JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 25, No. 1, January–February 2002

Apache Helicopter Stabilization Using Neural Dynamic Programming

Russell Enns* and Jennie Si†
Arizona State University, Tempe, Arizona 85287

A new form of neural control is introduced, neural dynamic programming (NDP), a model-free online learning control scheme. NDP is shown to perform exceedingly well as a learning controller for practical systems of higher dimension, such as helicopters. The discussion is focused on providing a viable alternative helicopter control system design approach rather than providing extensive comparisons among various available controllers. A comprehensive treatise of NDP and extensive simulation studies of NDP designs for controlling an Apache helicopter under different flight conditions is presented. Design robustness is addressed by performing simulations under various disturbance conditions. All of the designs are based on FLYRT, a sophisticated industry-scale nonlinear validated model of the Apache helicopter.

Nomenclature

a	=	subscript for action network
$b_{i,k}$	=	bias for node k in layer i
c	=	subscript for critic network
e	=	error signal
J	=	approximation of R
p, q, r	=	aircraft body rates (roll, pitch, yaw), rad/s
R	=	cost function
r	=	reinforcement signal
U_c	=	ultimate cost
u, v, w	=	aircraft velocities, body axes (longitudinal, lateral, vertical), ft/s
$\boldsymbol{u}_a$	=	actuator control vector $[z_a, z_b, z_c, z_d]$, in.
$w_{i,j,k}$	=	weight for node k in layer i with input j
$\boldsymbol{x}$	=	state vector $[u, w, q, \theta, v, p, r, \phi, \psi]$
α	=	discount factor
β	=	network learning rate
ϕ, θ, ψ	=	aircraft Euler angles (roll, pitch, yaw), rad

Introduction

ALTHOUGH neural networks have been used for control purposes for well over the last decade, their application either has been limited to low-dimensional plants, typically a single control and occasionally two controls, or has been limited to higher-dimensional systems that can be suitably decoupled into simpler subsystems.[1–3] This paper demonstrates how a new neural network control scheme, neural dynamic programming (NDP), can be made to control more realistic higher-dimensional systems such as helicopters by providing an approximate solution to optimal control problems that are often solved by dynamic programming.

The original motivation for using an NDP-based helicopter control methodology was to find reconfigurable control solutions for helicopters. NDP was perceived as a strong candidate for such a task because of a number of its features. In particular, it is a learning control system that does not require system model knowledge a priori (it is model free) and it can be applied to complex systems such as helicopters without the need to decouple the control system into simpler subsystems. In fact, it can learn to take advantage of any of the system's cross-coupling characteristics when generating its control solution, including coupling benefits that may not be apparent to a control systems design engineer. However, a necessary first step in determining the merits of such a control methodology is to determine if it is satisfactorily able to control, and in particular stabilize, an unstable aircraft. This is especially relevant because applications of NDP to date have themselves been limited to simple single control systems,[4–7] hence, the focus of this paper.

The potential advantages of NDP over other optimal control based solutions may be summarized as follows. First, NDP does not require an explicit model of the system that is to be controlled. The controller learns and improves its performance on the fly. Second, NDP avoids the curse of dimensionality that dynamic programming suffers from by providing approximate solutions.[8] This, however, may also be considered as the downside of NDP when true optimality is demanded. Third, NDP does not require an explicitly defined system performance measure, R, which is usually a function of the system states and the control actions in the classical optimal control theory.

NDP can be used alone, or it can augment other control methods to improve system performance in the presence of model errors and uncertainties and unmodeled plant nonlinearities. Such a blending of techniques allows us to combine the optimality of classic techniques such as linear quadratic control (under assumptions of linearity) with NDP's ability to compensate for nonlinearities and modeling errors. For the purposes of this paper, we restrict our attention to using NDP alone.

Numerous flight control design methods have been developed over the last several decades. Classic flight control design methods, consisting of methods such as proportional–integral–derivative (PID) control, date back to the earliest flight control systems[9] and are still being used today with much success. Modern control design methods such as linear quadratic regulators, linear quadratic Gaussian, control, loop transfer recovery,[10,11] model following,[11] eigenstructure assignment,[12] receding horizon optimal control,[13] H-infinity control, and variable structure control[14] have been used with varying degrees of success.

The application of neural networks to flight control has been more recent, and most results either have been limited to simulation studies of simple (usually scalar) control subsystems,[15–19] or have decoupled the sophisticated systems into smaller subsystems guided by the designer's expertise.[20–22] Several of these papers use neural networks to either approximate or to improve on the approximation of an aircraft's inverse dynamics. There exist some notable exceptions to this. For example, Ha[23] 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.[4] However, the research limits itself to the longitudinal axis, and as a result, the system only has a single

Received 1 June 2000; revision received 21 March 2001; accepted for publication 26 March 2001.

*Graduate Student, Department of Electrical Engineering.
†Professor, Department of Electrical Engineering.

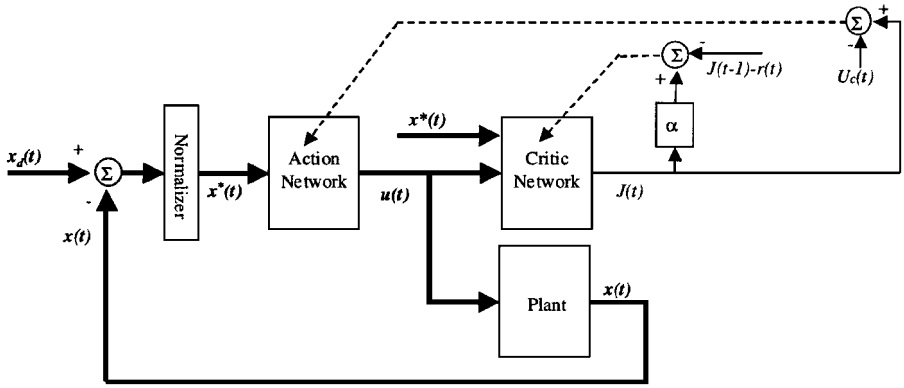

Fig. 1 Basic NDP controller.

control. Further, an explicit system model was used, which contrasts to NDP, where such a model is not necessarily required.

Kim and Calise have contributed a large body of work that uses neural networks to improve on an underlying dynamic model inversion control methodology.[20–22,24] The neural networks compensate for any model inversion error that exists by augmenting a control adjustment to the nominal P–D control term. Generally, these methods are applied to either a single control axis or the flight control system is decoupled into individual control axes with a neural network for each axis.

Much of the flight control research to date either uses a linear model, or a fairly simple nonlinear model, and does not model actuator dynamics. Some notable exceptions to this are the work in Refs. 20–22 and 24 already described. No such limitations are made in this paper either.

The focus of the paper is on using NDP designs to stabilize a helicopter. In particular, we show how NDP can be used to stabilize the aircraft for five flight conditions, hover, 30, 60, 90, and 120 kn. Simulations are performed in both clear air and in the presence of turbulence and step gusts. Our NDP designs and our simulations are conducted using the FLYRT model, a sophisticated and very realistic system with nonlinearities, sensor and actuator dynamics, etc.

The paper is organized as follows. We first cover NDP comprehensively in the next section. In the third section, we briefly describe the helicopter model. The fourth section defines our design objectives and provides results from simulation studies. Some discussions and conclusions are then given in the last section.

NDP and Multi-Input/Multi-Output Rotorcraft Control

The objective of a NDP controller is to optimize a desired performance measure by learning to create appropriate control actions through interaction with the environment. The controller is designed to learn to perform better over time using only sampled measurements and with no prior knowledge about the system.

Dynamic programming has been applied extensively in different fields of engineering, operations research, economics, and so on. Unfortunately the computational costs of dynamic programming are often very high, a result of the so-called curse of dimensionality. In recent years, a new approach to dynamic programming has surfaced, which may represent a breakthrough in the practical applications of dynamic programming to complex problems. Initial expositions of this simulation-based approximate dynamic programming technique were referred to as reinforcement learning.[25,26] Using artificial neural networks for approximate dynamic programming, or NDP, was proposed in Refs. 27–29.

One of the early reinforcement learning paradigms was introduced by Barto et al. in 1983 (Ref. 25), in which pattern-based dynamic programming was proposed. Also in that original creation, a basic form of the temporal difference (TD) (λ) algorithm started to take shape. A more systematic treatment of the TD (λ) was provided by Sutton in 1988 (Ref. 26). The notion of Q learning was consequently introduced by Watkins[30] shortly after. Prokhorov and Wunsch have developed a number of NDP paradigms, starting from heuristic programming. However, in their autolander application, their results are only for the scalar control case.[5,6] Bertsekas and Tsitsiklis[31] provide a more systematic treatment of approximate dynamic programming, which includes NDP. Reference 31 reveals the connections between classic dynamic programming and NDP from a theoretic viewpoint. It also represents a broad collection of suboptimal control methods in addition to neural-network-based approximate dynamic programming. However, a thorough and effective treatment of the design and implementation of NDP is only in the early stages. More work is needed in all aspects from basic concepts, to implementation, application, and a thorough systematic analysis.

This section defines the underlying NDP control framework that this work is based on and then expands on this definition to provide a more comprehensive framework that can serve as a basis for a helicopter flight control system. A significant difference between previous NDP work[4,6,7] and the more comprehensive framework presented here is that the earlier work has been limited to scalar control, whereas the latter is applicable to multi-input/multi-output (MIMO) control. Though the basic design framework for the two are similar, the multiple control implementation is significantly more complex. The underlying NDP control framework, shown in Fig. 1, consists of two main blocks, an action generator and a critic network, each of which is described in the following sections.[7]

The essence of the NDP critic is to approximate a cost function $R(t)$ by $J(t)$ through learning, where $R(t)$ can be a future reward-to-go and $J(t)$ is the critic network output in Fig. 1. When this approximation is achieved, one has obtained an (approximate) solution to the Bellman equation. The learning is performed without requiring an explicit system model (such as the form $x(t+1) = f[x(t), u(t)]$). Instead, the system dynamics information is implicitly absorbed by both the action and critic networks. The approximate solution to the Bellman equation is implemented one-step back in time instead of one-step forward. Therefore, the need for state prediction to obtain $x(t+1)$ [and, thus, $J(t+1)$] is replaced by the requirement to store $J(t-1)$, as shown in Fig. 1.

Both the action and critic networks are trained toward optimizing a global objective, namely, the Bellman equation for the critic network and an ultimate performance objective for the action network. During the learning process, the action network is constrained by the critic network to generate controls that optimize the future reward-to-go instead of only temporarily optimal solutions. For the helicopter stabilization application, the future reward-to-go is used to measure the success or failure of each learning attempt, and the ultimate performance objective U_c is to stabilize the helicopter for the specified flight condition.

In contrast to usual neural network applications, there is no readily available training sets of input–output pairs used for approximating $R(t)$ in terms of a least-squares fit. Instead, both the control action u and the critic output J are updated according to an error function that changes from one time step to the next. Therefore, the steepest decent algorithm does not hold valid for either of the two networks. With this characteristic in mind, a recursive stochastic algorithm developed by Robbins and Monro has been used to characterize convergence in a statistical average sense for the action and

critic networks individually. Namely, with one network completed updating, the other network with the updating rules to be described asymptotically reaches a (local) minimum of the statistical average of the error objective function for the network.[7]

Critic Network

The critic network approximates a cost function should an explicit cost function not be convenient or possible to represent. For example, the network output $J(t)$ can approximate a cost function such as the discounted total reward-to-go,

$$R(t) = r(t+1) + \alpha r(t+2) + \alpha^2 r(t+3) + \cdots \tag{1}$$

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 $r(t+1)$ is the external reinforcement value at time $t+1$.

Typically NDP has been applied to systems where explicit feedback 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). In the flight control case discussed here, because more explicit state information is available continually, we can extend $r(t)$ to be a more informative quadratic reinforcement signal, that is,

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

where x_i is the ith state of the state vector $\boldsymbol{x}$, $x_{i,d}$ is the desired reference state, and $x_{i,\max}$ is a state normalization factor.

The critic network can be implemented with a standard multilayer feedforward network. The network can be linear or have a nonlinear sigmoid function to fan out outputs depending on the complexity of the problem. We typically use a two-layer weight network with sigmoid functions for the nonlinearities as shown in Fig. 2. The network output, denoted by the vector $\boldsymbol{o} = [\boldsymbol{o}_1 \cdots \boldsymbol{o}_p]^T$ in Fig. 2, is simply the scalar J in the critic network.

The critic network is trained as follows. Define the prediction error for the critic element as

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

which also represents the principle of optimality if the equation is balanced, that is, $e_c(t) = 0$. The learning objective is to minimize the square of the prediction error in terms of the critic network output,

$$E_c(t) = \tfrac{1}{2} e_c^2(t) \tag{4}$$

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

$$w_c(t+1) = w_c(t) + \Delta w_c(t) \tag{5}$$

$$\Delta w_c(t) = \beta_c(t) \left[-\frac{\partial E_c(t)}{\partial w_c(t)} \right] = \beta_c(t) \left[-\frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial w_c(t)} \right] \tag{6}$$

where $\beta_c(t)$ is the learning rate of the critic network at time t, which usually decreases with time to a small value. The network weights are adjusted according to Eqs. (5) and (6), where

$$\frac{\partial E_c(t)}{\partial J(t)} = \frac{\partial E_c(t)}{\partial e_c(t)} \frac{\partial e_c(t)}{\partial J(t)} = \alpha e_c(t) \tag{7}$$

and $\partial J(t)/\partial w_c(t)$ is a function of the critic network's structure.

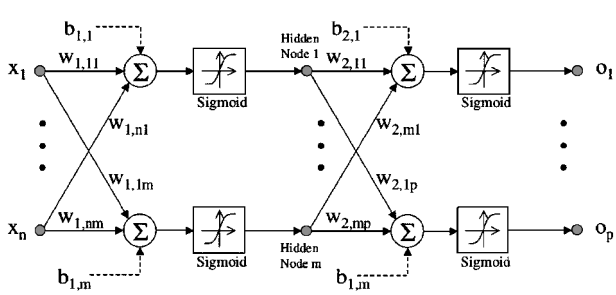

Fig. 2 Two-layer nonlinear feedforward neural network.

Action Network

The action network generates the desired plant control given measurements of the plant states. As with the critic network, the action network can be implemented with the standard multilayer linear or nonlinear feedforward neural network shown in Fig. 2. In this case, the number of network outputs equals the control space dimension.

The principle in adapting the action network is to backpropagate the error between the desired ultimate cost objective, denoted by U_c, and either the actual cost or the cost approximation. U_c is set to 0 in the helicopter control application. Either the actual cost function $R(t)$ or an approximation to it, $J(t)$, can be used depending on whether an explicit cost function is available or a critic network needs to be used. In the latter case, which is used in this paper, backpropagation is done through the critic network.

The weight updating in the action network is formulated as follows. Let

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

The weights in the action network are updated to minimize the following performance error measure:

$$E_a(t) = \tfrac{1}{2} e_a^2(t) \tag{9}$$

The update algorithm is similar to that used for the critic network. By a gradient descent rule,

$$w_a(t+1) = w_a(t) + \Delta w_a(t) \tag{10}$$

$$\Delta w_a(t) = \beta_a(t) \left[-\frac{\partial E_a(t)}{\partial w_a(t)} \right] \tag{11}$$

where

$$\frac{\partial E_a(t)}{\partial w_a(t)} = \frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u(t)} \frac{\partial u(t)}{\partial w_a(t)} \tag{12}$$

$$\frac{\partial E_a(t)}{\partial J(t)} = \frac{\partial E_a(t)}{\partial e_a(t)} \frac{\partial e_a(t)}{\partial J(t)} = e_a(t) \tag{13}$$

where $\partial J(t)/\partial u(t)$ is calculated through the critic network if a critic is used or through the cost function if it is used and $\partial u(t)/\partial w_a(t)$ is calculated through the action network and depends on the network's particular structure.

MIMO NDP Implementation for Helicopter Control

The basic NDP framework described needs to be further developed to actually implement NDP as a viable form of helicopter control. Expanding on this paradigm results in the more sophisticated NDP control structure, shown in Fig. 3. Such a control structure can also be used to solve command tracking and other control problems more sophisticated than stabilization. This section describes the details of this more advanced NDP controller and the rationale behind them.

First, a trim network that schedules the nominal control trim position as a function of aircraft state and environmental/flight parameters (such as aircraft weight, air density, etc.) is required. Having NDP determine the control trim position is key to successfully using NDP to control general systems. Previous NDP control designs were successful because the systems that were tested, for example, the inverted pendulum, had a zero trim requirement.[7] Often flight control papers have presented control methodologies that were designed and tested on linear models that have a zero trim requirement because the linear model is linearized about a trim condition. Such linearization can misleadingly indicate good results. This, too, would be the case for NDP and, hence, the required use of an accurate nonlinear model, which, as a consequence of its nonlinearities, has a nonzero trim requirement.

The trim network, shown in Fig. 3, is trained using a collected set of trim data over the range of flight conditions of interest. An NDP-based method for determining the trim position for a given flight condition has been developed, and a summary of the technique is presented here.[32]

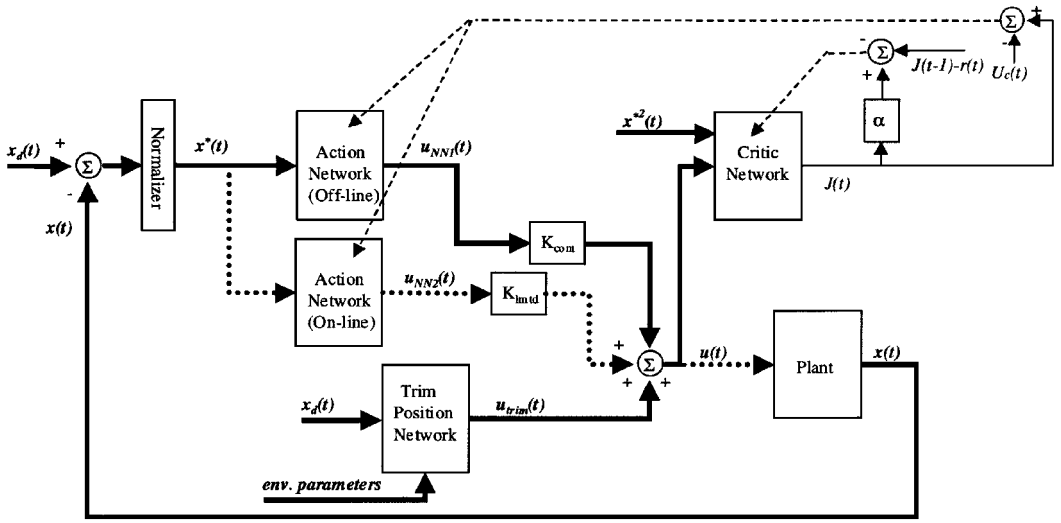

Fig. 3 Comprehensive NDP controller.

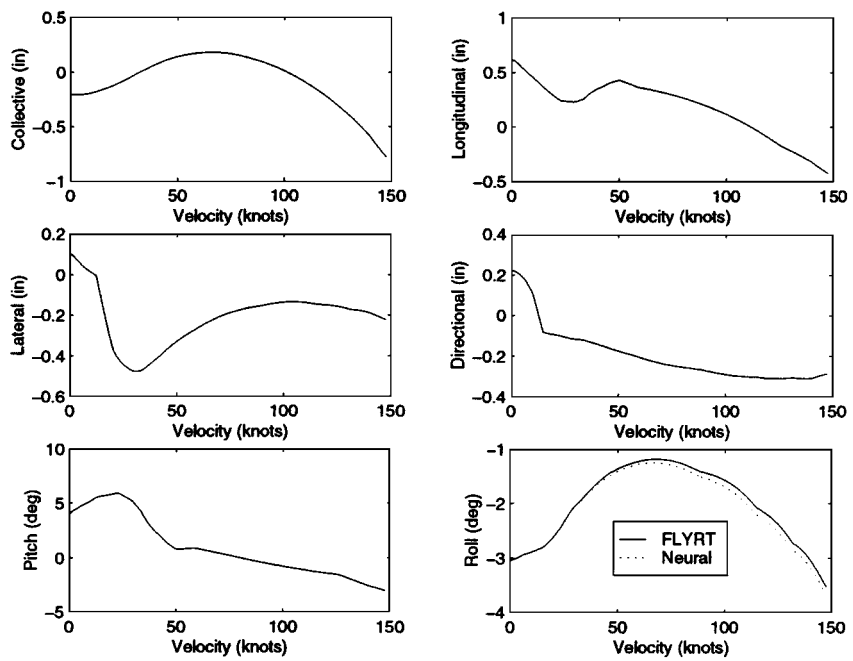

Fig. 4 Trim positions as a function of airspeed for neural and FLYRT methods.

The NDP-based method for determining the trim control position for a given flight condition uses the offline action network with its inputs zeroed. An initially assumed trim control position is picked, using system knowledge if available. With the network inputs zeroed, the trim position can be equivalently stored as network biases in the offline action network. The system is repeatedly evolved over a specified time interval (200 ms) with the actuators fixed at the trim position's current value. During each iteration, backpropagation training is used to tune the network biases (equivalently the trim positions) until the squared error between the system's evolved states and the system's desired trim states is minimized sufficiently. During the training procedure, the network's learning rate β_a is set to decrease as a function of iteration number.

The method works very well, as shown in Fig. 4, which compares the NDP generated trim positions to those generated by FLYRT over the range of 0–150 kn forward speed. In fact, the results appear to be identical except for roll, which has a worst-case difference of less than 0.15 deg.

Several other considerations need to be made to implement the NDP controller for control of systems such as a helicopter. First, the action network is implemented as a traditional two-layer feedforward network. However, because the action neural network's output (control) is typically limited to ± 1 by the sigmoidal nonlinearity present in the last stage of the network, a control scaling factor is used for each control. The value chosen is typically $K_{\text{cont}} = u_{\max}$ where $u_{\max}$ is the maximum control authority of the actuators. It is necessary to incorporate this scaling into the backpropagation when training the action network.

Second, the quadratic reinforcement signal described earlier is used. Not only does this provide better information than the binary reinforcement signal, it is requisite for the command tracking control problem. Additionally, the normalization factor used in the reinforcement function is decreased as a function of time at a specified rate until it reaches a lower limit. This allows the relative importance of each state to change with time as required by the application.

Third, the failure criteria used in the original NDP framework are also decreased as a function of time. This helps reduce the allowable error in the plant's states as time increases.

Fourth, the input to the action network, x^*, needs to be a normalization of $x - x_d$ rather than simply a normalization of x. This provides nonzero state stabilization and command tracking capabilities. The input to the critic is $(x^*)^2$, which helps shape the J form and has resulted in significant performance improvements over results in previous research.

Fifth, network biases are added to the action network to accommodate control biases and disturbances in the plant (much like integrators are added to linear quadratic controllers).

Finally, an additional action neural network (ANN) can be implemented to perform online learning to adapt to local flight conditions while the first ANN's weights are frozen after having been trained offline under specific common flight conditions. The second ANN's online weight adaptations based on its experiences should improve the controller's performance. The second ANN can be authority limited as required by the application. The results in this paper do not include the second ANN.

Helicopter Model

NDP is tested using the helicopter model shown in Fig. 5. The model, run at 50 Hz, consists of three parts: an actuator model, an actuator to blade geometry model, and FLYRT.

At the heart of the helicopter model is FLYRT, a sophisticated nonlinear flight simulation model of the Apache helicopter developed by The Boeing Company over the past two decades.[33] FLYRT models all of the forces and moments acting on the helicopter. The rotor is modeled using a blade element model. FLYRT dynamically couples the six-degree-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.

In addition to FLYRT, our model also consists of actuator models as well as a model of the mechanical geometry between the actuators and the helicopter blades. 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. Thus, the inputs to FLYRT are the four commanded blade angles of the helicopter: collective, longitudinal, lateral, and directional. The outputs from FLYRT are numerous; for flight control purposes, they are limited to the aircraft's translational and rotational velocities and the aircraft's orientation for a total of nine states.

The operating conditions for which our simulation studies are performed are shown in Table 1. The center of gravity (c.g.) is listed in the standard Apache fuselage station/water line/butt line coordinate frame.[33]

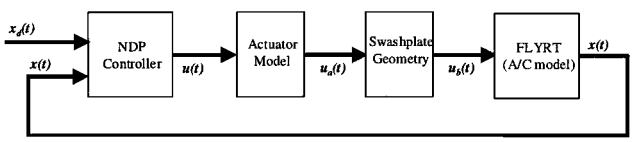

Fig. 5 Flight control system overview.

Results

This section presents results showing the performance of NDP in stabilizing the Apache helicopter. Characteristic to previous NDP research, the performance of NDP is summarized statistically in tables. Five flight conditions are considered, the stabilization of the helicopter at hover and at 30, 60, 90, and 120 kn. Each flight condition is tested in three wind conditions: case **A**, no wind; case **B**, 10-ft/s step gust for 5 s; and case **C**, Dryden turbulence 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 (Figs. 6 and 7) of the aircraft states are provided for two cases, turbulence at 30 kn and step gusts at 90 kn. Figures 6 and 7 show the performance during the testing phase. Time history plots for the other flight conditions are similar.

The objective of stabilization is to drive all aircraft states to their desired values for the given flight conditions during network training, regardless of the vehicle's initial conditions. The states of interest are the aircraft's translational (u, v, w) and rotational (p, q, r) velocities and the aircraft's Euler angles pitch θ, roll ϕ, and yaw ψ. Failure criteria are used to bound each state's allowed error. The allowed errors, shown in Table 2, are initially large and decrease as a function of time to an acceptable minimum.

Table 1 Helicopter operating conditions

Parameter	Value
Weight	16,324 lb · ft
Fuselage station, c.g.	201.6 in.
Butt line, c.g.	0.2 in.
Water line, c.g.	144.3 in.
Temperature	59°F
Altitude	1,770 ft

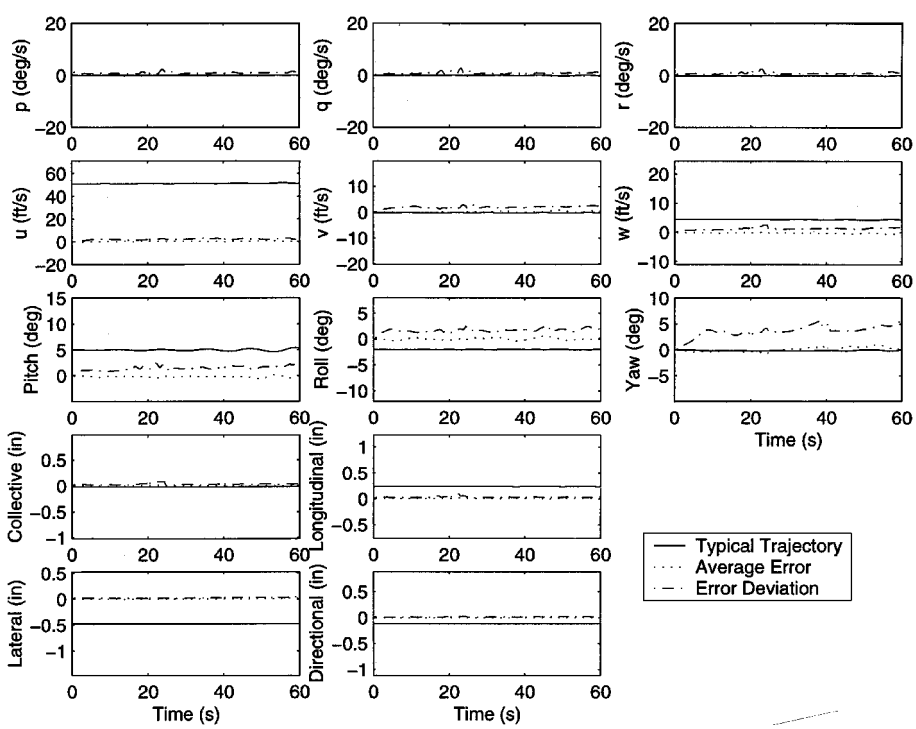

Fig. 6 Typical and statistical state and control trajectories of NDP stabilization at 30 kn in turbulence.

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 stabilize the helicopter both in nominal conditions and when subjected to disturbances. Heuristic failure criterion 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 statistical success of the NDP controller's ability to learn to control the helicopter is evaluated for the five flight conditions. For each flight condition, 100 runs were performed to evaluate NDP's performance, where for each run the neural networks' initial weights were set randomly. Each run consists of up to 500 attempts (trials) to learn to control the system successfully. An attempt is deemed successful if the helicopter stays within the failure criteria bounds described in Table 2 for the entire flight duration (1 min). If the controller successfully controls the helicopter within 500 trials, the run is considered successful, if not, the run is considered a failure.

The statistical training results for the 15 flight conditions simulated are shown in Table 3. The percentage of successful runs reflects the percentage of runs for which the NDP system successfully learns to control the helicopter. The average number of trials is the average number of trials that it takes the NDP system to learn to control the helicopter.

The neural network parameters used during training are provided in Table 4. The learning rates, β, for the ANN and critic network are potentially 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 N_{cyc} internal update cycles of the weights have occurred. N_h is the number of hidden nodes in the neural networks. Note that these parameters were chosen based on experience but were not tuned to optimize the results.

The results indicate that a large number of trials must be made before successful stabilization. This is not suprising for a learning system that is learning from experience without any a priori system knowledge. The ramification is that this training is done offline, that is, not in a real helicopter, where failures can be afforded, until the controller is successfully trained. Once trained, the neural network

Table 2 Failure criterion for helicopter stabilization

Aircraft state	Initial allowed error	Final allowed error	Error rate
u, v, w	20 ft/s	4 ft/s	−0.8 (ft/s)/s
p, q, r	30 deg/s	6 deg/s	−1.2 (deg/s)/s
θ, ϕ, ψ	30 deg	6 deg	−1.2 deg/s

Table 3 Learning statistics for NDP control of the Apache helicopter for three wind conditions

	Condition				
Case	Hover	30 kn	60 kn	90 kn	120 kn
Case A					
Percentage of successful runs, %	100	100	100	100	100
Average number of trials	18	47	36	30	70
Case B					
Percentage of successful runs, %	28	72	82	66	30
Average number of trials	201	148	162	176	191
Case C					
Percentage of successful runs, %	88	63	74	77	45
Average number of trials	128	214	206	186	200

Table 4 Neural network parameter values

Parameter	Value
$\beta_a(t_0)$	0.1
$\beta_a(t_f)$	0.1
$\beta_c(t_0)$	0.1
$\beta_c(t_f)$	0.01
N_{cyc}, a	200
N_{cyc}, c	100
E_{tol}, a	0.005
E_{tol}, c	0.1
N_h	6
K_{cont}	5.0

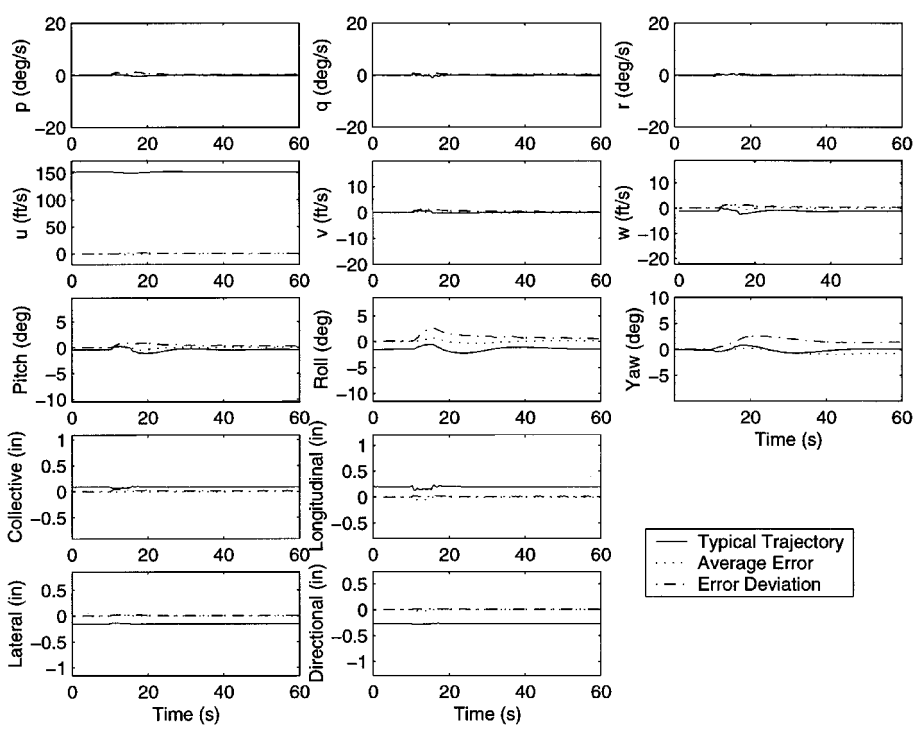

Fig. 7 Typical and statistical state and control trajectories of NDP stabilization at 90 kn with step gust.

weights are frozen, and the controller structure shown in Fig. 3 can be implemented in a helicopter. Limited authority online training can then be performed to improve system performance.

Once the system is successfully trained, the action network's weights are frozen, and the system can be tested. Typical and statistical time histories of the system when tested are shown for a flight at 30 kn in turbulence (Fig. 6) and a flight at 90 kn exposed to a step gust (Fig. 7). The typical time history was a randomly chosen simulation selected from the repertoire of test runs for that flight condition. The average error and error deviation reflect the Monte Carlo sample mean and sample variance over time when testing all of the successfully trained networks reflected in the Table 3 statistics for the specified flight condition. The results show that once the NDP controller has been successfully trained, it can reliably and consistently drive the system to its desired states.

Conclusions

This paper has taken NDP and successfully applied it to helicopter stabilization, a sophisticated and realistic control problem. Results presented here show the NDP controller able to stabilize successfully the Apache helicopter over a wide range of flight conditions and subject to various disturbances. Several new developments described in this paper have been made to extend NDP to the general MIMO control problem. The net result is that NDP shows promise as a viable control system design methodology, especially where model-free control system designs are required or where there is complex multi-axes coupling issues.

Acknowledgments

This research was supported by the National Science Foundation under Grants ECS-9553202 and ECS-0002098, and by Motorola. The first author was also supported by an academic leave provided by The Boeing Company.

References

[1]Omidvar, O., and Elliot, D. (eds.), *Neural Systems for Control*, Academic, San Diego, CA, 1997.

[2]Gupta, M., and Rao, D. (eds.), *Neuro-Control Systems Theory and Applications*, IEEE Press, New York, 1994.

[3]Vemuri, V. (ed.), *Artificial Neural Networks Concepts and Control Applications*, IEEE Computer Society Press, Los Alamitos, CA, 1992.

[4]Balakrishnan, S., and Biega, V., "Adaptive-Critic-Based Neural Networks for Aircraft Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 4, 1996, pp. 731–739.

[5]Prokhorov, D., Santiago, R., and Wunsch, D., II, "Adaptive Critic Designs: A Case Study for Neuro-Control," *Neural Networks*, Vol. 8, No. 9, 1995, pp. 1367–1372.

[6]Prokhorov, D., and Wunsch, D., II, "Adaptive Critic Designs," *IEEE Transactions on Neural Networks*, Vol. 8, No. 5, 1997, pp. 997–1007.

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

[8]Kirk, D., *Optimal Control Theory*, Prentice–Hall, Englewood Cliffs, NJ, 1970.

[9]McRuer, D., Ashkenas, I., and Graham, D., *Aircraft Dynamics and Automatic Control*, Princeton Univ. Press, Princeton, NJ, 1973.

[10]Bryson, A., *Control of Spacecraft and Aircraft*, Princeton Univ. Press, Princeton, NJ, 1994.

[11]Stevens, B., and Lewis, F., *Aircraft Control and Simulation*, Wiley, New York, 1992.

[12]Sobel, K., and Shapiro, E., "Application of Eigenstructure Assignment to Flight Control Design: Some Extensions," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 1, 1987, pp. 73–81.

[13]Ward, D., and Barron, R., "A Self-Designing Receding Horizon Optimal Flight Controller," *Proceedings of the 1995 American Control Conference*, Inst. of Electrical and Electronics Engineers, New York, 1995, pp. 3490–3494.

[14]Thukral, A., and Innocenti, M., "Controls Design Challenge: A Variable Structure Approach," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 5, 1994, pp. 942–949.

[15]Sadhukhan, D., and Feteih, S., "F8 Neurocontroller Based on Dynamic Inversion," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 1, 1996, pp. 150–156.

[16]Troudet, T., Garg, S., and Merrill, W., "Neural Network Application to Aircraft Control System Design," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA, Washington, DC, 1991, pp. 993–1009.

[17]Burgin, G., and Schnetzler, S., "Artificial Neural Networks in Flight Control and Flight Management Systems," *Proceedings of the IEEE National Aerospace and Electronics Conference*, Inst. of Electrical and Electronics Engineers, New York, 1990, pp. 567–573.

[18]McGrane, D., Smith, R., and Mears, M., "A Study of Neural Networks for Flight Control," *Proceedings of the 1994 American Control Conference*, Inst. of Electrical and Electronics Engineers, New York, 1994, pp. 2511–2515.

[19]Napolitano, M., and Kincheloe, M., "On-Line Learning Neural-Network Controllers for Autopilot Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 6, 1995, pp. 1008–1015.

[20]Kim, B., and Calise, A., "Nonlinear Flight Control Using Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 26–32.

[21]Kim, B., "Nonlinear Flight Control Using Neural Networks," Ph.D. Dissertation, Georgia Inst. of Technology, Atlanta, GA, Dec. 1993.

[22]Calise, A., "Neural Networks in Nonlinear Aircraft Flight Control," *Aerospace and Electronic Systems Magazine*, July 1996, pp. 5–10.

[23]Ha, C., "Neural Networks Approach to AIAA Control Design Challenge," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 4, 1995, pp. 731–739.

[24]Calise, A., "Neural Network Modeling of Flight Control System," *Aerospace and Electronic Systems Magazine*, Sept. 1998, pp. 27–29.

[25]Barto, A., Sutton, R., and Anderson, C., "Neuron Like Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, No. 5, 1983, pp. 834–847.

[26]Sutton, R., "Learning to Predict by the Methods of Temporal Difference," *Machine Learning*, Vol. 3, 1988, pp. 9–44.

[27]Werbos, P., "A Menu of Design for Reinforcement Learning over Time," *Neural Networks for Control*, edited by W. Miller III, R. Sutton, and P. Werbos, MIT Press, Cambridge, MA, 1990, Chap. 3.

[28]Werbos, P., "Neuro-Control and Supervised Learning: An Overview and Valuation," *Handbook of Intelligent Control*, edited by D. White and D. Sofge, Van Nostrand, New York, 1992, Chap. 3.

[29]Werbos, P., "Approximate Dynamic Programming for Real-Time Control and Neural Modeling," *Handbook of Intelligent Control*, D. White and D. Sofge, Van Nostrand, New York, 1992, Chap. 13.

[30]Watkins, C., and Dayan, P., "Q-Learning," *Machine Learning*, Vol. 8, No. 3–4, 1992, pp. 257–277.

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

[32]Enns, R., and Si, J., "Helicopter Flight Control Design Using a Learning Control Approach," *Proceedings of the 2000 IEEE Conference on Decision and Control*, Inst. of Electrical and Electronics Engineers, New York, 2000, pp. 1754–1759.

[33]Kumar, S., Harding, J., and Bass, S., "AH-64 Apache Engineering Simulation Non-Real Time Validation Manual," U.S. Army Aviation Systems Command, USAAVSCOM TR 90-A-010, Oct. 1990.