Helicopter Flight Control Design Using a Learning Control Approach[1]

Russell Enns and Jennie Si
Department of Electrical Engineering
Arizona State University
Tempe, AZ 85287-7606

Abstract

In this paper we introduce a new neural learning control mechanism for helicopter flight control design. The significance of our contribution is twofold. First neural dynamic programming (NDP) is in its early development stage and successful applications to date have been limited to simple systems, typically those possessing only a single control and a handful of states. With our industrial scale helicopter model, we consider a very realistic class of complex design problem. To accommodate such complex systems we introduce the concept of a trim network which is seamlessly integrated into our NDP control structure and is trained using our NDP control structure. Second, we introduce a new class of design methodologies to the helicopter control system design community. This approach is expected to be effective in dealing with real-time learning applications such as reconfigurable control. The paper consists of a comprehensive treatise of NDP and extensive simulation studies of NDP designs for controlling an Apache helicopter. All of our designs are tested using FLYRT, a sophisticated industry-scale non-linear validated model of the Apache helicopter. Though illustrated for helicopters, our NDP control system framework should be applicable to general control systems.

1 Introduction

While neural networks have been proposed for a variety of control systems, including flight controls, for well over the last decade, their application has been exclusively limited to low dimensional plants with only one one or two controls. A number of researchers have applied neural networks to the more specific domain of aircraft flight control but they too suffer from the same limitations, as well as others [2]. For example, Ha uses neural networks for flight control but the study is limited to lateral-directional control for a linear model [6]. Balakrishnan uses a form of reinforcement learning (adaptive critic based networks) for aircraft flight controls [3]. However, the research limits itself to the longitudinal axis and the system only has a single control.

This paper focuses on developing an NDP based control system to trim and stabilize a helicopter. In particular, we provide results showing how a neural network can be used to trim the helicopter to hover over a wide range of velocities. We also show how NDP can be used to stabilize the aircraft for five flight conditions, hover, 30, 60, 90 and 120 knots. Unlike many results which are based on linearized models and corresponding assumptions, our NDP designs and simulations are conducted using the FLYRT model. We thus are dealing with a very realistic system with nonlinearities, sensor and actuator dynamics, etc..

2 Neuro-Dynamic Programming for MIMO Rotorcraft Control

The objective of a neuro-dynamic programming (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. The historical development of this simulation-based dynamic programming technique is expounded on in [5, 7].

Figure 1 outlines the neuro-dynamic programming control structure. It consists of four blocks: two action networks (one for on-line fine tuning and the other for off-line training), a critic network, and a trim network. The action networks provide the required controls for a given system state. The critic network is used to provide an estimate/approximation of the cost function if an explicit cost function does not exist. A trim network is also used to provide nominal control positions as a function of the system's desired operating condition.

2.1 The Critic Network

A critic network is used to approximate a cost function should an explicit cost function not be convenient or

[1]Supported in part by NSF under grant ECS-9553202 and by EPRI-DOD under grant WO8333-01.

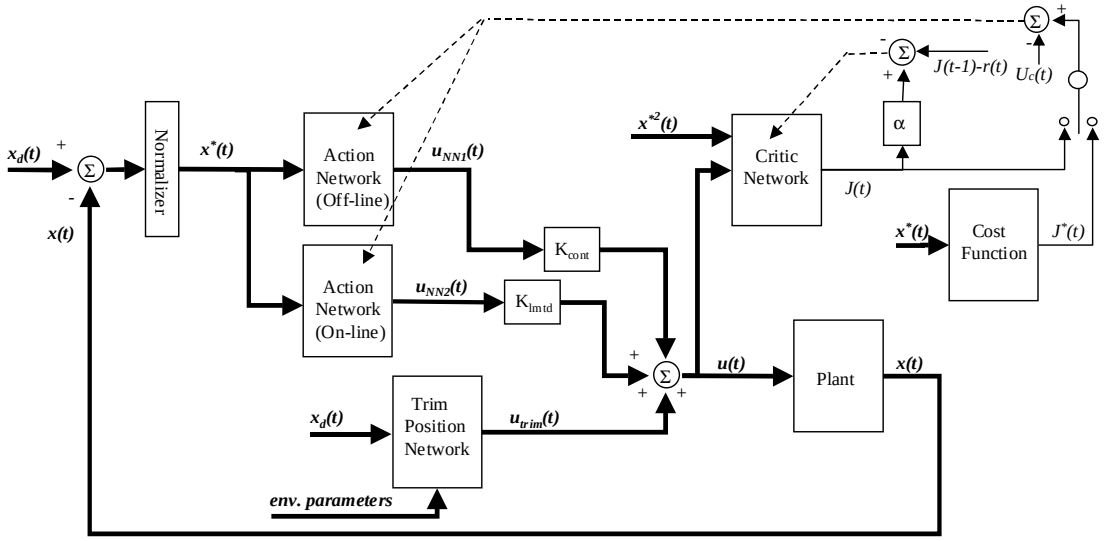

Figure 1: Neuro-dynamic Programming Based Controller.

possible to represent. For example, the network output $J(t)$ can approximate a cost function such as the discounted total reward-to-go,

$$R(t) = r(t+1) + \alpha r(t+2) + \alpha^2 r(t+3) + \cdots, \quad (1)$$

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 $r(t)$ was simply a binary reinforcement signal given by $r(t) = \{$ 0 if successful, -1 if failure $\}$ For the tracking problem at hand, we developed a more informative quadratic reinforcement signal $r(t) = -\sum_{i=1}^{n} \left(\frac{(x_i - x_{i,d})}{x_{i,max}}\right)^2$ where x_i is the i-th state of the state vector $\mathbf{x}$, $x_{i,d}$ is the desired reference state and $x_{i,max}$ is the nominal maximum state value.

The critic network can be implemented with a standard multi-layer feedforward neural network. The network can be linear or have a nonlinear sigmoid function depending on the complexity of the problem. We typically use a two layer weight neural network with sigmoid functions for the network nonlinearities. The critic network output, shown in Figure 1, is simply the scalar J.

The critic network is trained to minimize the following objective function

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

where

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

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

$$\Delta w_c(t) = \beta_c(t) \left[-\frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial w_c(t)} \right], \quad (4)$$

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

2.2 The Action Network

The action network generates the desired plant control given either measurements of the plant states or plant features from the feature extractor. As with the critic network, the action network can be implemented with a standard multi-layer linear or nonlinear feedforward neural network. For the action network the number of network outputs equals the control space dimension.

The principle in adapting the action network is to back-propagate 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 back-propagation is done through the critic network. For notational simplicity $J(t)$ represents either the actual or approximate cost function, depending on which is being used, for the remainder of the paper.

The weight updating in the action network adjusts the action network's weights to minimize the following objective function,

$$E_a(t) = \frac{1}{2} e_a^2(t), \quad (5)$$

where,

$$e_a(t) = J(t) - U_c(t). \quad (6)$$

The weights in the action network are then updated

according to

$$\Delta w_a(t) \quad = \quad \beta_a(t) \left[-\frac{\partial E_a(t)}{\partial w_a(t)} \right]. \qquad (7)$$

To take full advantage of the NDP design mechanism, we propose a second on-line action network. The second ANN performs on-line learning to adapt to local flight conditions while the first ANN's weights are frozen after having been trained under specific common flight conditions. The rationale is that the first ANN will provide a reasonable but suboptimal overall solution while the second ANN will adapt on-line to improve the controller's performance based on its experiences. The second ANN can be authority limited as required by the application.

2.3 The Trim Network

The 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.) Determining the control trim position is key to successful application of NDP to controlling general systems. Previous NDP control designs were successful because the systems that were tested (e.g. the inverted pendulum) had a zero trim requirement. Many flight control papers have assumed linear models in which case there is also a zero trim requirement since the linear model is linearized about a trim condition. However, in general trim requirements cannot be ignored for non-linear models, including those for aircraft.

A neural network based technique for trimming the helicopter at a specified operating condition is a second focus of this paper and is the subject of the following section. This technique can then be used to trim the helicopter over all desired flight conditions. The trim neural network shown in Figure 1 is then trained using the resulting trim data.

3 Trimming the Helicopter

Trimming is a technique of finding where the controls of the helicopter, as well as its orientation, need to be in order to minimize (typically zero) the helicopter's rotational and translational accelerations. 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 controller's integrators serving as the trim component. However, previously NDP based designs had not addressed the trim issue.

Trimming capability is also expected to be necessary for the reconfigurable flight control case. We contend that in order to achieve some form of control in the presence of actuator failures, one needs to determine the nominal control setting that minimizes the helicopter's accelerations despite the failure. Trimming is an open loop solution to this problem that should give us insight into forming a closed loop solution. However, we need a trim technique that minimizes the accelerations, as opposed to explicitly driving them to 0, as was done with FLYRT's trim method.

A third reason for a neural network based trim method is to facilitate realistic simulations. Often we want 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 only perform trim for a limited set of initial conditions. Thus a method of trimming the helicopter at an arbitrary specified initial condition is desired.

For these three reasons we propose a neural network based method for trimming the helicopter. This method of trimming, though applied to helicopters here, can be applied to any general physical control system. The major difference between existing helicopter trim methods and the proposed method is that existing methods take a aero-mechanical viewpoint rather than a control system viewpoint [8] [9]. That is, the states of interest that they trim (blade flapping angles, lag angles etc. and controls) are different than what we need to trim (body orientation and controls). Further, much of the existing work assumes knowledge of the equations governing the system. In our approach, no such assumption is made, instead a strictly numerical approach is taken. Further research may be warranted to determine if our approach is also applicable to the aero-mechanical trim problem.

3.1 Neural Network Based Trim Method

In this paper we propose a novel method for trimming a helicopter. The method takes a neural network approach that uses the existing NDP neural network framework and hence can be seamlessly integrated with the NDP controller for a very simple and efficient implementation. The neural network based method that we present is analogous to the shooting method, however, instead of using Newton's method the neural network uses the less sophisticated gradient descent method. This method appears to have better convergence properties at the expense of convergence time which is not an issue for us. Results shown later indicate that this new method is very robust and accurate.

The neural network structure used to determine the aircraft's trim position is shown in Figure 2. The underlying premise is that the neural network's biases (in its last layer) can be used as control position trims. In fact, by using the NDP controller's action neural net-

work with the inputs to the network zeroed, we can train the network biases to provide an open loop control trim solution. The network biases, b, which when trained are equivalent to the control trim u_{trim}, are updated using the same gradient descent training method described for NDP. That is $u_{trim} = b$ where the biases are trained via

$$b(m+1) = b(m) + \Delta b, \quad (8)$$

$$\Delta b = \lambda \left[-\frac{\partial E_a(t)}{\partial b(t)} \right], \quad (9)$$

where

$$\frac{\partial E_a(t)}{\partial b(t)} = \frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial x(t)} \frac{\partial x(t)}{\partial u(t)} \frac{\partial u(t)}{\partial b(t)}. \quad (10)$$

The cost function used is $J = 1/2 \bar{x}^T \bar{x}$ where $\bar{x}_i = \frac{(x_i - x_{i,d})}{x_{i,max}}$ and $x \equiv [p, q, r, u, v, w]$.

Instead of updating the biases at every time step as done for the weights in NDP, the biases are updated iteratively at the end of every simulation. M iterations are performed where in each iteration, m, the plant is initialized to the latest trim position given by the neural network, the plant's dynamics are simulated for a nominal time (e.g. 500 ms), and the biases are then updated according to the above equations.

Define the vector $\boldsymbol{\theta} \equiv [\theta, \phi, \psi]$. We can determine the aircraft trim angle vector $\boldsymbol{\theta}_{trim}$ using the same method described for determining the trim controls, as indicated by the dotted line in Figure 2. That is,

$$\boldsymbol{\theta}_{trim}(m+1) = \boldsymbol{\theta}_{trim}(m) + \Delta \boldsymbol{\theta}(m)_{trim}, \quad (11)$$

$$\Delta \boldsymbol{\theta}_{trim} = \lambda \left[-\frac{\partial E_a(t)}{\partial \boldsymbol{\theta}(t)} \right], \quad (12)$$

where

$$\frac{\partial E_a(t)}{\partial \boldsymbol{\theta}(t)} = \frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial x(t)} \frac{\partial x(t)}{\partial \boldsymbol{\theta}(t)}. \quad (13)$$

The first two partials are evaluated in the manner described earlier while the partial $\frac{\partial x(t)}{\partial \boldsymbol{\theta}(t)}$ can be calculated numerically via perturbations to the system. Alternatively, it 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 partial at hover only and used it for all flight conditions to generate our results.

An advantage of this neural network based method is that no assumptions are made about necessarily driving system accelerations to zero as is the case with trimming methods such as that developed for FLYRT. Rather, this method minimizes the change in states (velocities) over a specified time period. This is equivalent to minimizing accelerations but not necessarily driving them to zero.

4 The Helicopter Model

NDP is tested using a helicopter model, run at 50 Hz, that consists of three parts: an actuator model, an actuator to blade geometry model, and FLYRT. The inputs to the model are the three main rotor actuator positions and the tail rotor actuator position. The outputs from model are numerous; for flight control purposes they are limited to the aircraft's translational (u, v, w) and rotational (p, q, r) velocities and the aircraft's orientation (θ, ϕ, ψ) for a total of 9 states.

At the heart of the helicopter model is FLYRT, a sophisticated non-linear flight simulation model of the Apache helicopter developed by Boeing over the past two decades [1]. FLYRT models all the forces and moments acting on the helicopter. The rotor can be modeled using either a blade element model or a hybrid quasi-static rotormap combined with a closed form analytic solution to account for the transient effects. The rotormap outputs six states, the rotor disk loading coefficient, shaft torque, in plane forces, and the blade flapping angles as a function of three parameters in the control plane, collective pitch, advance ratio and inflow ratio.

The model 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.

FLYRT also possesses a trimming routine to trim the helicopter if NDP is not used to do so. In addition, FLYRT features a linear model generator to generate linear models of the helicopter at specified operating conditions. Such models can be used for analysis or other purposes.

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. The mechanical interface between the actuators and the helicopter blades is also modeled.

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 FS/WL/BL coordinate frame [1].

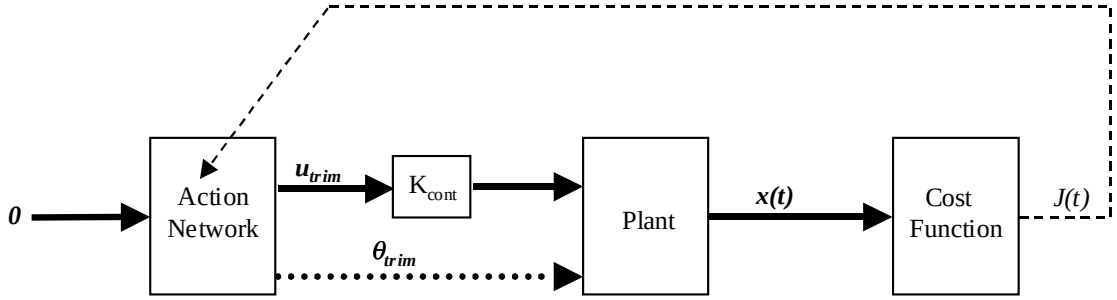

Figure 2: Neural Network Structure for Determining Trim

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

Table 1: Helicopter Operating Conditions

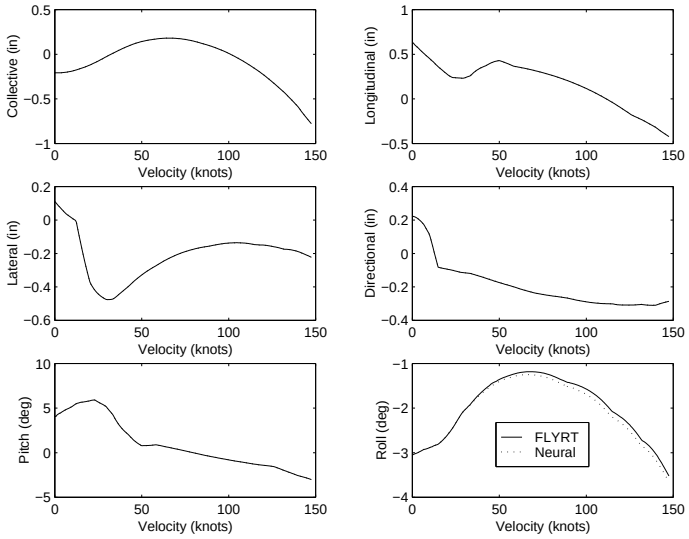

Figure 3: Trim Positions as a Function of Airspeed For Neural and FLYRT Methods

5 Results

This section presents results showing the performance of the neural network based trim method and the performance of the neuro-dynamic programming controller in stabilizing the Apache helicopter. The trim method worked very well for the Apache helicopter. Figure 3 compares the NDP generated trim positions to those generated by FLYRT over the range of 0 to 150 knots forward speed. In fact, the results appear to be identical except for roll which has a difference of less than 0.15^o.

The performance of NDP as a helicopter stabilizer is summarized statistically for five cases: hover, 30, 60, 90 and 120 knots. In addition, plots of the average state error and error deviation as well as a typical time history plot are provided for the 30 knot case. Plots for the other cases 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 statistical success of the NDP controller's ability to learn to stabilize the helicopter is evaluated for each of the five flight conditions. 100 runs were performed to evaluate NDP's performance. Each run consists of up to 500 attempts (trials) to learn how to successfully control the system. An attempt is deemed successful if the helicopter stays within prescribed failure criteria bounds for the entire flight duration (5 minutes) [7].

The statistical training results for the five flight conditions are shown in Tables 2. Case A shows the statistics when the aircraft's initial conditions are the nominal trim positions. Case B shows the statistics when the aircraft's initial conditions have a Gaussian distribution about the nominal trim position. The one sigma variations are 2 ft/s for the velocities, 2^o for the aircraft angles, and $3^o/s$ for the aircraft angular rates. The statistical average error and error deviation for the successful runs for the 30 knot case are shown in Figure 4.

The neural network parameters used during training are provided in Table 3. Note that these parameters were chosen based on experience but were not tuned to optimize the results.

Parameter	Value
Initial ANN learning rate	0.1
Final ANN learning rate	0.1
Initial CNN learning rate	0.1
Final CNN learning rate	0.01
Max ANN internal cycles	200
Max CNN internal cycles	100
Number of hidden nodes	6
Control scaling factor	5.0

Table 3: Neural Network Parameter Values

	Condition	Hover	30 knots	60 knots	90 knots	120 knots
Case A	Percentage of Successful Runs	90%	86%	96%	95%	93%
	Average No. of Trials	84	129	107	134	135
Case B	Percentage of Successful Runs	83%	97%	88%	89%	66%
	Average No. of Trials	147	88	100	131	128

Table 2: Learning Statistics for NDP Control of the Apache Helicopter

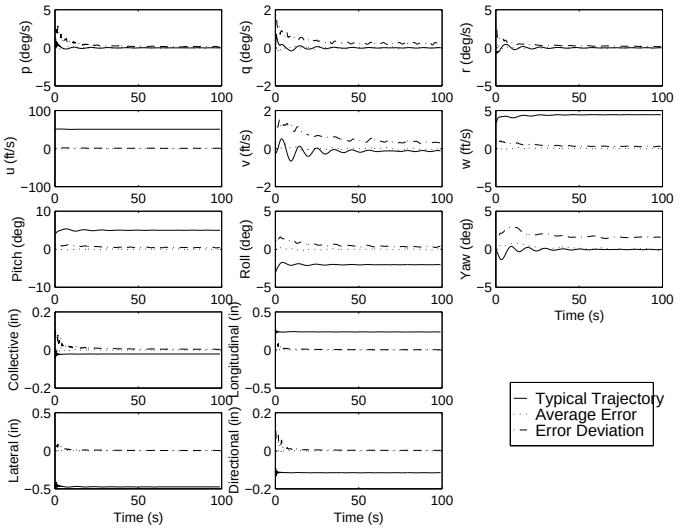

Figure 4: Statistical and Typical State and Control Trajectories of NDP Stabilization at 30 kts.

An important observation can be made about the results in Table 2. First, 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 apriori system knowledge. The ramification is that this training is done off-line (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 Figure 1 can be implemented in a helicopter. Limited authority on-line training can then be performed to improve system performance.

6 Conclusions

This paper has advanced neural-dynamic programming control research by developing a comprehensive control structure using NDP as its basic element. Included in this development was a novel method for trimming a system using a neural network based approach. An industrial scale nonlinear validated model of the Apache helicopter was used to test the controller. Our results show the NDP controller able to successfully trim and stabilize the Apache helicopter over a wide range of flight conditions. Future research will extend our NDP control system to address both command tracking and reconfigurable flight control issues.

References

[1] McDonnell Douglas Helicopter Company, *AH-64 Apache engineering simulation program documentation (FLYRT - Revision 5.0)*, April 1989.

[2] B. Kim and A. Calise, "Nonlinear flight control usig neural networks", *AIAA Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 26-32, Jan.-Feb. 1997.

[3] S. Balakrishnan and V. Biega, "Adaptive-critic-based neural networks for aircraft optimal control", *AIAA Journal of Guidance, Control, and Dynamics*, vol. 19, no. 4, pp. 731-739, Jul.-Aug. 1996.

[4] D. Kirk, *Optimal Control Theory*, Englewood Cliffs, NJ: Prentice-Hall Inc., 1970.

[5] J. Si and J. Wang "On-line learning by association and reinforcement", *IEEE Transactions on Neural Networks*, to appear.

[6] C. Ha, "Neural networks approach to AIAA control design challenge", *AIAA Journal of Guidance, Control, and Dynamics*, vol. 18, no. 4, pp. 731-739, Jul.-Aug. 1995.

[7] R. Enns and J. Si, "Neuro-dynamic programming applied to helicopter flight control", *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Denver, CO, August 15-17, 2000.

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

[9] D. Peters and D. Barwey, "General theory of rotorcraft trim", *Proceedings of the AIAA/ASME/ASCE/AHS Structures, Structural Dynamics & Materials Conference*, New Orleans, LA, April 10-13, 1995, pp. 2558-2590.