Helicopter Trimming and Tracking Control Using Direct Neural Dynamic Programming

Russell Enns and Jennie Si

***Abstract*—This paper advances a neural-network-based approximate dynamic programming control mechanism that can be applied to complex control problems such as helicopter flight control design. Based on direct neural dynamic programming (DNDP), an approximate dynamic programming methodology, the control system is tailored to learn to maneuver a helicopter. The paper consists of a comprehensive treatise of this DNDP-based tracking control framework and extensive simulation studies for an Apache helicopter. A trim network is developed and seamlessly integrated into the neural dynamic programming (NDP) controller as part of a baseline structure for controlling complex nonlinear systems such as a helicopter. Design robustness is addressed by performing simulations under various disturbance conditions. All designs are tested using FLYRT, a sophisticated industrial scale nonlinear validated model of the Apache helicopter. This is probably the first time that an approximate dynamic programming methodology has been systematically applied to, and evaluated on, a complex, continuous state, multiple-input–multiple-output nonlinear system with uncertainty. Though illustrated for helicopters, the DNDP control system framework should be applicable to general purpose tracking control.**

***Index Terms*—Approximate dynamic programming, helicopter flight control, helicopter trim, neural dynamic programming.**

I. Introduction

THIS paper focuses on the application of direct neural dynamic programming and demonstrates how it can be used to control complex, realistic, and higher dimensional systems. The direct neural dynamic programming (DNDP) is to be used to control a helicopter to perform realistic maneuvers and the paper demonstrates how this method provides an approximate solution to this optimal control problem that is often solved by dynamic programming, and, in doing so avoiding the curse of dimensionality. The DNDP mechanism was first introduced in [8] and later in [9], which provided basic design principles along with a comprehensive evaluation of helicopter stabilization. In this paper, we expand on the original DNDP mechanism to provide the ability to do command tracking for complex systems, again demonstrating the control mechanism on a helicopter.

The approach taken in the present paper resides in the area of "approximate dynamic programming." This is an interdisciplinary area that has been actively researched but may not be known to all relevant disciplines. Different terminologies have been given to this area, such as reinforcement learning [1], neuro-dynamic programming [2], adaptive critics [3], and so forth. Recently and most often, it has been referred to as approximate dynamic programming (ADP) [4]. This paper is not in a position to discuss which name fits the field the most. Rather, we consider techniques that converge to an (approximately) optimal policy over time in a nonlinear stochastic decision and control problem. Particularly, in this paper, we show that a recently proposed learning control framework [8], still under the theme of neural network, can solve very complex problems such as tracking of Apache helicopter.

For the ease of discussion, the terms "discrete-event" approaches and "continuous-state" approaches are used to discuss solutions of ADP. The former refers to the fact that controls/actions are obtained by search algorithms and the problems are discrete event in nature. The latter refers to the fact that (approximate) gradient information is used in value function approximation and action generation, and the problems can be in both continuous or discrete-state spaces.

Until very recently [5], generalization problems remain a major hurdle in reinforcement learning community when dealing with continuous or large discrete-state spaces or action spaces. In discrete environments, there is a guarantee that any operation that updates the value function (according to the Bellman equation) can only reduce the error between the current value function and the optimal value function. However, this guarantee does not generalize well when the size of the discrete-states are large. Using some simple examples, it was shown in [6] that the value function errors may grow arbitrarily large by using value iteration. Besides, the problems tackled by some state-of-the-art algorithms [5]–[7] are still artificial and small. They are also lacking in a systematic evaluation of system performance. A car climbing hill example was used in [6]. The sparse coarse coding concept was proposed in [7] and implemented using CMAC, but only demonstrated on a small Acrobot. The most recent work in [5] makes use of variable resolution discretization. It demonstrated the various approaches to splitting on the familiar, nonlinear, nonminimum phase, and two-dimensional (2-D) problem of the "car on the hill." It then evaluates the performance of a variety of splitting criteria on many small benchmark problems, paying careful attention to their number-of-cells versus closeness-to-optimality tradeoff curves. There is no clear evidence of how this approach may generalize.

In this paper, we show that our DNDP design [8] is promising as a robust algorithm (measured by learning statistics, problem scalability, the range of problems handled) to the ADP problems. We present a comprehensive case study of this design in

Manuscript received June 19, 2001; revised July 24, 2002. This work was supported by the National Science Foundation under Grants ECS-9553202 and ECS-0002098.

The authors are with the Department of Electrical Engineering, Arizona State University, Tempe, AZ 85287-7606 USA.

Digital Object Identifier 10.1109/TNN.2003.813839

1045-9227/03$17.00 © 2003 IEEE

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

tracking an Apache helicopter using a full-scale industrial helicopter design model. This is probably the first time that ADP has been systematically applied to and evaluated on a complex continuous state multi-input–multi-output (MIMO) nonlinear system with uncertainty.

DNDP is an ADP method. More specifically, it is an approximate neural dynamic programming (NDP) method that does not require explicitly building a system model prior to learning to improve system performance. DNDP was perceived as a strong candidate for a learning system for helicopter flight control because, in addition to its ability of online learning, it can be applied to complex systems such as helicopters without the need to decouple the control system into simpler subsystems. As such, it can learn to take advantage of any of the system cross coupling characteristics when generating its control solution, including coupling benefits that may not be apparent to a control systems design engineer. NDP methods can deal with both explicitly and implicitly defined system performance measures which are usually a function of the system states and control actions. NDP methods avoid the "curse of dimensionality" that dynamic programming methods suffer from by providing approximate solutions. This, however, may also be considered as the down side of the NDP when true "optimality" is demanded.

From a flight control perspective, there have been numerous control methodologies successfully applied to many flight control problems. In fact, there have already been many examples of neural networks in flight controls. Much of the neural-network research has either been limited to simulation studies of simple (usually scalar) control subsystems [10], [11] or has decoupled sophisticated systems into smaller subsystems guided by the designer's expertise [12]–[14]. Further, most of these papers use neural networks to either approximate or improve on the approximation of an aircraft inverse dynamics. Calise *et al.* have contributed a large body of work that uses neural networks to improve on an underlying dynamic model inversion control methodology [12]–[14]. The neural networks compensate for any model inversion error that exists by augmenting a control adjustment to the nominal proportional-derivative control term.

Only a handful of research has been done in the area of reinforcement learning for flight control. Ha [15] uses neural networks as a direct form of control, though the study is limited to lateral-directional control for a linear model. Balakrishnan is one of the first to use a form of reinforcement learning (adaptive critic-based networks) for aircraft flight controls [16]. However, the research limits itself to the longitudinal axis and as a result the system only has a single control. Prokhorov *et al.* [17] have demonstrated their adaptive critic designs (ACDs) in an auto-lander application, which takes in altitude, vertical speed, and horizontal position (three states total), and in some cases pitch and horizontal speed (for five states total), and computes the required pitch command (one output total). Thus, their demonstration is limited to scalar control. Their system was tested on a linearized 2-D model of a commercial aircraft. Note that the above reinforcement learning papers take a model-based approach, as opposed to the model free approach that can be used by DNDP.

The purpose of this paper is not to compare our methods to any of the existing methods, be they reinforcement learning-based, neural-network-based, or otherwise. The purpose is to demonstrate the power of DNDP as an approximate dynamic programming control methodology on a challenging controls problem that other approximate dynamic programming algorithms may not be able to handle. We provide a number of cases to show the ability of the learning control system to perform aircraft maneuvers. Statistical results showing DNDP's ability to learn acceleration maneuvers from hover to 50 ft/s at various accelerations, up to the aircraft upper limits of 0.25 g (8 $\mathrm{ft/s^2}$), are provided. Results are also shown for deceleration maneuvers from 100 to 50 ft/s at various decelerations. Simulations are performed in both clear air and in the presence of turbulence and step gusts using an industrial Dryden model. Plots of the typical and statistical tracking performance are also shown for two representative cases. Unlike many results which are based on linearized models and corresponding assumptions, our DNDP designs and simulations are conducted using the FLYRT model. Thus, we are dealing with a very realistic system with nonlinearities, actuator dynamics, etc.

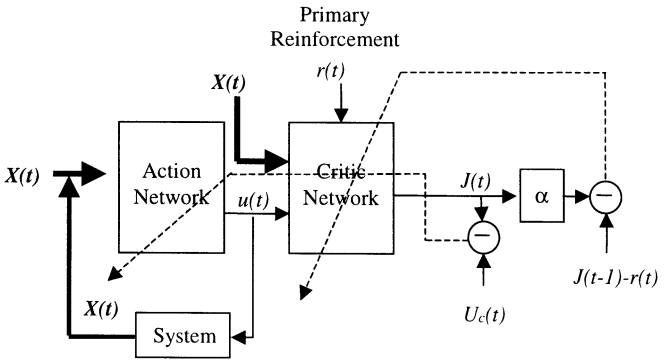

Fig. 1. Schematic diagram for implementation of the DNDP. The solid lines represent signal flow, while the dashed lines are the paths for parameter tuning.

This paper is organized as follows. Section II provides a comprehensive description of the DNDP mechanism. Section III briefly describes the helicopter model used for evaluating the NDP designs. Section IV applies the DNDP methodology developed in Section II to the helicopter flight control tracking problem. Section V focuses on developing a trim network, a critical element required for successful DNDP designs. Section VI defines the design objectives and provides simulation results. Section VII then provides some conclusions.

II. DNDP Mechanism

Fig. 1 is a schematic diagram of the original approximate dynamic programming control scheme [8]. The binary reinforcement signal $r(t)$ is provided from the external environment and, typically, is either a "0" or a "−1" corresponding to "success" or "failure," respectively.

In our on-line learning control design, the controller is "naive" when it just starts to control, namely the action network and the critic network are both randomly initialized in their weights/parameters. Once a system state is observed, an action will be subsequently produced based on the parameters in the action network. A "better" control value under the specific system state will lead to a more balanced equation of the

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

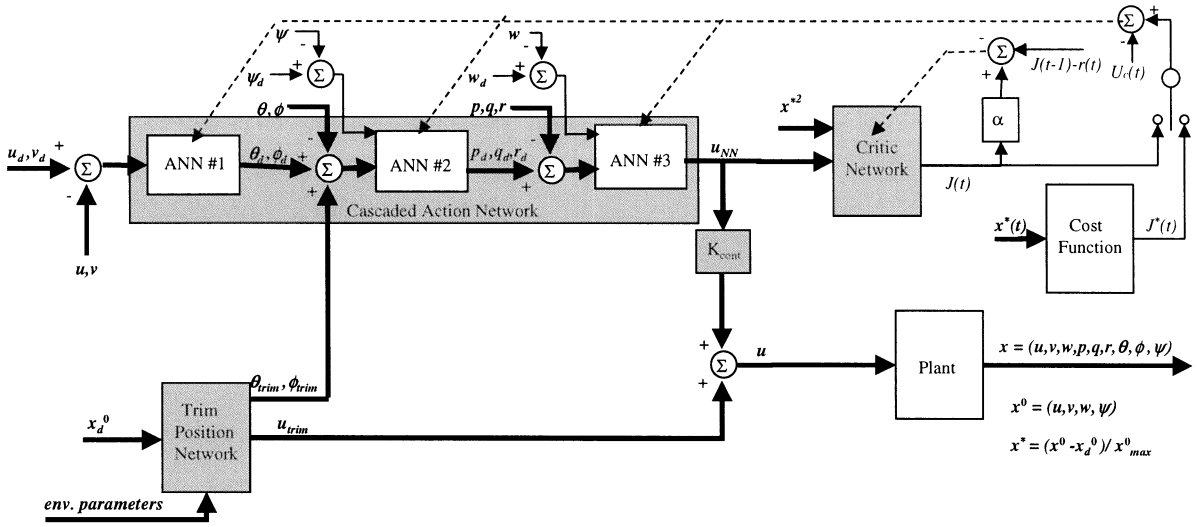

Fig. 2. DNDP-based helicopter controller integrating three cascaded artificial neural networks in one action network, a trim network, and a critic network. Also allows for an explicit cost function to be used.

principle of optimality. This set of system operations will be reinforced through memory or association between states and control output in the action network. Otherwise, the control value will be adjusted through tuning the weights in the action network in order to make the equation of the principle of optimality more balanced.

More specifically, consider the critic network in Fig. 1. The output of the critic element (the J function) approximates the discounted total reward-to-go. Specifically, it approximates $R(t)$ at time t given by

$$R(t) = r(t+1) + \alpha r(t+2) + \cdots \tag{1}$$

where $R(t)$ is the future accumulative reward-to-go value at time t, α is a discount factor for the infinite-horizon problem ($0 < \alpha < 1$), and $r(t)$ is the external reinforcement value at time t.

A. Critic Network

The critic network is trained to approximate the "value function" $J(t)$ by minimizing the objective function, which represents the balance of the principle of optimality

$$E_c(t) = \frac{1}{2} e_c^2(t) \tag{2}$$

where

$$e_c(t) = \alpha J(t) - [J(t-1) - r(t)]. \tag{3}$$

The weights of the critic network are updated according to a gradient-descent algorithm

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \Delta \mathbf{w}_c(t) \tag{4}$$

$$\Delta \mathbf{w}_c(t) = \beta_c(t) \left[-\frac{\partial E_c(t)}{\partial \mathbf{w}_c(t)} \right] = \beta_c(t) \left[-\frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial \mathbf{w}_c(t)} \right] \tag{5}$$

where $\beta_c(t)$ is the learning rate of the critic network at time t, which usually decreases with time to a small value

$$\frac{\partial E_c(t)}{\partial J(t)} = \frac{\partial E_c(t)}{\partial e_c(t)} \frac{\partial e_c(t)}{\partial J(t)} = \alpha e_c(t) \tag{6}$$

and $\partial J(t)/\partial \mathbf{w}_c(t)$ is a function of the critic network structure [8].

B. Action Network

The principle in adapting the action network is to backpropagate the error between the desired ultimate objective, denoted by U_c, and the cost function $R(t)$. Either the actual cost function $R(t)$, or an approximation to it $J(t)$, is used depending on whether an explicit cost function or a critic network is available. In the latter case backpropagation is done through the critic network as shown in Fig. 1 and later Fig. 2. For notational simplicity $J(t)$ represents either the actual or approximate cost function, depending on which is being used.

The weight updating in the action network adjusts the action network weights to minimize the following objective function:

$$E_a(t) = \frac{1}{2} e_a^2(t) \tag{7}$$

where

$$e_a(t) = J(t) - U_c(t). \tag{8}$$

The weights in the action network are then updated similarly to the critic network according to

$$\mathbf{w}_a(t+1) = \mathbf{w}_a(t) + \Delta \mathbf{w}_a(t) \tag{9}$$

$$\Delta \mathbf{w}_a(t) = \beta_a(t) \left[-\frac{\partial E_a(t)}{\partial \mathbf{w}_a(t)} \right] \tag{10}$$

where

$$\frac{\partial E_a(t)}{\partial \mathbf{w}_a(t)} = \frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u(t)} \frac{\partial u(t)}{\partial \mathbf{w}_a(t)} \tag{11}$$

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

where $\partial J(t)/\partial u(t)$ is obtained by backpropagation through the critic network, and $\partial u(t)/\partial \mathbf{w}_a(t)$ is a function of the action network structure [8].

In every time frame the weight equations are updated until either the error has been sufficiently reduced ($E < E_{tol}$) or the internal update cycles (N_{cyc}) of the weights have been reached. The action and the critic networks are updated alternately. Further details of the learning algorithm can be found in [8].

III. Helicopter Model

A helicopter is a sophisticated system with multiple inputs used to control a significant number of states. There exists a large amount of cross coupling between control inputs and states. Further, the system is highly nonlinear and changes significantly as a function of operating condition. For these reasons the helicopter serves as an excellent and challenging platform for testing approximate dynamic programming systems.

The helicopter's states are controlled by a main rotor and a tail rotor. There are three main rotor actuators whose positions, z_A, z_B, and z_C, control the position and orientation of a swash plate which in turn controls the main rotor's blade angles as a function of rotational azimuth. There is a single tail rotor actuator position (z_D) which controls the tail rotor's blade angles. The aircraft states are numerous. For flight control purposes, the rates of interest are limited to the aircraft translational (u, v, w) and rotational (p, q, r) velocities and the aircraft orientation (θ, ϕ, ψ) for a total of nine states. The helicopter's longitudinal (u), lateral (v), and vertical velocities (w) are in feet per second. The helicopter's roll rate (p), pitch rate (q), and yaw rate (r) are in degrees per second. The helicopter's Euler angles, pitch (θ), roll (ϕ), and yaw (ψ) are in degrees. The states can be written in vector form as $\mathbf{x} = [u, v, w, p, q, r, \theta, \phi, \psi]$. The controls can be written in vector form as $\mathbf{u} = [z_A, z_B, z_C, z_D]$.

For simulation purposes, we use a detailed helicopter model run at 50 Hz for evaluating our NDP controller's performance. The model, named FLYRT, is a sophisticated nonlinear flight simulation model of the Apache helicopter developed by Boeing over the past two decades [26].

FLYRT models all the forces and moments acting on the helicopter. The rotor is modeled using a blade element model. FLYRT dynamically couples the six-degrees-of-freedom rigid body of the helicopter to the main rotor through Euler equations. The drive train is represented as a single degree of freedom model and is coupled to the main rotor, tail rotor, and engine. The engine is modeled in sufficient detail to cover performance over all phases of flight, including ground modes. The landing gear is modeled as three independent units interfacing with a rigid airframe. Quaternions are used during state integration to accommodate large attitude maneuvers.

FLYRT also models the mechanical geometry between the actuators and the helicopter blades as well as the dynamics of the actuators. Each actuator is modeled as a first-order lag with time constant $\tau = 0.03$, reflective of a typical actuator. Actuator rate and position limits are also modeled.

The operating conditions for which our simulation studies are performed are shown in Table I. The center of gravity (C.G.) is listed in the standard Apache FS/WL/BL coordinate frame [26].

TABLE I
Helicopter Operating Conditions

Weight	16324 lb
C.G. - FS/BL/WL	201.6 in, 0.2 in, 144.3 in
Temperature	59^o F
Altitude	1770 ft

IV. DNDP Mechanism Applied to Helicopter Control

We now turn our attention toward applying the DNDP framework to the tracking control of an Apache helicopter. The objective is to learn to create appropriate control actions solely by observing the helicopter states, evaluating the controller performance and adjusting the neural networks accordingly.

Fig. 2 outlines the DNDP control structure applied to helicopter tracking control. It consists of a structured cascade of neural networks forming the action network, the critic network, and the trim network. The action network provides the controls required to drive the helicopter to the desired system state. The critic network approximates the cost function if an explicit cost function does not exist. The trim network provides nominal trim control positions as a function of the system desired operating condition.

In this paper, we introduce the concept of a structured cascade of artificial neural networks (ANNs) as the action network. The explicit structure embedded in this ANN, lacking in earlier DNDP designs [8], allows the NDP controller to more easily learn and take advantage of the physical relationships and dependencies of the system. To perform command tracking a vector $\mathbf{x}_d$ as a function of time is to be specified first. For helicopters, it is well established that four of the states in the state vector $\mathbf{x}$ are explicitly controllable. In this experiment, we desire to control the velocities u, v, and w and the aircraft's yaw ψ. The rotational velocities and remaining Euler angles (pitch and roll) will be determined by NDP to achieve the specified tracking goal. We denote this new desired tracking vector $\mathbf{x}_d^0 = [u, v, w, \psi]$: a subset of the original desired state vector $\mathbf{x}_d$.

Such structure in the action network is similar to classic controllers for helicopters, providing for inner loop body rate control, attitude control and outer loop velocity control. In this way, the explicit relationships between body angular rates, attitudes and translational velocities are taken advantage of. The potential advantage of the structured ANN over classic design methodologies is that it permits full cross-axes control coupling that many single-input–single-output (SISO) proportional integral derivative (PID) controller designs do not. However, the structured ANN does introduce a level of human knowledge/expertise to its implementation that is not transparent to nonexperts.

It is possible, but rather cumbersome, to show that a classic proportional controller can be equated to one instance (one set of weights) of our structured ANN if the network nonlinearities are removed (or linearized about the network operating point). Such a relationship between the two designs can be used to provide a good first guess of the action network weights should one want to apply this "expert" knowledge to the learning system.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

However, all results shown in this paper were obtained while the learning system was trained from scratch without using any expert knowledge.

Refer to Fig. 2, the inputs to the first ANN are the longitudinal and lateral velocity errors $u_{\rm err} = u_d - u$ and $v_{\rm err} = v_d - v$, respectively. The first ANN outputs are the resulting desired pitch and roll of the helicopter θ_d and ϕ_d. The inputs to the second ANN are the errors in the aircraft attitudes $\theta_{\rm err} = \theta_d + \theta_{\rm trim} - \theta$, $\phi_{\rm err} = \phi_d + \phi_{\rm trim} - \phi$, and $\psi_{\rm err} = \psi_d - \psi$. The second ANN outputs are the desired roll, pitch, and yaw rates (p_d, q_d, r_d) of the helicopter, as a function of the attitude errors. These are then summed with the actual angular rates to obtain the angular rate errors, $p_{err} = p_d - p$, $q_{\rm err} = q_d - q$ and $r_{err} = r_d - r$. The angular rate errors and the vertical velocity error $w_{\rm err} = w_d - w$ then form the inputs to the third ANN. The third ANN then computes the controls $\mathbf{u}_{\rm NN} = [u_{{\rm NN},1}, u_{{\rm NN},2}, u_{{\rm NN},3}, u_{{\rm NN},4}]$ as a function of the angular rate and vertical velocity errors. The resulting $\mathbf{u}_{\rm NN}$, which is normalized because of the ANN structure, is then scaled by the controller scaling gain $K_{\rm cont}$ and summed with the nominal trim control from the trim network. That is, $\mathbf{u} = K_{\rm cont}\mathbf{u}_{NN} + \mathbf{u}_{\rm trim}$.

As described in Section IV, the objective of the NDP controller is to create a series of control actions to optimize a discounted total reward-to-go, which is rewritten as follows:

$$R(t) = r(t+1) + \alpha r(t+2) + \alpha^2 r(t+3) + \cdots. \qquad (12)$$

Typically, NDP has been applied to systems where explicit feedback, or instantaneous cost evaluation, is not available at each time step. In such cases, the reinforcement signal $r(t)$ takes a simple binary form with $r(t) = 0$ when the final event is successful (an objective is met), or $r(t) = -1$ if the final event is a failure (the objective is not met). For many real-world tracking problems however, explicit feedback is continually available and so we can define a more informative quadratic reinforcement signal

$$r(t) = -\sum_{i=1}^{n} \left(\frac{(x_i^0 - x_{d,i}^0)}{x_{{\rm max},i}^0} \right)^2 \qquad (13)$$

where, in our application, $n = 4$, x_i^0 is the ith state variable of $x^0 = [u, v, w, \psi]$, $x_d^0 = [u_d, v_d, w_d, \psi_d]$ is the desired state, and $x_{{\rm max},i}^0$ is a normalization term.

The critic and action networks are then trained per Section IV. The equations governing the training of the action network are significantly more complex than prior work since backpropagation must be performed through ANNs 2 and 3 for the training of ANN 1. In essence, it is equivalent to training a six-layer network if we assume that each ANN has two layers of weights.

The objective is to train the controller to perform the specified maneuver regardless of the operating conditions or the vehicle initial conditions. The states of interest are the aircraft translational (u, v, w) and rotational (p, q, r) velocities and the aircraft Euler angles pitch (θ), roll (ϕ), and yaw (ψ). Failure criteria are used to bound allowed error for each state. The allowed errors, shown in Table II, are initially large and decrease as a function of time to an acceptable minimum.

TABLE II
FAILURE CRITERION FOR HELICOPTER STABILIZATION

Aircraft State	Initial Allowed Error	Final Allowed Error	Error Rate
u, v, w	$20ft/s$	$4ft/s$	$-0.8(ft/s)/s$
p, q, r	$30^o/s$	$6^o/s$	$-1.2^o/s/s$
θ, ϕ, ψ	30^o	6^o	$-1.2^o/s$

These failure criteria were chosen judiciously but no claims are made to their optimality. The results show that these criteria create a control system that can control the helicopter both in nominal conditions and when subjected to disturbances. Heuristic failure criteria is one of the advantages of NDP if one does not have an accurate account of the performance measure. This is also one characteristic of the NDP design that differs from other neural control designs. The critic network plays the role of working out a more precise account of the performance measure for credit/blame assignment derived from the heuristic criteria. If the networks have converged, an explicitly desired state has been achieved which is reflected in the U_c term in the NDP structure.

The trim network is a neural network, or lookup table, that is trained, or programmed, to schedule the aircraft's nominal actuator position and aircraft orientation as a function of operating conditions. It is a critical element in nonlinear flight control system design. In the following, we show what trim is and how to integrate this trim design seamlessly into the DNDP control design.

V. TRIMMING THE HELICOPTER

Controlling the helicopter is a nonlinear control system design problem. An important part of practical helicopter control is the ability to determine the trim states for the helicopter over all flight conditions. For nonlinear systems, the trim states are those that lead to steady-state flight conditions. More specifically, they are the positions of the controls and the dependent states associated with achieving a desired steady-state condition. For example, with an aircraft, one tries to adjust (trim) the controls to balance the aerodynamic, inertial, and gravitational forces and moments in all axes at all times. The aircraft is trimmed when the desired balance is achieved or the aircraft enters a desired steady state. In the case of a helicopter, the controls to be trimmed are the four actuators and the dependent states to be trimmed are the pitch, roll, and yaw, i.e., seven trim state variables in total. These states are trimmed for the desired specified steady-state translational velocities (u, v, w) and angular rates (p, q, r), i.e., six variables representing desired steady states. When flying a conventional mechanically controlled helicopter, a pilot continually trims the helicopter via his closed-loop control. Similarly, traditional PID-based control techniques inherently trim the helicopter, the integrators serve as the trim component.

The concept of trim can also be applied to physical systems other than aircraft. In such a case, one tries to balance the system

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

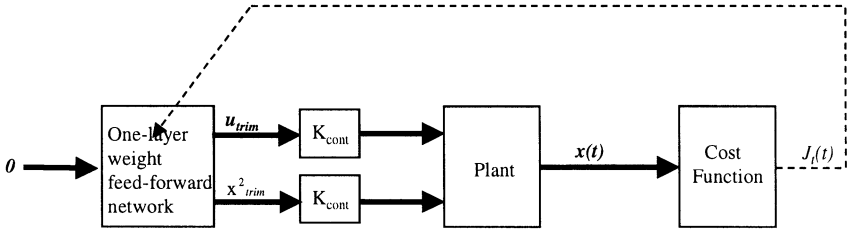

Fig. 3. Neural-network structure for determining trim. The objective is to mininimize the specified cost function $J_t(t)$.

so that the change in state derivative (e.g., acceleration) is either zero or minimized.

The ability to use NDP to trim the helicopter is paramount to successfully applying DNDP to the tracking control problem. Previous NDP control designs were successful because the systems that were tested (e.g., the inverted pendulum) had a zero trim requirement [8]. Similarly, many flight control papers have assumed linear models, in which case, there is also a zero trim requirement since the model is linearized about a trim condition. However, in general trim requirements cannot be ignored for controllers of nonlinear systems, including those for aircraft.

Trim capability is also useful for facilitating realistic simulations. Often one wants to start a simulation at an arbitrary initial condition rather than performing the task of flying the helicopter to that particular state (which in certain cases may be hard to do precisely). Existing trim methods such as that used by FLYRT are limited to performing trim for only a limited set of initial conditions. Thus, a method of trimming the helicopter at an arbitrary specified initial condition is desired.

This paper develops a neural-network-based method for trimming the helicopter that addresses the above issues. The resulting trim network schedules the nominal control trim position as a function of aircraft state and environmental/flight parameters (such as aircraft weight, air density, etc.). This trim method, though applied to helicopters here, can be applied to any general physical control system.

A. Existing Trim Methods

Several techniques have been developed to trim helicopters [18], [19]. Two methods that are commonly used are shooting methods, typically using (damped) Newton's method, and finite element methods [20], [21]. Both are iterative numerical techniques that can be implemented in either a serial or parallel scheme. In the serial scheme, during each iteration the control input trim positions are updated first and then the estimates of the trimmed initial conditions are updated. In the parallel scheme, both control input and initial trim values are updated simultaneously. The claim is that the parallel scheme generally finds the solutions faster but with a higher risk of solution divergence. A third trim method is the auto-pilot trim procedure [22], [23]. Often, this method is augmented with one of the two other methods [24]. A comprehensive review of trimming issues, theory and techniques is provided in [25].

The major difference between existing papers and this research is that the existing papers describe how to trim the helicopter from an aeromechanical viewpoint rather than a control systems viewpoint. That is, the states that are trimmed in these papers (blade flapping angles, lag angles, and controls) are different than what we need to trim (body orientation and controls). Further, most of the existing work assumes knowledge of the equations governing the system (i.e., model-based approaches). In our approach, no such assumption is made. Instead a strictly numerical approach is taken. Further research may be warranted to determine if the proposed approach is also applicable to the aero-mechanical trim problem.

B. Neural-Network-Based Trim Method

This paper proposes a neural-network approach for trimming a helicopter. The method uses the existing DNDP neural-network framework and, hence, can be seamlessly integrated with the NDP controller for a very simple and efficient implementation.

Trimming requires determining the seven trim state variables to include four control positions and three body angles (later defined to be $\mathbf{x^2}$) for a given flight condition. For this reason, we divide the original state vector $\mathbf{x}$ into two parts $\mathbf{x^1} = [u, v, w, p, q, r]$ and $\mathbf{x^2} = [\theta, \phi, \psi]$, where $\mathbf{x^1}$ represents the desired steady state variables for which we want to determine the seven trim positions.

Fig. 3 shows the neural-network structure used to determine the seven aircraft trim positions $\mathbf{u}_{\rm trim}$ and $\mathbf{x^2}_{\rm trim}$ for any desired trim steady-state $\mathbf{x^1}_{\rm trim}$. To make use of the existing NDP programming structure for seamless integration, it is natural to use the action network as the trim network. However now the trim network is a one layer weight feedforward network with seven biases ($\mathbf{b}$) that corresponding to the seven trim states. The network has a nonlinear sigmoid function fanning out the outputs. In using the action network for trim, the inputs to the trim network are zeroed and there are seven outputs from the network including four controls and three body angles.

State equations associated with Fig. 3 are as follows.

$$[\,\mathbf{u}_{\rm trim} \quad \mathbf{x^2}_{\rm trim}\,] = \mathbf{f}_{\rm sig}(\mathbf{b}) \tag{14}$$

where

$$\mathbf{f}_{\rm sig}(\mathbf{b}) = [f_{\rm sig}(b_1), \ldots, f_{\rm sig}(b_7)] \tag{15}$$

$$f_{\rm sig}(\xi) = 2\frac{(1-e^{-\xi})}{(1+e^{-\xi})}. \tag{16}$$

The cost function used is $J_t = 1/2\bar{\mathbf{x}}^T\bar{\mathbf{x}}$, where $\bar{\mathbf{x}}_i = (x^1{}_i - x^1{}_{\rm trim,i})/x^1{}_{\rm max,i}$, $i = 1, \ldots, 6$.

The basic technique for determining the trim positions is to train a network, whose inputs are zeroed and that has biases, to minimize an objective (cost function). After training is com-

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

plete, the resulting biases then provide the aircraft trim positions. The weights in this network are of no importance other than the bias vector. This trim network is trained over a number of epochs where in each epoch m the FLYRT model is initialized, the plant dynamics are evolved for a specified time t_f, and backpropagation is performed to adjust the trim network biases to minimize the objective at time t_f for that epoch m. In general, we denote the variable v at time t_f for epoch m as $v(m)$. For example, the objective function at time t_f for epoch m is denoted $J_t(m)$.

The network biases, $\mathbf{b}$, are trained using the same gradient descent training method described for NDP. That is

$$\mathbf{b}(m+1) = \mathbf{b}(m) + \Delta\mathbf{b}(m) \tag{17}$$

$$\Delta b_i(m) = \lambda(m)\left[-\frac{\partial J_t(m)}{\partial b_i(m)}\right], i = 1\ldots 7. \tag{18}$$

For the four biases corresponding to the trim controls

$$\frac{\partial J_t(m)}{\partial b_i(m)} = \frac{\partial J_t(m)}{\partial \mathbf{x^1}(m)}\frac{\partial \mathbf{x^1}(m)}{\partial u(m)}\frac{\partial u(m)}{\partial b_i(m)}, i = 1\ldots 4. \tag{19}$$

For the three biases corresponding to the trim body angles

$$\frac{\partial J_t(m)}{\partial b_i(m)} = \frac{\partial J_t(m)}{\partial \mathbf{x^1}(m)}\frac{\partial \mathbf{x^1}(m)}{\partial \mathbf{x^2}(m)}\frac{\partial \mathbf{x^2}(m)}{\partial b_i(m)}, i = 5\ldots 7. \tag{20}$$

In both cases the first partial is evaluated in the manner described earlier in the DNDP mechanism. For the trim angles bias updates, the partials $\partial\mathbf{x^1}(m)/\partial\mathbf{x^2}(m)$ and $\partial\mathbf{x^1}(m)/\partial\mathbf{u}(m)$ can be calculated numerically via perturbations to the system. Alternatively, the partial $\partial\mathbf{x^1}(m)/\partial\mathbf{x^2}(m)$ can be approximated analytically at hover from simple physics (rotate the gravity vector into the body frame and use appropriate small angle approximations). We computed the partials at hover only and used them for all flight conditions to generate our results.

Once training is complete the trim positions are then determined from the network biases via the sigmoid function. Note that the trim network is trained off-line first, independent of the action and the critic networks in the DNDP controller.

C. Neural-Network-Based Trim Algorithm Implementation

An implementation procedure of the trim algorithm can be summarized as follows.

Step 1) Specify the desired steady state to be trimmed as $\mathbf{x^1}_{\text{trim}}$.

Step 2) Initialize the trim network biases $\mathbf{b}(0)$.
Loop $m = 1$ to M, where M is the total number of epochs for determining trim.

Step 3) Set the FLYRT initial conditions and initial controls to the trim values

$$\mathbf{x}(t_o) = [\mathbf{x^1}_{\text{trim}}\ \mathbf{x^2}_{\text{trim}}(m-1)] \tag{21}$$

$$\mathbf{u}(t) = \mathbf{u}_{\text{trim}}(m-1), t_o \le t \le t_f. \tag{22}$$

Step 4) Let the plant dynamics evolve for time t_f (e.g., 500 ms).

TABLE III
NEURAL-NETWORK PARAMETER VALUES FOR DETERMINING TRIM

Parameter	λ_0 in control trim	λ_0 in attitude trim	t_f	K_{cont}
Value	3.0	0.3	500 ms	5.0

Step 5) Measure the states of interest

$$\mathbf{x^1}(m) = \mathbf{x^1}(t_f) \tag{23}$$

$$\mathbf{x^2}_{\text{trim}}(m) = \mathbf{x^2}(t_f). \tag{24}$$

Step 6) Evaluate the objective function $J_t(m)$.

Step 7) Update the network biases per the update (17)–(20).
End loop

Step 8) Compute the trim positions from the network biases per (14).

The number of epochs was chosen to be $M = 1000$ by experience. The learning rate was set to decrease as a function of epoch number, $\lambda(m) = \lambda_0(1 - m/M)$. For the results presented in this paper, the network biases were initialized to zero. However, one could use system knowledge to make a good initial guess of the trim positions and then calculate the corresponding biases and initialize the biases with these values. The model run time t_f is chosen to be large enough to have some measurable response but small enough so that linearity properties about that operating point holds during the run time.

Note that we desire to trim the aircraft for a specified ground speed (i.e., in the earth reference frame), rather than for body frame velocities (as implied by existing trim papers interested in aero-mechanical issues). A problem is that our model initialization and cost function are specified in terms of body frame velocities rather than earth frame reference frame velocities. The two reference frames are related by the still to be determined trim body angles. The solution to this problem is to use the current body angle trim estimates to resolve the two frames.

D. Trim Results

The neural-network-based trim method is tested for a variety of flight conditions for the Apache helicopter. Since there are an infinite number of trim solutions at a given operating condition (infinite combinations of ϕ_{trim} and ψ_{trim}), a unique solution was found by constraining ψ_{trim} to zero. The neural-network parameters used for determining the trim positions are presented in Table III.

This neural trim method has proven to be efficient for Apache helicopter through numerous simulation studies. Fig. 4 compares the NDP generated trim positions to those generated by FLYRT over the range of 0 to 150 kn forward speed. The results appear to be identical except for roll which has a worst case difference of less than 0.15°. Fig. 4 is typical for other flight conditions when compared to the FLYRT.

VI. DNDP–BASED TRACKING CONTROL RESULTS

This section presents results showing the performance of the DNDP controlling the Apache helicopter for a variety of maneuvers. This is a complex MIMO nonlinear control system design problem. It provides a realistic test bed for how well an approximate dynamic programming algorithm can generalize.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

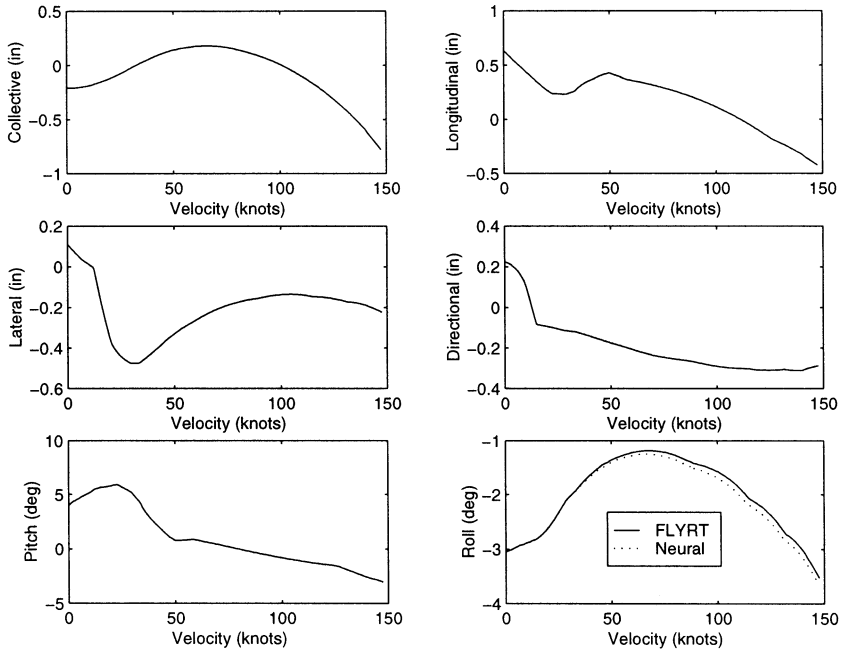

Fig. 4. Trim positions as a function of airspeed for neural and FLYRT methods.

TABLE IV
LEARNING STATISTICS FOR HOVER TO 50 FT/S MANEUVER AT VARIOUS ACCELERATIONS FOR THREE WIND CONDITIONS

Condition	Acceleration (ft/s^2)	2	3	4	5	6	7	8
Case A	Success Percentage	94%	62%	67%	65%	66%	66%	74%
	Average No. of Trials	1600	2019	2115	1950	1983	2028	1870
	Learning Deviation	214	339	324	307	293	306	252
Case B	Success Percentage	96%	66%	76%	70%	50%	57%	53%
	Average No. of Trials	1367	1720	1770	1874	2173	1970	2419
	Learning Deviation	191	280	255	275	381	337	400
Case C	Success Percentage	95%	98%	97%	85%	60%	56%	58%
	Average No. of Trials	642	824	1126	1843	1842	2145	2379
	Learning Deviation	115	128	165	263	313	333	403

Characteristic to prior NDP research, the performance of the DNDP is summarized statistically in tables. Fourteen maneuvers are considered: seven accelerations from hover to 50 ft/s at various accelerations and seven decelerations from 100 ft/s to 50 ft/s at various decelerations. Each maneuver is tested in three wind conditions: case A) no wind, case B) 10 ft/s step gust for 5 s, and case C) turbulence simulated using a Dryden model with a spatial turbulence intensity of $\sigma = 5$ ft/s and a turbulence scale length of $L_W = 1750$ ft. In addition to the tabular statistics provided, both statistical and typical time history plots of the aircraft states are provided for two cases, a hover to 50 ft/s at 5 ft/s^2 maneuver with turbulence and a 100 ft/s to 50 ft/s at -4 ft/s^2 maneuver with a step gust.

The statistical success of the DNDP to learn to control the helicopter is evaluated for the five flight conditions. For each flight condition, 100 runs were performed to evaluate the DNDP performance, where for each run initial weights in each network were set randomly. Each run consists of up to 5000 attempts (trials) to learn to successfully control the system. An attempt is deemed successful if the helicopter stays within the failure criteria bounds described in Table II for the entire flight duration (50 s). After each failed trial, the system is restarted with the same initial state as the previous trial but with the network weights at the previous trial's termination. If the controller successfully controls the helicopter within 5000 trials, the run is considered successful; if not, the run is considered a failure.

Table IV statistically summarizes the learning ability of the DNDP controller to perform a hover to 50 ft/s maneuver at a number of different accelerations. Results for 100 to 50 ft/s decelerations at a number of deceleration rates are provided in Table V. In both cases results are provided for the three wind conditions cited above. The success percentage reflects the percentage of runs for which the DNDP system successfully learns to control the helicopter. The average number of trials is what it takes the DNDP system to learn to control the helicopter. Stan-

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

TABLE V
LEARNING STATISTICS FOR 100 TO 50 FT/S MANEUVER AT VARIOUS DECELERATIONS FOR THREE WIND CONDITIONS

Condition	Acceleration (ft/s^2)	2	3	4	5	6	7	8
Case A	Success Percentage	98%	90%	85%	80%	84%	76%	73%
	Average No. of Trials	759	1700	1610	2114	1516	1800	2045
	Learning Deviation	105	227	226	291	219	248	292
Case B	Success Percentage	99%	85%	74%	71%	77%	76%	78%
	Average No. of Trials	1260	1460	1650	1979	2030	1950	1737
	Learning Deviation	181	215	229	298	295	264	239
Case C	Success Percentage	100%	98%	93%	93%	97%	89%	91%
	Average No. of Trials	778	1258	1373	1489	1236	1350	1677
	Learning Deviation	105	180	183	200	162	186	220

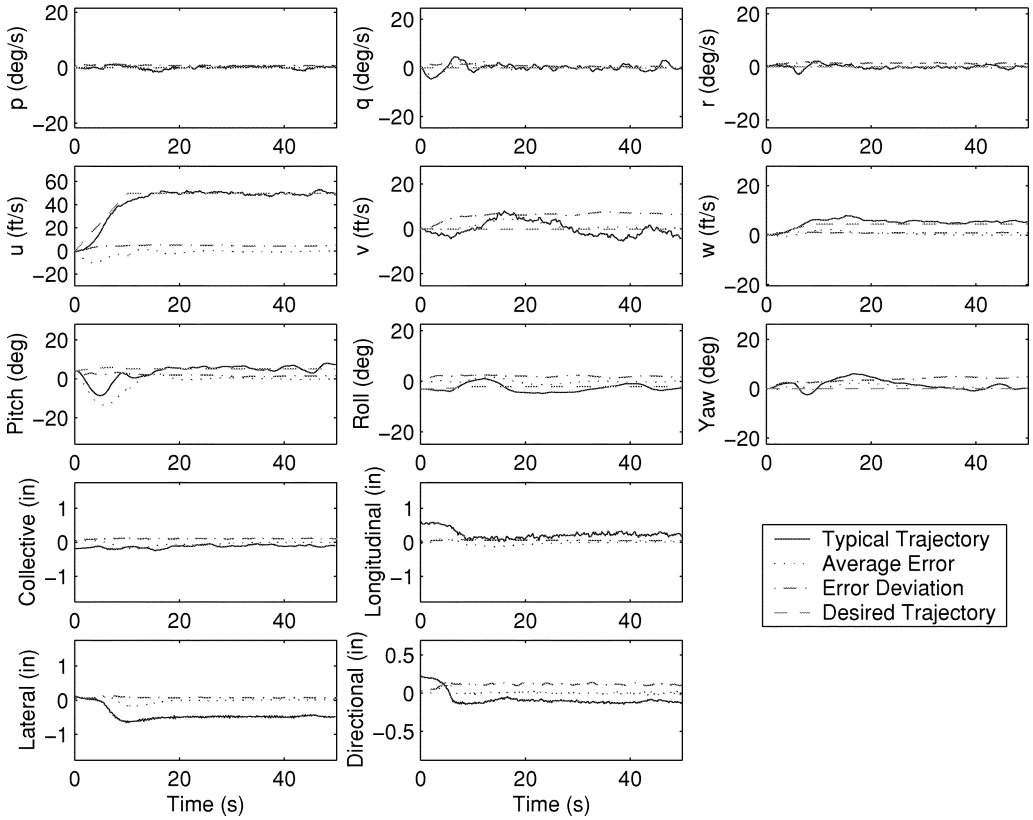

Fig. 5. Statistical and typical state and control trajectories of the helicopter for a hover to 50 ft/s maneuver at 5 ft/s^2 acceleration in turbulence. Turbulence is simulated using a Dryden model with a spatial turbulence intensity of $\sigma = 5$ ft/s and a turbulence scale length of $L_W = 1750$ ft.

dard deviations from successful runs for various maneuvers are also used to demonstrate the (in)consistency of the learning control performance.

Fig. 5 shows both the statistical average state error and error deviation over all successful runs and a typical plot of the controller performance for a hover to 50 ft/s maneuver at an aggressive 5 ft/s^2 acceleration in the presence of turbulence. Fig. 6 shows both the statistical average state error and error deviation and a typical plot of the controller performance for a 100 ft/s to 50 ft/s maneuver at 4 ft/s^2 deceleration in the presence of a step gust. Helicopter and control dynamics are similar for the other maneuvers once the learning controller becomes stabilized in learning.

The neural-network parameters used during training are provided in Table VI. The learning rates β for the action network and critic network are scheduled to decrease linearly with time (typically over a few seconds). In every time frame, the weight equations are updated until either the error has sufficiently converged ($E < E_{\text{tol}}$) or the internal update cycles (N_{cyc}) of the weights have been reached. N_{h} is the number of hidden nodes in the neural networks.

It is worth mentioning that a comprehensive analysis on the convergence performance of an entire NDP system in general does not exist, and neither does an analytical framework on the relationship between the performance of the NDP learning

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

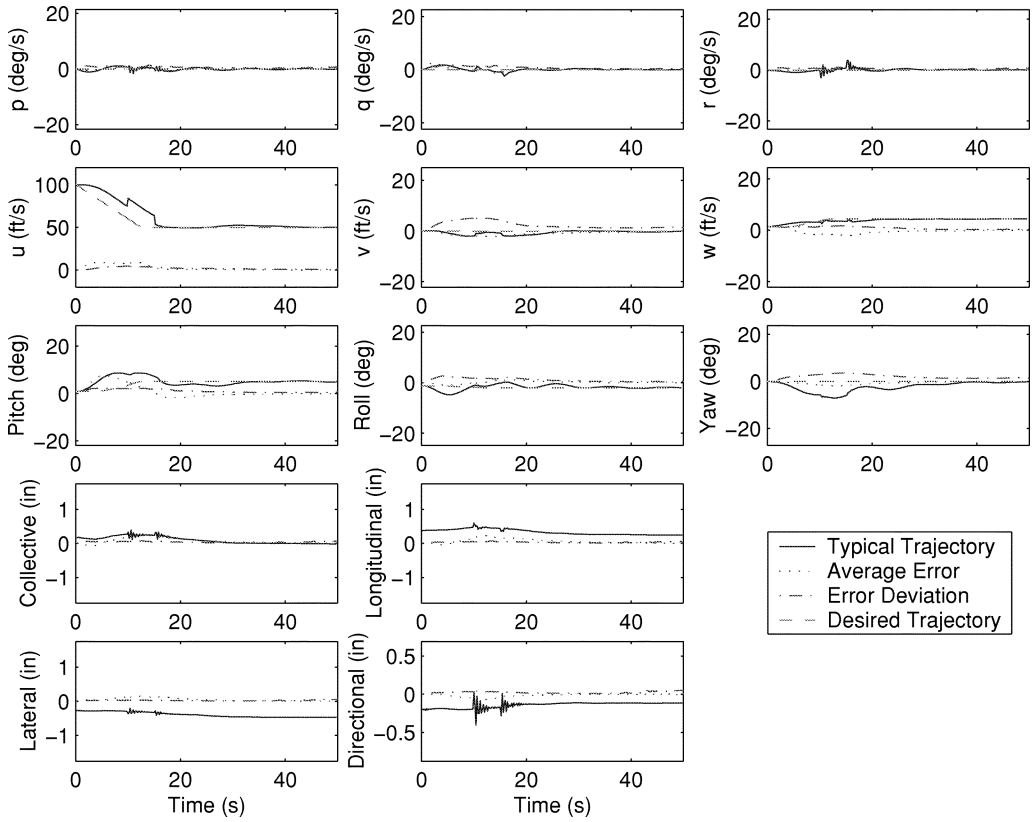

Fig. 6. Statistical and typical state and control trajectories of the helicopter for a 100 to 50 ft/s maneuver at -4 ft/s^2 acceleration in a step gust. The step gust is 10 ft/s in magnitude and has a 5-s duration.

TABLE VI
NEURAL-NETWORK PARAMETER VALUES FOR TRAINING

Parameter	α	$\beta_a(t_0)$	$\beta_a(t_f)$	$\beta_c(t_0)$	$\beta_c(t_f)$	$N_{cyc,a}$	$N_{cyc,c}$	$E_{tol,a}$	$E_{tol,c}$	N_h	K_{cont}
Value	0.95	0.02	0.02	0.1	0.01	200	100	0.005	0.1	6	2.5

controller versus the learning parameters. It has been argued [8] that updating individual networks alone, action or critic for example, may be viewed as a stochastic approximation problem and therefore, conditions similar to the Robbins–Monro algorithm may be used as guidelines in scheduling the learning parameters. Quantitatively, we have observed that the DNDP learning parameters do impact the learning ability of the learning controller. For example, the learning rate for action networks can be tuned to perform different maneuvers with different system outcomes. This is illustrated by Table VII, which shows the DNDP system performance for learning rates (β_a) of 0.2 and 0.02 for both more aggressive and less aggressive accelerations. Lower learning rates improve the success for more aggressive maneuvers but decrease the learning ability (increase the number of trials) required to learn for less aggressive maneuvers.

TABLE VII
LEARNING STATISTICS FOR HOVER TO 50 FT/S MANEUVERS AS A FUNCTION OF LEARNING RATE

β_a	Acceleration (ft/s^2)	2	6	8
0.02	Success Percentage	94%	66%	74%
	Average No. of Trials	1600	1983	1870
	Learning Variance	214	293	252
0.2	Success Percentage	100%	80%	20%
	Average No. of Trials	248	1708	2809
	Learning Variance	71	421	439

Interesting to note is that despite what one may expect, overall the DNDP controller more reliably and more quickly learns to control the helicopter in the presence of turbulence. This is clearly evident in Tables IV and V. The learning performance improvement can be attributed to the sustained larger excitation, due to turbulence, to both the neural-networks inputs and the cost function evaluation when there is turbulence. As a result, the network weights change more in (10) and, thus, the learning system explores more of the solution space and is less likely to become trapped in local minima that do not provide an adequate control solution. This suggests that in applications where turbulence or other excitation is not natural, it may be prudent to create an artificial equivalent in order to improve learning performance.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.

Also note that, as with previous NDP designs, a large number of trials must occur to successfully learn to perform the maneuver. Further, the more aggressive the maneuver (the higher the acceleration), the more trials that are required. This is not surprising for a learning system that is learning from experience without any a priori system knowledge. The ramification is that this training is done offline (i.e., not in a real helicopter), where failures can be afforded, until the controller is successfully trained. Once trained, the neural-network weights are frozen and the controller structure shown in Fig. 2 can be implemented in a helicopter. Limited authority on-line training can then be performed to improve system performance.

VII. Conclusion

This paper has advanced neural dynamic programming control research by introducing the DNDP control structure to a sophisticated tracking control problem, contrasting to earlier related works that have been limited to stabilization only [9]. Paramount to this was the development of a DNDP-based method for trimming the helicopter system and a structured approach to implementing the action network. A sophisticated nonlinear validated model of the Apache helicopter was used to test the controller and its ability to learn to perform a number of difficult maneuvers. Our research has shown that the DNDP is able to successfully control the Apache helicopter for a wide range of realistic maneuvers and over a wide range of flight conditions, a few examples of which were used to illustrate the results in this paper. Thus, it appears that NDP is a viable candidate for controlling complex MIMO systems and is suited particularly well for on-line and complex multiaxes coupling control applications. Our results have demonstrated the generalization capability of a learning control system, namely DNDP, to a large continuous-state problem. The same principle can also be directly applied to large discrete-state/action problems.

References

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[2] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[3] P. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control*, D. White and D. Sofge, Eds. New York: Van Nostrand, 1992.

[4] . presented at NSF Workshop on Learning and Approximate Dynamic Programming. [Online]. Available: http://ebrains.la.asu.edu/ nsfadp/

[5] R. Munos and A. Moore, "Variable resolution discretization in optimal control," *Machine Learning*, vol. 49, no. 2–3, pp. 291–323, Nov. 2001.

[6] J. Boyan and A. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Proc. Neural Information Processing Systems 7*, Jan. 1995, pp. 731–739.

[7] R. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Proc. Neural Information Processing Systems 8*, Cambridge, MA, 1996, pp. 1038–1044.

[8] J. Si and J. Wang, "On-Line learning by association and reinforcement," *IEEE Trans. Neural Networks*, vol. 12, pp. 264–276, Mar. 2001.

[9] R. Enns and J. Si, "Apache helicopter stabilization using neural dynamic programming," *AIAA J. Guidance, Contr., Dyn.*, vol. 25, no. 1, pp. 19–25, Jan.–Feb. 2002.

[10] D. Sadhukhan and S. Feteih, "F8 neurocontroller based on dynamic inversion," *AIAA J. Guidance, Contr., Dyn.*, vol. 19, no. 1, pp. 150–156, Jan.–Feb. 1996.

[11] M. Napolitano and M. Kincheloe, "On-line learning neural-network controllers for autopilot systems," *AIAA J. Guidance, Contr., Dyn.*, vol. 18, no. 6, pp. 1008–1015, Nov.–Dec. 1995.

[12] B. Kim and A. Calise, "Nonlinear flight control using neural networks," *AIAA J. Guidance, Contr., Dyn.*, vol. 20, no. 1, pp. 26–32, Jan. 1997.

[13] B. Kim, "Nonlinear flight control using neural networks," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, 1994.

[14] A. Calise, "Neural networks in nonlinear aircraft flight control," *IEEE Aerosp. Electron. Syst. Mag.*, pp. 5–10, July 1996.

[15] C. Ha, "Neural networks approach to AIAA control design challenge," *AIAA J. Guidance, Contr., Dyn.*, vol. 18, no. 4, pp. 731–739, July 1995.

[16] S. Balakrishnan and V. Biega, "Adaptive-critic-based neural networks for aircraft optimal control," *AIAA J. Guidance, Contr., Dyn.*, vol. 19, no. 4, pp. 731–739, July–Aug. 1996.

[17] D. Prokhorov and D. Wunsch II, "Adaptive critic designs," *IEEE Trans. Neural Networks*, vol. 8, pp. 997–1007, Sept. 1997.

[18] F. Kim and R. Celi, "Forward flight trim and frequency response validation of a helicopter simulation model," *J. Aircraft*, vol. 30, no. 6, pp. 854–863, Nov.–Dec. 1993.

[19] J. McVicar and R. Bradley, "Robust and efficient trimming algorithm for application to advanced mathematical models of rotorcraft," *J. Aircraft*, vol. 32, no. 2, pp. 439–442, Mar.–Apr. 1995.

[20] N. Achar and G. Gaonkar, "Helicopter trim analysis by shooting and finite element methods with optimally damped Newton iterations," *AIAA J.*, vol. 31, no. 2, pp. 225–234, Feb. 1993.

[21] J. McVicar and R. Bradley, "Efficient and robust algorithms for trim and stability analysis of advanced rotorcraft simulations," *Aeronautical J.*, vol. 101, no. 1008, pp. 375–387, Oct. 1997.

[22] D. Peters and B. Kim, "Control settings for a trimmed stalled rotor by an automatic feedback system," in *Proc. AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, Materials Conf.*, New York, Apr. 6–10, 1981, pp. 463–470.

[23] M. Peters and D. Peters, "Discrete control theory and dynamic observers applied to rotorcraft stability and trim," in *Proc. 54th Annu. Forum Amer. Helicopter Soc.*, Washington, DC, May 20–22, 1998, pp. 684–699.

[24] D. Peters, P. Bayly, and S. Li, "Hybrid periodic-shooting, auto-pilot method for rotorcraft trim analysis," in *Proc. 52nd Annu. Forum Amer. Helicopter Soc.*, Washington, DC, June 4–6, 1996, pp. 780–793.

[25] D. Peters and D. Barwey, "General theory of rotorcraft trim," in *Proc. AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, Materials Conf.*, New Orleans, LA, Apr. 10–13, 1995, pp. 2558–2590.

[26] S. Kumar, J. Harding, and S. Bass, "AH-64 Apache Engineering Simulation Non-Real Time Validation Manual," USAAVSCOM TR 90-A-010, 1990.

Russell Enns received the B.A.Sc. degree in electronics and communications from Simon Fraser University, Burnaby, BC, Canada, in 1990 and the M.S. and Ph.D. degrees in electrical engineering from Arizona State University, Tempe.

From 1990 to 1991, he was a Design Engineer for Interactive Circuits and Systems, Ltd., Ottawa, ON, Canada, and from 1991 to 1993, he was an ASU Industrial Fellow sponsored by McDonnell Douglas Helicopter Company (MDHC), Mesa, AZ. He has been developing fire and flight controls systems at MDHC/Boeing since 1993. His research interests include control system development and the application of artificial neural learning systems to controls.

Jennie Si received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1985 and 1988, respectively, and the Ph.D. degree from University of Notre Dame, Notre Dame, IN.

Since 1991, she has been on faculty at Arizona State University, Tempe, where she is now Professor in the Department of Electrical Engineering. Her current research interest includes theory and application of artificial neural learning systems. Major application areas of her research are learning controllers, semiconductor manufacturing process optimization, and neural cortical information processing.

Dr. Si is a recipient of the 1995 NSF/White House Presidential Faculty Fellow Award. She was Associate Editor of the IEEE Transactions on Automatic Control in 1998 and 1999, IEEE Transactions on Semiconductor Manufacturing from 1998 to 2002, and IEEE Transactions on Neural Networks since 2000.

Authorized licensed use limited to: BEIJING UNIVERSITY OF POST AND TELECOM. Downloaded on March 02,2010 at 23:40:37 EST from IEEE Xplore. Restrictions apply.