and Stengel, R. F., "Classical/Neural Synthesis of Nonlinear Control Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 3, 2002, pp. 442–448.
[20]Ferrari, S., and Stengel, R. F., "Model-Based Adaptive Critic Designs," *Learning and Approximate Dynamic Programming*, edited by J. Si, A. Barto, and W. Powell, Wiley (in press).
[21]Werbos, P. J., "Backpropagation Through Time: What It Does and How to Do It," *Proceedings of the IEEE*, Vol. 78, No. 10, 1990, pp. 1550–1560.
[22]Reidmiller, M., and Braun, H., "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," *Proceedings of the IEEE International Conference on NN (ICNN)*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1993.
[23]Narendra, K. S., and Parthasaranthy, K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions Neural Networks*, Vol. 1, No. 1, 1990, pp. 4–27.
[24]Huang, C., and Stengel, R. F., "Restructurable Control Using Proportional–Integral Model Following," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 2, 1990, pp. 303–309.
[25]Ferrari, S., "Algebraic and Adaptive Learning in Neural Control Systems," Ph.D. Thesis, Dept. of Mechanical and Aerospace Engineering, Princeton Univ., Princeton, NJ, 2002.
[26]Stengel, R. F., *Flight Dynamics*, Princeton Univ. Press, Princeton (in press).
[27]Kalviste, J., "Spherical Mapping and Analysis of Aircraft Angles for Maneuvering Flight," *Journal of Aircraft*, Vol. 24, No. 8, 1987, pp. 523–530.
[28]Etkin, B., *Dynamics of Atmospheric Flight*, Wiley, Toronto, 1972.
[29]Nelson, R. C., *Flight Stability and Automatic Control*, McGraw–Hill, New York, 1989.
[30]Ferrari, S., and Stengel, R. F., "Algebraic Training of a Neural Network," *Proceedings of the American Control Conference*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2001, pp. 1605–1610.
[31]Friedland, B., "Observers," *The Control Handbook*, edited by W. S. Levine, CRC Press, Boca Raton, FL, 1996, pp. 607–618.