Proceedings of the American Control Conference
Anchorage, AK May 8-10, 2002

An Adaptive Critic Global Controller

Silvia Ferrari* and Robert F. Stengel†

Princeton University
Department of Mechanical and Aerospace Engineering
Princeton, NJ 08544

Abstract

A nonlinear control system comprising a network of networks is taught using a two-phase learning procedure realized through novel techniques for initialization, on-line training, and adaptive critic design. The neural networks are initialized algebraically by observing that the gradients of the networks must equal corresponding linear gain matrices at chosen operating points. On-line learning is based on a dual heuristic adaptive critic architecture that improves control for large, coupled motions by accounting for plant dynamics and nonlinear effects. The result is an adaptive controller that is as conservative as the linear designs and as effective as the global controller. The design method is implemented to control the full six-degree-of-freedom simulation of a business jet aircraft.

1. Introduction

The problem of optimizing a desired metric over time lies at the basis of virtually all robust and fault-tolerant control and identification schemes. Dynamic programming uses the principle of optimality to find an optimal strategy of action in a nonlinear environment. *Backwards* or *discrete* dynamic programming methods discretize the state space and make a direct comparison of the cost associated with all feasible trajectories, guaranteeing solution of the optimal control problem [1]. This approach leads to a number of computations that grows exponentially with the number of state variables ("curse of dimensionality") [2]. Adaptive critic designs constitute a class of *approximate* dynamic programming methods [3] that uses incremental optimization combined with a parametric structure to efficiently approximate the optimal cost and control. They optimize a short-term cost metric that ensures optimization of the cost over all future times. Neural networks are the parametric structures of choice, because they easily handle large-dimensional input and output spaces and can learn in an incremental fashion.

The simplest adaptive critic architectures are based on heuristic dynamic programming (HDP). They implement a critic network to approximate the cost-to-go in the Bellman equation [2] and an action network to approximate the optimal control law. This paper presents a design approach based on a modification of HDP, referred to as dual heuristic programming (DHP), where the critic network approximates the derivatives of the cost-to-go with respect to the state. DHP is more promising than its earlier counterpart because it learns more quickly and alleviates persistence of excitation problems by computing the correlation between the cost and the individual state elements [4].

The advantages brought about by using prior knowledge in conjunction with on-line training are widely recognized in the neurocontrol literature [5]. In the present approach, the nonlinear control system, comprising a network of networks, is taught using a two-phase learning procedure. During the first phase, referred to as *initialization*, the network size and parameters are determined from well-established linear control theory solely by solving algebraic equations that identify the exact matching of gain matrices at chosen operating points. During a second phase, on-line learning by a DHP approach improves control response for large, coupled motions, based on the actual state of the plant. This on-line phase accounts for differences between actual and assumed dynamic models and for nonlinear effects not captured by the linear designs. Classical control theory provides a unifying framework for the two training phases. The algebraic initialization is based on the linear quadratic regulator; the DHP approach is based on approximate dynamic programming.

2. Foundations

The goal of the adaptive critic design is to approximate the optimal control law for an infinite horizon problem subject to the real-time dynamics of a continuous plant or simulation. The neural controller adapts on line, with the plant operating over the entire range of state and command-input elements, $\{\mathbf{x}(\mathbf{y}_c), \mathbf{y}_c\}$, or some suitably dense set in the space denoted by *OR*. The plant state, $\mathbf{x}$, and the command input, $\mathbf{y}_c$, are fed to the controller on-line and are unknown prior to operation. It is assumed that linearized time-invariant plant models are known *a priori* for a subset of operating points, $OP \subset OR$. Corresponding linear control data are used to initialize the action and critic neural networks. These networks are further adjusted over time through the DHP architecture sketched in Fig. 1.

---

[1] Graduate Student.
† Professor. Fellow IEEE, AIAA.
Presented at the 2002 American Control Conference,
Anchorage, AK, May 2002.

2.1. Problem Statement

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

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

The objective is to determine the control law for which this cost function is stationary, subject to the dynamic equation:

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t)], \ \mathbf{x}(t_0) \text{ given} \quad (2)$$

Plant motions and controls are sensed in the $e_s \times 1$ output vector $\mathbf{y}_s$,

$$\mathbf{y}_s(t) = \mathbf{h}_s[\mathbf{x}(t), \mathbf{u}(t)] \quad (3)$$

It is assumed that perfect measurements are available and that the output views all elements of the state. The mission goals are expressed by the $e_c \times 1$ command input, $\mathbf{y}_c$, which can be viewed as some desirable combination of state and control elements with $e_c \leq m$.

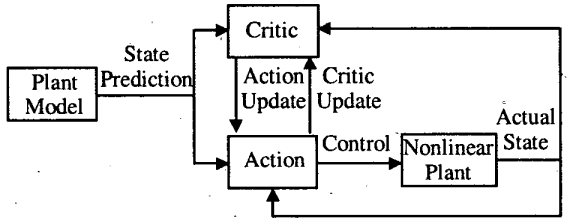

Figure 1. Dual heuristic programming adaptive critic.

The *action network* models the control law, which is assumed to be a function of the state. It can be written as the sum of a nominal and a perturbed effect,

$$\mathbf{u}^*[\mathbf{x}^*(t)] = \mathbf{u}_0^*[\mathbf{x}_0^*(t)] + \Delta\mathbf{u}^*[\mathbf{x}_0^*(t), \Delta\mathbf{x}^*(t)] \quad (4)$$

where, $\mathbf{x}^*(t) = \mathbf{x}_0^*(t) + \Delta\mathbf{x}^*(t)$, and $(\bullet)^*$ denotes the optimal solution. When the control law depends on parameters and command inputs as well as the state [6], 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:

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

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

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

Single-hidden-layer sigmoidal neural networks of the type shown in Fig. 2 are chosen to model the action and critic functionals. They have input $\mathbf{p}(t) = [\mathbf{x}(t)^T \ \mathbf{a}(t)^T]^T$, where $\mathbf{a}$ 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, $\mathbf{W}$, of the output weights, $\mathbf{V}$, and of the input and output biases, $\mathbf{d}$ and $\mathbf{b}$. The output of the network is computed as the nonlinear transformation of the weighted sum of the input and the input bias:

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

$\boldsymbol{\sigma}[\bullet]$ 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 [7].

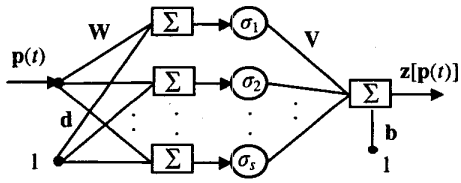

Figure 2. Sample vector-input vector-output sigmoidal network with s nodes in the hidden layer.

2.2. Initialization Phase

The goal of the initialization phase is to incorporate linear control knowledge in the nonlinear control system. The procedure is based on the observation that the network gradients must equal corresponding linear control matrices at selected operating points, OP, indexed by $\kappa = 1, 2, \ldots, p$. Linearized models of the plant can be obtained from eq. 2 for the subset OP by assuming small perturbations about corresponding equilibria, and ignoring time-varying effects:

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{F}\Delta\mathbf{x}(t) + \mathbf{G}\Delta\mathbf{u}(t), \ \Delta\mathbf{x}(t_0) \text{ given} \quad (8)$$

The optimization goals are expressed as a quadratic function of the state and control

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

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 defining an average cost,

$$J_A = \lim_{t_f \to \infty} \frac{J}{t_f} \quad (10)$$

that has the same optimality conditions as J [6]. As t_f approaches infinity, it is reasonable to let the terminal cost, $\phi[\mathbf{x}(t_f)]$, equal zero. Furthermore, it can be shown [8] that the value function,

$$V^*[\Delta\mathbf{x}^*(t)] = \frac{1}{2}\Delta\mathbf{x}^{*T}(t)\mathbf{P}(t)\Delta\mathbf{x}^*(t) \quad (11)$$

is optimal for eq. 8 and 9, and that $\mathbf{P}(t)$ approaches its steady-state value $\mathbf{P}$. The following closed-form linear-optimal control law can be derived [6]:

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

LTI control laws that satisfy desired engineering criteria [9] can be designed for *OP* to provide a set of locally optimal gains and Riccati matrices $\{\mathbf{C}, \mathbf{P}\}_\kappa$. The gradient of the action network at the κ^{th} operating point, which has value in initializing the network, is found by differentiating eq. 4 with respect to $\mathbf{x}^*(t)$. Using the result in eq. 12:

$$\left.\frac{\partial\mathbf{u}^*\left[\mathbf{x}^*(t)\right]}{\partial\mathbf{x}^*(t)}\right|_{\mathbf{x}_0^*,\,\mathbf{a}_\kappa} = \left.\frac{\partial\Delta\mathbf{u}^*(t)}{\partial\Delta\mathbf{x}^*(t)}\right|_{\Delta\mathbf{x}^*=0,\,\mathbf{a}_\kappa} = -\mathbf{C}_\kappa \qquad (13)$$

$\mathbf{C}_\kappa$ is known from the LQ optimal gain matrices, and $\mathbf{a}_\kappa$ is the scheduling vector evaluated at the κ^{th} operating conditions. 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 $\lambda^*[\mathbf{x}^*(t)]$ or simply $\lambda^*(t)$ (eq. 6). The LQ optimal value function, eq. 11, can be differentiated twice with respect to the state to seek the following derivative,

$$\left.\frac{\partial\lambda^*\left[\mathbf{x}^*(t)\right]}{\partial\mathbf{x}^*(t)}\right|_{\mathbf{x}_0^*,\,\mathbf{a}_\kappa} = \left.\frac{\partial^2 V^*\left[\Delta\mathbf{x}^*(t)\right]}{\partial\Delta\mathbf{x}^*(t)^2}\right|_{\Delta\mathbf{x}^*=0,\,\mathbf{a}_\kappa} = \mathbf{P}_\kappa \qquad (14)$$

where, $\mathbf{P}_\kappa$ is known and is used to initialize the critic.

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

$$\left.\mathbf{z}[\mathbf{x}(t), \mathbf{a}(t)]\right|_{\mathbf{x}_0^*,\,\mathbf{a}_\kappa} = \mathbf{0} \qquad (15)$$

The network architecture, number of nodes, and parameters that match these requirements exactly are determined in one step by solving sets of linear algebraic equations.

2.3. Dual Heuristic Programming Adaptive Critic

The on-line logic is implemented in discrete time through an incremental optimization scheme based on dual heuristic dynamic programming. During each time interval $\Delta t = t_{k+1} - t_k$, the action and critic networks are adapted to more closely approximate the optimal control law and value function derivatives, respectively. Adaptation criteria are derived from the recurrence relation by discretizing the optimal control problem [1]. Howard's form of the recurrence relation [3] can be used to approximate the value function over time,

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

where $V[\mathbf{x}(t_{k+1})]$ is necessarily a predicted value. The control $\mathbf{u}(t_k)$ is defined as the function of $\mathbf{x}(t_k)$ that minimizes the right-hand side of eq. 16. When the function $V[\mathbf{x}(t_k)]$ is calculated from eq. 16 based on the current control, and $\mathbf{u}(t_k)$ is adjusted to minimize this optimal value function approximation, the method iteratively converges to an optimal strategy [3]. For simplicity, the asterisks will be omitted in the remainder of the paper.

At time t_k, the control strategy for which the value function is stationary satisfies the optimality condition:

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

Equation 16 is differentiated with respect to the state to obtain a recurrence relation for the DHP critic, which approximates the functional $\lambda[\mathbf{x}(t)]$:

$$\begin{aligned}\lambda[\mathbf{x}(t_k)] \equiv \frac{\partial V[\mathbf{x}(t_k)]}{\partial\mathbf{x}(t_k)} &= \frac{\partial L[\mathbf{x}(t_k), \mathbf{u}(t_k)]}{\partial\mathbf{x}(t_k)} + \\ &\frac{\partial L[\mathbf{x}(t_k), \mathbf{u}(t_k)]}{\partial\mathbf{u}(t_k)}\frac{\partial\mathbf{u}[\mathbf{x}(t_k)]}{\partial\mathbf{x}(t_k)} + \lambda[\mathbf{x}(t_{k+1})]\frac{\partial\mathbf{x}(t_{k+1})}{\partial\mathbf{x}(t_k)} + \\ &\lambda[\mathbf{x}(t_{k+1})]\frac{\partial\mathbf{x}(t_{k+1})}{\partial\mathbf{u}(t_k)}\frac{\partial\mathbf{u}[\mathbf{x}(t_k)]}{\partial\mathbf{x}(t_k)}\end{aligned} \qquad (18)$$

The critic is then used to compute $\lambda[\mathbf{x}(t_{k+1})]$ in eq. 17, once the prediction of the state, $\mathbf{x}(t_{k+1})$, is known from the model of the plant (eq. 2).

3. On-line Phase Implementation

The DHP criteria are implemented for the networks' adaptation, based on $\mathbf{x}(t_k)$, as shown by the schematic in Figs. 3 and 4. The adjustable parameters (or weights) of each network are updated to minimize the mean-squared error between a desired output or target and the network's actual output, $\mathbf{z}[\mathbf{p}(t_k)]$, for the input $\mathbf{p}(t_k)$. Equations 17 and 18 are used to generate the action and the critic desired outputs corresponding to $\mathbf{p}(t_k)$, $\mathbf{u}_D(t_k)$ and $\lambda_D(t_k)$, respectively. During the first time interval $(t_1 - t_0)$, the initialization weights are used prior to the network update. Later, the weights obtained during $(t_k - t_{k-1})$ are used as prior weights for the interval $(t_{k+1} - t_k)$.

3.1. Action and Critic Network Target Generation

The action network target, $\mathbf{u}_D(t_k)$, is obtained by solving the optimality condition, eq. 17, which consists of a set of nonlinear equations. A guess to the solution, $\mathbf{u}_D(t_k)^G$, initially is provided by the action network (using the prior weights). Subsequently, it is perturbed by an established algorithm (e.g., Newton-Raphson) until the stopping condition is met. $\lambda(t_{k+1})$ is computed by the critic network based on the prediction of $\mathbf{x}(t_{k+1})$, as shown in Fig. 3. Once the action network has been updated, the critic's desired output is computed from eq. 18 based on the exact values of $\mathbf{u}(t_k)$ and $\partial\mathbf{u}(t_k)/\partial\mathbf{x}(t_k)$. In the case of the critic (Fig. 4), no iteration is needed to compute its target $\lambda_D(t_k)$, which is at best a prediction of $\lambda(t_k)$. The derivatives $\partial L[\bullet]/\partial\mathbf{x}(t_k)$ and $\partial L[\bullet]/\partial\mathbf{u}(t_k)$ are computed analytically from $L[\mathbf{x}(t_k),\mathbf{u}(t_k)]$. The transition matrices, $\partial\mathbf{x}(t_{k+1})/\partial\mathbf{u}(t_k)$ and $\partial\mathbf{x}(t_{k+1})/\partial\mathbf{x}(t_k)$, are obtained numerically from eq. 2.

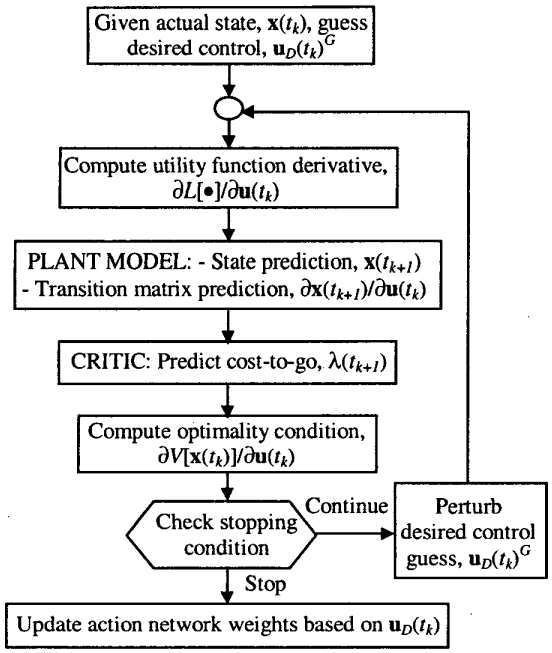

Figure 3. Action network adaptation, during $\Delta t = t_{k+1} - t_k$.

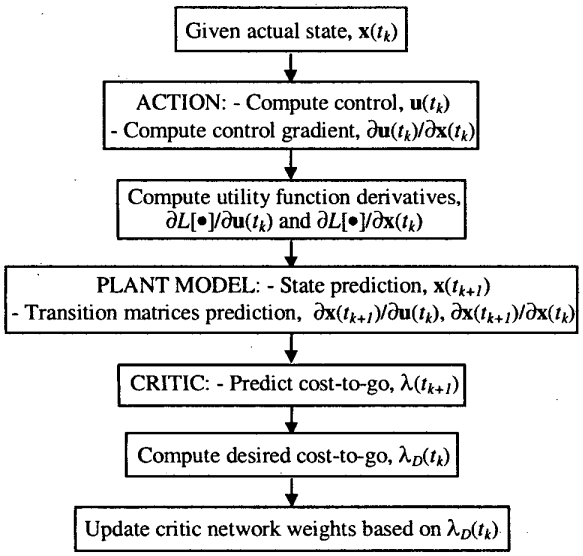

Figure 4. Critic network adaptation, during $\Delta t = t_{k+1} - t_k$.

3.2. On-line Training Algorithm

The on-line training algorithm minimizes an error function E, defined in terms of one desired output $\mathbf{z}_D$ and the actual network output $\mathbf{z}$, with respect to $\mathbf{w}$, a vector of ordered weights w_ℓ indexed by $\ell = 1, 2, \ldots$:

$$E(\mathbf{w}) \equiv \frac{1}{2}\left\|\mathbf{z}_D - \mathbf{z}(\mathbf{w})\right\|^2 \tag{19}$$

Since the initialized weights are close to being optimal, the on-line minimization is kept local. Based on the idea of backpropagation learning [11], at each epoch, i, the on-line training algorithm modifies each weight $w_\ell^{(i)}$ by an increment $\Delta w_\ell^{(i)}$, based on the derivative $\partial E(\mathbf{w})/\partial w_\ell$, i.e.:

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

The on-line phase is effective and reliable when the error begins decreasing at the onset of training, and the update algorithm does not degrade prior network weights. Because of the high-dimensional nature of real-world applications, the neural networks implemented typically are large and their parameters differ by many orders of magnitude, causing the derivatives to be highly dissimilar. Speed and memory requirements are particularly stringent on line, rendering this training phase arduous in practice.

For these reasons, the resilient backpropagation algorithm (RPROP) [12] is modified and implemented. RPROP is based only on the temporal behavior of the signs of the gradients [12]. Therefore, it has low memory requirements and no dependence on the size of the derivatives. The individual size of each increment, denoted by Δ_ℓ, is increased by a factor η^+ when the derivative is not changing sign, while it is decreased by a factor η^- when the derivative is changing sign. This process accelerates convergence in shallow regions and slows the search down 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 previous value $w_\ell^{(i-1)}$ by a *backtracking* epoch [12].

Backtracking is a key algorithmic feature that allows the search to remain local. Another crucial element is the initial increment value $\Delta_\ell^{(0)}$. Setting all initial increments equal to the same constant value (e.g., 0.1) for weights of dissimilar sizes [12] is equivalent to disregarding prior network weights. Instead, initial increments are chosen commensurate with a fraction, f_w, of the corresponding prior weights and perturbed by f_0 to account for zero weights:

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

The same weight update routine is used for the action and the critic networks by letting $\mathbf{z}_D = \mathbf{u}_D(t_k)$ in the action update, and $\mathbf{z}_D = \lambda_D(t_k)$ in the critic update.

4. Adaptive Critic Proportional Integral Neural Network Control Design

The neural controller structure is motivated by a multivariable linear controller; proportional-integral (PI) control is considered for illustration. A PI controller modifies the stability and transient response of the plant through the feedback gain matrix, $\mathbf{C}_B$, and provides Type-1 response to command inputs through the proportional gain matrix, $\mathbf{C}_F$, and the command-integral gain matrix $\mathbf{C}_I$ [10]. These gains are computed by minimizing a cost function in the form of eq. 9 formulated in terms of the augmented state

$\mathbf{x}_a$ and the control deviation $\tilde{\mathbf{u}}$. $\mathbf{x}_a$ includes the state deviation $\tilde{\mathbf{x}}$ and the output error's time integral ξ, i.e., $\mathbf{x}_a \equiv [\tilde{\mathbf{x}}^T \ \xi^T]^T$, where $\tilde{\mathbf{x}} \equiv \mathbf{x} - \mathbf{x}_c$. $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{y}}$ are similarly defined. The set point ($\mathbf{x}_c$, $\mathbf{u}_c$) is a function of the command input, $\mathbf{y}_c$, [6]. The LQ law (eq. 12) provides for the optimal control in terms of the newly defined deviations:

$$\tilde{\mathbf{u}}(t) = -\mathbf{C}_a \mathbf{x}_a(t) = -\mathbf{C}_B \tilde{\mathbf{x}}(t) - \mathbf{C}_I \xi(t) \tag{22}$$

The gains and the Riccati matrix $\mathbf{P}_a$ are obtained by solving a matrix Riccati equation [6] formulated in terms of $\mathbf{x}_a$ and $\tilde{\mathbf{u}}$. The weighting matrices **Q**, **M**, and **R**, are designed using implicit model following [10].

The corresponding neural network structure is obtained by replacing each linear gain matrix with a nonlinear control network, $\mathbf{NN}_B$ for $\mathbf{C}_B$, $\mathbf{NN}_F$ for $\mathbf{C}_F$, and $\mathbf{NN}_I$ for $\mathbf{C}_I$ [10]. In addition to the scheduling vector, the networks $\mathbf{NN}_B$, $\mathbf{NN}_F$, and $\mathbf{NN}_I$ are provided with the state deviation, the command input, and the command error integral, respectively. Each network contributes to the total control,

$$\begin{aligned} \mathbf{u}(t) &= \mathbf{u}_c(t) + \Delta\mathbf{u}_B(t) + \Delta\mathbf{u}_I(t) \\ &= \mathbf{NN}_F[\mathbf{y}_c(t), \mathbf{a}(t)] + \mathbf{NN}_B[\tilde{\mathbf{x}}(t), \mathbf{a}(t)] + \mathbf{NN}_I[\xi(t), \mathbf{a}(t)] \end{aligned} \tag{23}$$

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

The action network, $\mathbf{NN}_A$, that approximates the minimizing control law consists of the algebraic sum of $\mathbf{NN}_B$ and $\mathbf{NN}_I$:

$$\tilde{\mathbf{u}}(t) = \mathbf{NN}_A[\mathbf{x}_a(t), \mathbf{a}(t)] \tag{24}$$

Given the same inputs, the critic network, $\mathbf{NN}_C$, computes the derivative of the value function $V[\mathbf{x}_a(t)]$ with respect to the augmented state:

$$\lambda_a(t) \equiv \frac{\partial V[\mathbf{x}_a(t)]}{\partial \mathbf{x}_a(t)} = \mathbf{NN}_C(\mathbf{x}_a(t), \mathbf{a}(t)) \tag{25}$$

The final neural controller structure is shown in Fig. 5. The *Scheduling Variable Generator* (*SVG*) produces **a** based on $\mathbf{y}_c$ and an exogenous vector, **e**, of measured variables. The *Command State Generator* (*CSG*) provides secondary elements of the state that are compatible with $\mathbf{y}_c$.

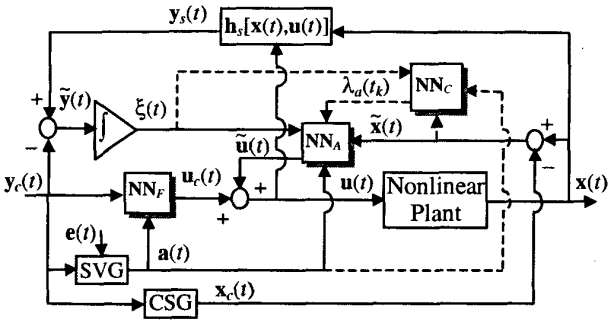

Figure 5. Action critic neural network controller.

Subsequently, eq. 17 and 18 formulated in terms of $\mathbf{x}_a$ and $\tilde{\mathbf{u}}$ are used on-line to adapt the action and the critic network, respectively.

5. Flight Control Simulation and Results

The adaptive controller is implemented on a six-degree-of-freedom business jet aircraft model. The simulation explores the full flight envelope, $OR = \{V, H, \gamma, \mu, \beta\}$. The control design is based on the state, $\mathbf{x} = [V\ \gamma\ q\ \theta\ r\ \beta\ p\ \mu]^T$, comprising airspeed V (m/s), path angle γ (rad), pitch rate q (rad), pitch angle θ (rad), yaw rate r (rad/s), sideslip angle β (rad), roll rate p (rad/s), and bank angle μ (rad). The independent controls being generated are throttle δT (%), stabilator δS (rad), aileron δA (rad), and rudder δR (rad); i.e., $\mathbf{u} = [\delta T\ \delta S\ \delta A\ \delta R]^T$. The command, $\mathbf{y}_c = [V_c\ \gamma_c\ \mu_c\ \beta_c]^T$, contains the state elements that, given the altitude H (m), uniquely specify a longitudinal-lateral-directional steady maneuver, postulating $\dot{\phi}_c = \dot{\theta}_c = 0$ with ϕ as the Euler roll angle. All angular and kinematic relations involved pertain to non-longitudinal, non-level flight, and are based on spherical trigonometry [13].

The neural control architecture specified in Section 4 is initialized based on the performance criteria established locally by the linear gains and the augmented Riccati matrix, with $\mathbf{NN}_F$ approximating the aircraft trim map [14]. Independent longitudinal and lateral-directional linear models are obtained for a set, *OP*, of thirty-four operating points chosen from $\{V, H\} \subset OR$, letting $\gamma_0 = \mu_0 = \beta_0 = 0$. Corresponding linear controllers are designed and used independently to initialize longitudinal and lateral-directional control networks [10]. Each initialized pair is algebraically joined into a full longitudinal-lateral-directional neural network. Then, the full feedback and command-integral networks, $\mathbf{NN}_B$ and $\mathbf{NN}_I$, are algebraically summed to form the action network, $\mathbf{NN}_A$.

During every time interval (0.1 sec), the critic and action networks are updated by the modified RPROP algorithm of Section 3.2 based on the respective targets, $\tilde{\mathbf{u}}_D$ and $(\lambda_a)_D$ (Figs. 3 and 4). The user-defined update parameters are: $\eta^+ = 1.2$, $\eta^- = 0.5$ [12], $f_w \sim O(10^{-5})$, and $f_0 << 1$. The modified RPROP algorithm (Section 3.2) is validated by comparing its performance to that of the *MATLAB 5.3* "trainrp" learning function, for the update of the action network at $t = 0.2$ sec, as illustrated in Fig. 6. Further studies also show that the modified RPROP better preserves the original weights and avoids overfitting.

During the on-line phase, the update algorithm terminates after the mean-squared measure of each network error $[\mathbf{z}_D - \mathbf{z}(\mathbf{w})]$ has decreased by 10% and at least 3 epochs have elapsed. When more than 3 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 $\forall\ \ell$. This typically requires several epochs during the first 0.2 sec to adjust the increment size, and three epochs during later time intervals to decrease the

network error. The initialization phase and the modified RPROP algorithm render the on-line phase feasible and efficient with respect to both time and storage consumption.

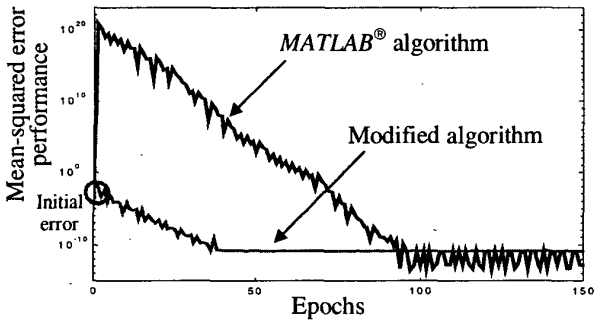

Figure 6. Comparison between the *MATLAB*® RPROP algorithm and its modified version (NN_A, at $t = 0.2$ s).

Histories of the state elements directly commanded by $\mathbf{y}_c$ are used to evaluate performance during a large-angle asymmetric maneuver and are plotted with a solid line in Fig. 7. At the initial time, the aircraft is flying level at a nominal airspeed V_0 of 95 m/s and an altitude H_0 of 2, 000 m, with $(V_0, H_0) \not\subset OP$. The state response is judged against an equivalent PI neural network controller (represented by a dashed line in Fig. 7) that is initialized with the same linear data but does not undergo on-line adaptation. The comparison shows that on-line adaptation brings about an improvement with respect to the linear controllers. The action and critic networks minimize the cost-to-go on line, in the presence of coupling and nonlinear effects unaccounted for by the linear designs, and they do so without unlearning previous information.

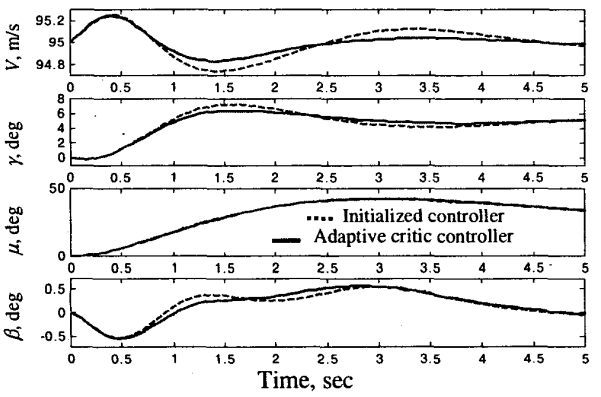

Figure 7. Comparison between on-line adaptive controller and initialized controller at $(V_0, H_0) = (95$ m/s, 2 Km), subject to 5-deg climb angle and 30-deg roll step command.

6. Conclusions

Advances in off-line and on-line learning techniques and in adaptive critic methods are presented and incorporated in a novel approach to neural control system design. The nonlinear control system is taught using a two-phase learning procedure encompassing an initialization phase that provides for reliability, and an on-line phase that accounts for actual plant dynamics. Both phases are founded on optimal control theory and are realized with significant computational savings. The nonlinear adaptive controller is successfully implemented for the command-input control of a full-scale aircraft simulation, with the neurocontrollers adjusting on line, while retaining their baseline performance. The adaptive controllers spontaneously utilize those parameters that were unused during initialization to learn newly available information. Also, a modified resilient backpropagation algorithm allows the networks to improve their performance in only one or few epochs over each time increment. The advancements of all key design stages combined bring about concrete potential for real-life applications.

Acknowledgement

This research has been supported by the Federal Aviation Administration and the National Aeronautics and Space Administration under FAA Grant No. 95-G-0011.

References

[1] Kirk, D. E., *Optimal Control Theory; and Introduction*, Prentice-Hall, Englewood Cliffs, NJ, 1970.

[2] Bellman, R. E., *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

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

[4] White, D. A, Sofge, D., *Handbook of Intelligent Control*, Van Nostrand Reinhold, New York, 1992.

[5] Narendra, K. S., Parthasaranthy, K., "Identification and control of dynamical systems using neural networks", *IEEE Trans. Neural Networks*, Vol. 1, 1990, pp. 4-27.

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

[7] 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.

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

[9] Stengel, R. F., Marrison, C., "Design of Robust Control Systems for Hypersonic Aircraft," *J. Guidance, Control, and Dynamics*, Vol. 21, No.1, 1997, pp.58-63.

[10] Ferrari, S., Stengel, R. F., "Classical/Neural Synthesis of Nonlinear Control Systems," to appear in *J. Guidance, Control and Dynamics*.

[11] Werbos, P. J., "Backpropagation Through Time: What It Does and How to Do It," *Proc. IEEE*, Vol. 78, 1990, pp. 1550-1560.

[12] Reidmiller, M., Braun, H., "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," *Proc. IEEE Int. Conf. on NN (ICNN)*, San Francisco, CA, 1993.

[13] Kalviste, J., "Spherical Mapping and Analysis of Aircraft Angles for Maneuvering Flight," *J. Aircraft*, Vol. 24, No. 8, 1987, pp. 523-530.

[14] Ferrari, S., Stengel, R. F. "Algebraic Training of a Neural Network," *Proc. American Control Conference*, Arlington, VA, 2001.