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

Real-Time Adaptation of Mode Transition Controllers

Freeman Rufus,* George Vachtsevanos,† and Bonnie Heck‡
Georgia Institute of Technology, Atlanta, Georgia 30332-0250

A real-time adaptation scheme is proposed for the online customization of mode transition controllers designed offline via blending local mode controllers. It consists of the desired transition trajectory model, the active plant model, and the mode transition controller. The active plant model, which incorporates local mode information, is initially trained offline to capture the desired transition trajectory and controls. Afterward, the active plant model is adapted online via structure and parameter learning to capture the input/output relationship of the nonlinear system to be controlled. Likewise, the blending gains portion of the mode transition controller is determined offline and is adapted online via structure and parameter learning to track the desired transition trajectory. The control sensitivity matrix and the one-step-ahead predicted output of the controlled system are used to develop the desired blending gains of the mode transition. The control sensitivity matrix and the predicted output are determined from the active plant model. The proposed adaptation scheme is illustrated for a hover-to-forward-flight mode transition control of a helicopter encountering parametric changes and wind disturbances.

I. Introduction

COMPLEX large-scale systems, such as unmanned aerial vehicles and industrial processes, must possess the intelligence required to behave in an autonomous manner under uncertain environmental conditions. Typically, these systems are required to operate in a finite number of operational modes that necessitate robust, stable, and smooth transitions between them. A (local) operational mode (or mode of operation) is considered to be a small region about an equilibrium point in which the system exhibits quasi-steady-state behavior. A mode transition (or mode-to-mode) controller refers to a controller that transitions a system from a start mode of operation to the goal mode. The problem of transitioning between two operational modes can be addressed by traditional control techniques, such as gain scheduling, sliding mode control, and adaptive control, when a desired trajectory is given.

Although there is no consistent theory that deals with dynamic transitions between various equilibria, gain scheduling has been used to design equilibrium-to-equilibrium controllers. The technique of gain scheduling constructs a nonlinear controller by combining the members of an appropriate family of linear controllers. In conventional gain scheduling, the transition between equilibria is governed normally by an auxiliary scheduling variable,[1,2] which should vary slowly with respect to the states. The disadvantages associated with gain scheduling include a reliance on a long trial-and-error design process, a lack of adaptability to online variations, and poor robustness to uncertainties.[3] The gain-scheduling procedure is generally as follows[4–7]: 1) parameterize the equilibrium operating points of the plant by a scheduling variable that involves some of the plant states; 2) for a family of equilibrium operating points parameterized by a scheduling variable, linear models of the plant are created and linear controllers are obtained for each linear model; and 3) an interpolation technique is used to interpolate between the linear controller gains for the $\text{equilibrium}_{\text{start}}$ to $\text{equilibrium}_{\text{goal}}$ transition. Although gain-scheduled controllers are typically designed using plant linearizations at a number of equilibrium operating points, it is possible to apply gain scheduling to control design of linear time-varying systems obtained via linearizations relative to a trajectory.[8–11] To overcome the restriction to near-equilibrium operation in traditional gain scheduling, a velocity-based gain-scheduling controller design has been developed.[12,13] This method uses plant dynamics at equilibrium and nonequilibrium operating points, which may lead to controller realizations that achieve better performance than classical gain-scheduling controllers.

Similar to gain scheduling, sliding mode control (SMC) uses more than one control law and is, in general, nonlinear. The performance index is specified as a manifold of space called the *sliding surface*. A sliding mode controller sends the system states onto the sliding surface and keeps them there. However, as a result of high control gain, SMC systems can suffer from the effects of actuator chattering due to switching and imperfect implementations.[14] When using SMC to track a desired trajectory, it may become possible for the closed-loop system to become unstable if the sliding surface changes faster than SMC can follow it. In Ref. 15, a slew-limiting prefilter is proposed to preprocess the reference trajectory such that the sliding surface does not drift too quickly for stable control. In Ref. 16, an SMC scheme is developed that modifies the reference trajectory to comply with actuator limitations and suppression of residual oscillations at the end of motion while competing against plant uncertainties. To overcome the disadvantages of the SMC systems, the fuzzy sliding mode control (FSMC) has been introduced to provide better damping and reduced chattering.[14] FSMC used for motion trajectory control is described in Ref. 17. In Ref. 18, FSMC was used with a fuzzy logic controller to track a prespecified position-velocity trajectory for an uncertain nonlinear system.

The basic objective of adaptive control is to maintain consistent performance of a system in the presence of uncertainty or unknown variation in plant parameters. Typically, adaptive control is developed for multi-input/multi-output (MIMO) linear systems, single input/single output (SISO) nonlinear systems, and certain classes of MIMO nonlinear systems. In Refs. 19 and 20, the development of adaptive controllers for a class of feedback-linearizable nonlinear systems are described. Masino and Tomei have proposed an adaptive output feedback tracking control for a class of SISO nonlinear systems with uncertain differentiable time-varying parameters.[21] Recently, adaptive techniques based on the one-step-ahead control strategy have been developed for more general nonlinear systems. Ma and Loh have proposed neural network-based one-step-ahead control strategies for a class of nonlinear SISO systems.[22,23] Tan and Cauwenberghe proposed a nonlinear one-step-ahead control scheme based on a recurrent neural network model for nonlinear SISO processes.[24] The neural network model was trained via a recursive least-squares (RLS) algorithm, and the gradient descent update rate for the control law was determined by stability considerations. For general linear dynamical systems, a multivariable one-step-ahead adaptive control scheme was proposed by Song and Hardt.[25] The adaptive scheme was applied to an arc-welding process in which the

Received 27 December 1999; revision received 27 February 2001; accepted for publication 28 February 2001. Copyright © 2001 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0731-5090/02 $10.00 in correspondence with the CCC.

*Graduate Research Assistant, School of Electrical and Computer Engineering.

†Professor, School of Electrical and Computer Engineering; george.vachtsevanos@ee.gatech.edu.

‡Associate Professor, School of Electrical and Computer Engineering.

process parameters were estimated using RLS. Finally, for a general class of MISO nonlinear systems, an adaptive quasi-one-step-ahead control law was proposed by Mingzhong and Fuli.[26] The control law was derived using the sensitivity between the controlled system input and output and the quasi-one-step-ahead predictive output. The sensitivity of the plant was estimated using RLS, and the predicted output was obtained by a recurrent neural network.

In this paper, an adaptation scheme is proposed for the real-time adaptation of mode transition controllers designed via blending local mode controllers (BLMCs). The control objective of the adaptation scheme is to adapt the blending gains portion of the mode transition controllers such that the nonlinear plant state vector tracks the desired transition trajectory from a start mode of operation to a goal mode. The adaptation scheme is composed of a desired transition model, an active plant model, and an active controller model, which is the mode transition controller. The desired transition model, the active plant model, and the blending gains portion of the active controller model are represented via a fuzzy neural network construct.[27] All three fuzzy neural models are trained offline, and the latter two models are adapted online. The active plant model is adapted via structure and parameter learning to capture the input/output behavior of the nonlinear system to be controlled. The new blending gains to be developed by mode transition controller are determined from the control sensitivity matrix and predicted output of the active plant model. The parameter learning for the active plant and active controller models uses local least-squares estimation.

In Sec. II, the fuzzy neural network construct used in the paper is described briefly. An overview of the systematic design of mode transition controllers is given in Sec. III, and the online adaptation of mode transition controllers is described in Sec. IV. Finally, in Sec. V, the real-time adaptation scheme proposed in this paper is illustrated for the hover-to-forward-flight transition of a helicopter encountering parametric changes and wind disturbances.

II. Fuzzy Neural Networks

Recently, fuzzy neural networks (FNNs) have been used for implementing fuzzy logic systems. FNNs combine the low-level learning and computational power of neural networks with high-level reasoning and decision making of fuzzy systems.[27] The FNN model of a desired input/output relation requires parameter learning and possibly structure learning. Structure learning relates to the choice of the number of fuzzy input partitions and the development of the fuzzy rule base; parameter learning refers to the adjustment of the network weights.[27] The FNN constructs can be classified into two categories: 1) fuzzy neural structures based on the Takagi–Sugeno inference method with crisp consequent functions,[28,29] and 2) fuzzy neural structures of fuzzy rules and fuzzy consequents.[29] The most familiar member of the first category is the ANFIS architecture, which employs a fixed fuzzy rule base while performing only parameter learning.[30,31] A significant member of the second category uses both structure and parameter learning.[30]

The fuzzy neural structure proposed by Theocharis and Vachtsevanos[27] is considered in this section. This FNN structure consists of a fuzzy rule base of Takagi–Sugeno fuzzy rules, with the rule consequents being linear polynomials of the input premise variables. Both structure learning and parameter learning are used to adaptively develop the FNN construct. The structure learning inserts new membership functions, creates new fuzzy rules, and selects initial parameters of the new rules on the basis of the desired output data. The parameter learning updates the mean and deviation of gaussian membership functions and the consequent weights via the back propagation algorithm.

A. Structure

The fuzzy neural architecture proposed in Ref. 27 is divided into the premise part, the consequent part, and the defuzzification part, as shown in Fig. 1. The premise module partitions the premise space, assigns membership functions to each premise cell, and develops the rule base of fuzzy rules. The consequent module consists of the rule consequents being linear polynomials of the input premise variables. Finally, the defuzzification module combines the firing strengths of the rules and the rule output functions to provide the final system output. Therefore, this FNN construct realizes the fuzzification, fuzzy reasoning, and defuzzification functionalities of a connectional fuzzy inference mechanism.

Let $\boldsymbol{x} = [x_1, \ldots, x_m]^T$ and $\boldsymbol{y} = [y_1, \ldots, y_p]^T$ denote the input and output vectors of the FNN, respectively. The fuzzy rule base of the FNN consists of a collection of N fuzzy rules of the form:

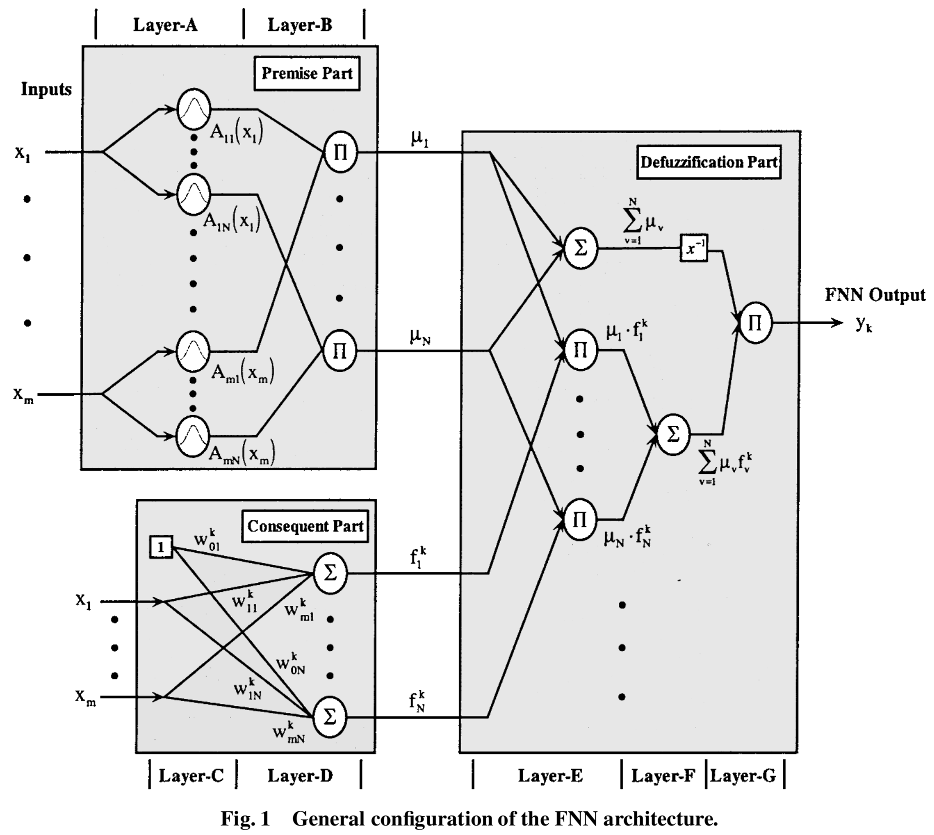

Fig. 1 General configuration of the FNN architecture.

$R^{(j)}$: IF x_1 is A_{1j} AND x_2 is A_{2j} AND $\cdots$ AND x_m is A_{mj} THEN

$$f_j^1 = w_{0j}^1 + w_{1j}^1 x_1 + \cdots + w_{mj}^1 x_m \text{ AND } \cdots \text{ AND}$$
$$f_j^p = w_{0j}^p + w_{1j}^p x_1 + \cdots + w_{mj}^p x_m \tag{1}$$

where f_j^ℓ denotes the jth rule output associated with the ℓth output component y_ℓ. $w_{0j}^\ell, \ldots, w_{ij}^\ell, \ldots, w_{mj}^\ell$ are the polynomial coefficients linearly connecting the input variables to the f_j^ℓ consequent function. Finally, $A_{1j}, \ldots, A_{ij}, \ldots, A_{mj}$ are labels of the fuzzy sets in the premise space associated with the jth rule $R^{(j)}$.

Each linguistic label A_{ij} is associated with a gaussian membership function, $\mu_{A_{ij}}(x_i)$, which specifies the degree to which a given x_i satisfies the quantifier A_{ij}:

$$\mu_{A_{ij}}(x_i) = \exp\left[-\frac{1}{2}\frac{(x_i - m_{ij})^2}{\sigma_{ij}^2}\right] \tag{2}$$

where m_{ij} and σ_{ij} denote the mean and standard deviation of the gaussian membership function. The degree of fulfillment (or the firing strength) of each rule $R^{(j)}$ is taken as

$$\mu_j(x_1, \ldots, x_m) = \mu_{A_{1j}}(x_1) \times \cdots \times \mu_{A_{mj}}(x_m) \tag{3}$$

Given an input vector $\boldsymbol{x}$, the ℓth output component y_ℓ of the fuzzy system is inferred as follows:

$$y_\ell = \frac{\sum_{j=1}^N \mu_j \cdot f_j^\ell}{\sum_{j=1}^N \mu_j}, \qquad \ell = 1, \ldots, p \tag{4}$$

The inferred outputs of Eq. (4) result from the application of the weighted-average defuzzification method.

B. Learning Algorithm Using no Local Model Information

The learning algorithm of this section uses both structure learning and parameter learning to determine the premise part structure and the appropriate premise/consequent parameters of a fuzzy neural network. Structure learning determines the appropriate fuzzy neural structure by performing membership function insertion and parameter setting of initial premise/consequent parameters of newly established rules. Parameter learning updates the mean and deviation of gaussian membership functions and the consequent weights via the back propagation algorithm. Details of the structure learning and parameter learning algorithms can found in Ref. 27.

C. Learning Algorithm Using Local Model Information

The learning algorithm of this section also uses both structure learning and parameter learning to determine the premise part structure and the appropriate premise/consequent parameters of an FNN. Structure learning determines the appropriate fuzzy neural structure by performing membership function insertion and parameter setting of initial premise/consequent parameters of newly established rules. Parameter learning updates consequent weights via a local least-squares estimation technique.[32]

1. *Structure Learning with Local Model Information*

Structure learning using local model information is exactly same as the structure learning scheme described in Ref. 9, except that the new rules' consequent weights are initialized in the following way:

$$\left[\begin{array}{c|ccc} w_{0,N+1}^1 & w_{1,N+1}^1 & \cdots & w_{m,N+1}^1 \\ \vdots & \vdots & \vdots & \vdots \\ w_{0,N+1}^p & w_{1,N+1}^p & \cdots & w_{m,N+1}^p \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{y}(\boldsymbol{t}_{k+1}) - \dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\boldsymbol{x}(\boldsymbol{t}_{k+1}) & \dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \end{array}\right] \tag{5}$$

where $\boldsymbol{x}(t_{k+1})$ is the model input at t_{k+1}, $\boldsymbol{y}(t_{k+1})$ is the desired output corresponding to $\boldsymbol{x}(t_{k+1})$, and $\partial\boldsymbol{y}/\partial\boldsymbol{x}$ is the Jacobian matrix defined at $\boldsymbol{x}(t_{k+1})$.

2. *Parameter Learning via Local Least-Squares Estimation*

Consider a training dataset S_M comprising M entries of input/output pairs of the form: $S_M = \{[\boldsymbol{x}^d(t_k), \boldsymbol{y}^d(t_k)], k = 1, \ldots, M\}$, where $\boldsymbol{x}^d(t_k) = [x_1^d(t_k), \ldots, x_m^d(t_k)]^T$ and $\boldsymbol{y}^d(t_k) = [y_1^d(t_k), \ldots, y_p^d(t_k)]^T$ denote the desired input and output vectors of the FNN model, respectively. Let S_{M_j} denote the local training set composed of input/ output pairs of S_M limited to the receptive field of the jth rule: $S_{M_j} = \{(\boldsymbol{x}^d, \boldsymbol{y}^d) \in S_M, \mu_j(\boldsymbol{x}^d) \geq \mu_{\min}\}$, where $\mu_{\min}$ is the rule firing-strength threshold. Parameter learning involves the computation of N locally weighted least-squares regressions, one for each rule, using only the training data within the rule's receptive field. The consequent weights are not updated for rules having zero training data within the receptive field. For $\{\boldsymbol{y}^d(t_k), \boldsymbol{x}^d(t_k)\} \in S_{M_j}$, we have the following equations:

$$\boldsymbol{y}(t_k) = [y_1(t_k) \quad \cdots \quad y_p(t_k)]^T = \boldsymbol{W}_j \begin{bmatrix} 1 \\ \boldsymbol{x}^d(t_k) \end{bmatrix}$$
$$= \begin{bmatrix} \bar{\boldsymbol{y}}^j & \boldsymbol{W}_j^b \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{x}^d(t_k) - \bar{\boldsymbol{x}}^j \end{bmatrix} \tag{6a}$$

$$\boldsymbol{y}^d(t_k) - \boldsymbol{y}(t_k) = \begin{bmatrix} \Delta\bar{\boldsymbol{y}}^j & \Delta\boldsymbol{W}_j^b \end{bmatrix}_k \begin{bmatrix} 1 \\ \boldsymbol{x}^d(t_k) - \bar{\boldsymbol{x}}^j \end{bmatrix} \tag{6b}$$

$$\boldsymbol{W}_j = \left[\begin{array}{c|ccc} w_{0j}^1 & w_{1j}^1 & \cdots & w_{\mathrm{mj}}^1 \\ \vdots & \vdots & \vdots & \vdots \\ w_{0j}^p & w_{1j}^p & \cdots & w_{\mathrm{mj}}^p \end{array}\right] = \left[\boldsymbol{W}_j^a \mid \boldsymbol{W}_j^b\right] \tag{6c}$$

$$\boldsymbol{W}_j^a = \bar{\boldsymbol{y}}^j - \boldsymbol{W}_j^b \bar{\boldsymbol{x}}^j \tag{6d}$$

where $\boldsymbol{W}_j$ is the consequent parameters of the jth rule, $\bar{\boldsymbol{x}}^j$ is the center of the jth rule such that $\mu_j(\bar{\boldsymbol{x}}^j) = 1$, $\bar{\boldsymbol{y}}^j$ is the consequent output of the jth given the input $\bar{x}^j$, and $k = 1, \ldots, M_j$. Let $\boldsymbol{Y} = [\boldsymbol{y}(t_1) \ \cdots \ \boldsymbol{y}(t_{M_j})]^T$ and $\boldsymbol{Y}^d = [\boldsymbol{y}^d(t_1) \ \cdots \ y^d(t_{M_j})]^T$. Determine $\Delta\boldsymbol{V}_j = [\Delta\bar{\boldsymbol{y}}^j \ \ \Delta\boldsymbol{W}_j^b]^T$ such that it minimizes the following weighted least-squares performance index for the jth rule:

$$\boldsymbol{P}_j = \left[(\boldsymbol{Y}^d - \boldsymbol{Y}) - \boldsymbol{\Phi}_j \cdot \Delta\boldsymbol{V}_j\right]^T \boldsymbol{Q}_j \left[(\boldsymbol{Y}^d - \boldsymbol{Y}) - \boldsymbol{\Phi}_j \cdot \Delta\boldsymbol{V}_j\right] \tag{7}$$

where

$$\boldsymbol{Q}_j = \operatorname{diag}\left\{\rho_j\left[\boldsymbol{x}^d(t_1)\right], \ldots, \rho_j\left[\boldsymbol{x}^d\left(t_{M_j}\right)\right]\right\}$$

$$\boldsymbol{\Phi}_j = \begin{bmatrix} 1 & \cdots & 1 \\ \boldsymbol{x}^d(t_1) & & \boldsymbol{x}^d(t_{M_j}) \end{bmatrix}^T$$

$$\rho_j\left[\boldsymbol{x}^d(t_k)\right] = \frac{\mu_j\left[\boldsymbol{x}^d(t_k)\right]}{\sum_{v=1}^{M_j} \mu_j\left[\boldsymbol{x}^d(t_v)\right]}, \qquad K = 1, \ldots, M_j$$

Therefore,

$$\frac{\partial P_j}{\partial\{\Delta\boldsymbol{V}_j\}} = \boldsymbol{0} = -\left[(\boldsymbol{Y}^d - \boldsymbol{Y}) - \boldsymbol{\Phi}_J \cdot \Delta\boldsymbol{V}_j\right]^T \boldsymbol{Q}_j \boldsymbol{\Phi}_j$$

$$\Rightarrow \quad \Delta\boldsymbol{V}_j = \left[\Delta\bar{\boldsymbol{y}}^j \quad \Delta\boldsymbol{W}_j^b\right]^T = \left(\boldsymbol{\Phi}_j^T \boldsymbol{Q}_j \boldsymbol{\Phi}_j\right)^{-1} \boldsymbol{\Phi}_j^T \boldsymbol{Q}_j (\boldsymbol{Y}^d - \boldsymbol{Y}) \tag{8}$$

The consequent parameters for the jth rule are updated using the following equation:

$$\boldsymbol{W}_j(t+1) = \boldsymbol{W}_j(t) + \Delta\boldsymbol{W}_j(t) \tag{9}$$

where $\Delta\boldsymbol{W}_j = [\Delta\boldsymbol{W}_j^a \mid \Delta\boldsymbol{W}_j^b]$, $\Delta\boldsymbol{W}_j^a = \Delta\bar{\boldsymbol{y}}^j - \Delta\boldsymbol{W}_j^b \bar{\boldsymbol{x}}^j$ and the terms $\Delta\bar{\boldsymbol{y}}^j$ and $\Delta\boldsymbol{W}_j^b$ are determined from Eq. (8).

3. *Learning Procedure*

1) Use a subset of the training dataset to perform network initialization. This network initialization is conducted offline using structure learning.
2) Perform structure learning if necessary.
3) Perform parameter learning.
4) Repeat steps 2 and 3 for all training data entries.

D. FNN Linear Incremental Model

Let $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$ and $\boldsymbol{y} = [y_1, y_2, \ldots, y_p]^T$ denote the input and output vectors, respectively, of the FNN model depicted in Fig. 1. Suppose that the FNN model has N fuzzy rules. Then, the rsth element of the incremental model of the FNN model is

$$\left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right]_{rs} = \frac{\partial y_r}{\partial x_s} = \sum_{v=1}^{N}\left(\frac{\partial y_r}{\partial \mu_v}\frac{\partial \mu_v}{\partial x_s} + \frac{\partial y_r}{\partial f_v^r}\frac{\partial f_v^r}{\partial x_s}\right) \tag{10}$$

where $r = 1, \ldots, p$ and $s = 1, \ldots, m$; μ_v denotes the firing strength of the vth fuzzy rule, and f_v^r denotes the rule consequent functions for the rth output and the vth fuzzy rule. Therefore, the rsth element of the incremental model of the FNN model is given by

$$\left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right]_{rs} = \frac{\sum_{v=1}^{N}\mu_v\left[\left(y_r - f_v^r\right)(x_s - m_{sv})/\sigma_{sv}^2 + w_{sv}^r\right]}{\sum_{\ell}^{N}\mu_\ell} \tag{11}$$

where

$$\frac{\partial y_r}{\partial y_v} = \frac{f_v^r - y_r}{\sum_{\ell=1}^{N}\mu_\ell}, \qquad \frac{\partial \mu_v}{\partial x_s} = -\mu_v \cdot \frac{(x_s - m_{sv})}{\sigma_{sv}^2}$$

$$\frac{\partial y_r}{\partial f_v^r} = \frac{\mu_v}{\sum_{\ell=1}^{N}\mu_\ell}, \qquad \frac{\partial f_v^r}{\partial x_s} = w_{sv}^r$$

μ_v denotes the firing strength of the vth fuzzy rule, f_v^r denotes the rule consequent functions for the rth output and the vth fuzzy rule, y_r is the rth component of the output vector, x_s is the sth component of the input vector, m_{sv} and σ_{sv} denote the mean and standard deviation parameters of the gaussian membership function $\mu_{A_{sv}}$, and w_{sv}^r is the consequent parameter corresponding to sth input component, rth output component, and the vth fuzzy rule.

III. Offline Design of Mode Transition Controllers

A. Mode Transition Problem

Given a large-scale dynamical system represented by the following state equation:

$$\dot{x} = \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{x} \in R^n, \boldsymbol{u} \in R^m \tag{12}$$

it is assumed that the system is composed of N_s subsystems S_i, $i = 1, 2, \ldots, N_s$, where each subsystem represents an operational mode of the system. The state equation for the ith subsystem is

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{u}_i), \qquad \boldsymbol{x}_i \in R^{n_i}, \boldsymbol{u}_i \in R^{m_i} \tag{13}$$

Let mode$_p$ and mode$_q$ denote the pth and qth subsystem, respectively. How do we design a controller that stably and smoothly transitions a system from mode$_p$ to mode$_q$?

B. Offline Control Design

An offline design methodology known as the BLMC approach was developed by Rufus et al. to design mode transition controllers.[33] This approach for designing mode transition controllers used the aggregated states of the start and goal modes, and the output vector of the mode transition controller was determined by blending the individual output vector of the start and goal mode controllers. The following is an outline of the BLMC approach:

1) Design regulators for the start and goal modes such that initial states are driven to the equilibrium of the respective modes.

2) Model the dynamics that correspond to the aggregated states and controls of the start and goal mode so that a transitional path from the start mode to the goal mode can be determined.

3) Determine an optimal transitional path from the equilibrium state of the start mode to the equilibrium state of the goal mode by solving a nonlinear optimal control problem.

4) Determine the desired blending gains using the desired state and control trajectory determined from step 3.

5) Realize the blending gains via an FNN construct.[27]

The structure of a mode transition controller designed via the BLMC method is shown Fig. 2, where $\boldsymbol{x}_{pq}$ is the aggregated state vector of $\boldsymbol{x}_p$ and $\boldsymbol{x}_q$; $\boldsymbol{x}_{pq}$ is the aggregated control vector of $\boldsymbol{u}_p$ and $\boldsymbol{u}_q$; $\boldsymbol{x}_p$ and $\boldsymbol{x}_q$ denote the state vectors of mode$_p$ and mode$_q$, respectively; $\boldsymbol{u}_p$ and $\boldsymbol{u}_q$ denote the control input vectors of mode$_p$ and mode$_q$, respectively; $\boldsymbol{x}_p^*$ and $\boldsymbol{x}_q^*$ denote the equilibrium of mode$_p$ and mode$_q$, respectively; and $\boldsymbol{K}_p$ and $\boldsymbol{K}_q$ are the blending matrices that are functions of $\boldsymbol{x}_{pq}$.

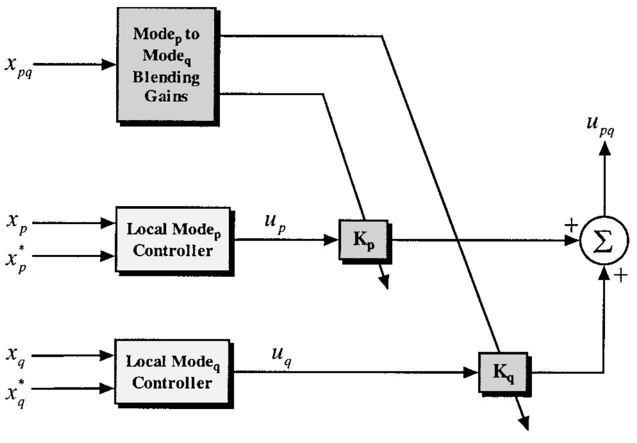
Mode$_p$ to Mode$_q$ Blending Gains
x_{pq}
u_{pq}
x_p
x_p^*
Local Mode$_p$ Controller
u_p
K_p
+
Σ
+
x_q
x_q^*
Local Mode$_q$ Controller
u_q
K_q

Fig. 2 Mode transition controller structure.

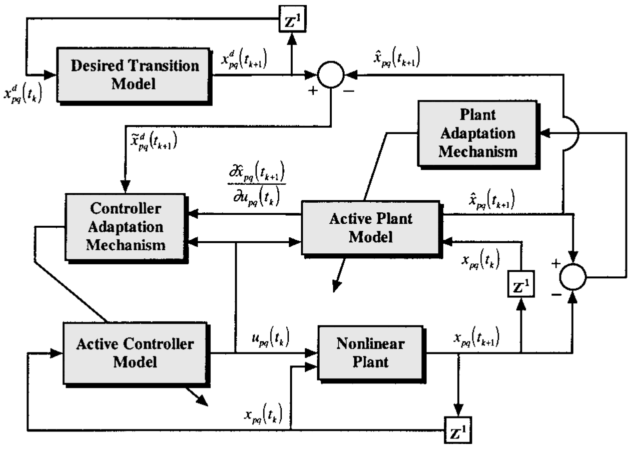

Fig. 3 Configuration for indirect adaptive mode transition control.

IV. Online Adaptation of Mode Transition Controllers

In this section, an adaptation scheme is proposed for the online customization of a mode$_p$ to mode$_q$ controller designed offline via the BLMC method. The control objective is to adapt the blending matrices such that the plant output vector tracks the output vector of a desired transition model. Note that the online adaptation of a nominal mode transition controller is based on a discrete-time adaptation scheme, which is applied to a continuous-time system. To apply the discrete-time controller scheme to a continuous-time system, it is assumed that the sample rate has been appropriately selected.

Figure 3 shows the configuration for indirect, adaptive mode transition control. The desired transition model is a fuzzy neural model of the desired mode transition trajectory that is trained offline only. The active controller model represents the mode-to-mode controller where the blending matrices are adapted online. The controller adaptation mechanism computes the best control values at each sample t_k using incremental linear model information from the active plant model, and it adapts the active controller model to capture the current input and the best control output values at time t_k. The active plant model is a fuzzy neural model of the unknown nonlinear plant along the desired mode transition trajectory; it is trained offline and adapted online using the plant adaptation mechanism to account for plant variations in real time.

A. Desired Transition Model

An offline trained fuzzy neural model of $\boldsymbol{x}_{pq}^d(t_k) \rightarrow \boldsymbol{x}_{pq}^d(t_{k+1})$ is determined for $k = 0, \ldots, N_1$ where $t_{k+1} - t_k = (t_f - t_0)/N_1$. Note that when $t_k \geq t_f$, then $\boldsymbol{x}_{pq}^d(t_k) = \boldsymbol{x}_{pq}^d(t_{k+1}) = \boldsymbol{x}_{pq}^d(t_f)$.

B. Active Plant Model

Given $x^d_{pq}(t_k)$ and $u^d_{pq}(t_k)$ for $k = 0, \ldots, N_1$ where $t_{k+1} - t_k = (t_f - t_0)/N_1$, a fuzzy neural model of the mapping $\{[x^d_{pq}(t_k), u^d_{pq}(t_k)] \rightarrow x^d_{pq}(t_{k+1})\}$ for $k = 0, \ldots, N_1$ is determined by offline training. Also, the linear model information $\partial x_{\mathrm{pq}}(t_{k+1})/\partial x_{\mathrm{pq}}(t_k)$ and $\partial x_{pq}(t_{k+1})/\partial u_{pq}(t_k)$ defined at $[x^d_{pq}(t_k), u^d_{pq}(t_k)]$ are incorporated into the consequent part of the fuzzy neural model. Afterward, the active plant model is adapted online via the plant adaptation mechanism.

C. Plant Adaptation Mechanism

The active plant is adapted online to account for plant variations in real time. At time instant t_k, the adaptation of the active plant model is accomplished by performing structure/parameter learning on the basis of the current input/output data $\{[x_{pq}(t_k), u_{pq}(t_k)] \rightarrow x_{pq}(t_{k+1})\}$. Because a desired output is not known ahead of time when structure learning is performed on the incoming input $[x_{pq}(t_k), u_{pq}(t_k)]$, the strongest fired rule's consequence is used. Likewise, the strongest fired rule's consequent parameters are used to initialize the consequent parameters of the newly formed rule because the required linear model information is not known ahead of time.

D. Active Controller Model

The active controller model is the mode$_p$ to mode$_q$ controller. Given $x^d_{pq}(t_k)$ and $k_{pq}(t_k)$ for $k = 0, \ldots, N_1$ where $t_{k+1} - t_k = (t_f - t_0)/N_1$, a fuzzy neural model of the mapping $x^d_{pq}(t_k), \rightarrow k_{pq}(t_k)$ for $k = 0, \ldots, N_1$ is determined by offline training as described in step 5 of Sec. III.B. Afterward, the blending weights of the active controller model are adapted online using the controller adaptation mechanism.

E. Controller Adaptation Mechanism

Let ACM and APM denote the active controller model and the active plant model, respectively. Let $u_{pq}(t_k)$ be the currently developed control input by the ACM that corresponds to $x_{pq}(t_k)$. Suppose that $x^d_{pq}(t_k)$ represents the desired trajectory at t_k provided by the desired transition model. Let $u'_{pq}(t_k)$ denote the control that is to be developed such that it is the weighted least-squares (WLS) optimal control value at t_k. The plant output vector corresponding to $u'_{pq}(t_k)$ is denoted as $x'_{pq}(t_k)$. Let $\tilde{u}_{\mathrm{pq}}(t_k) = u'_{pq}(t_k) - u_{pq}(t_k)$ and $\tilde{x}_{pq}(t_k) = x'_{pq}(t_k) - x_{pq}(t_k)$. Supposing that the $\tilde{u}_{pq}(t_k)$ is sufficiently small:

$$\tilde{x}_{pq}(t_{k+1}) = \frac{\partial x_{pq}(t_{k+1})}{\partial u_{pq}(t_k)}\tilde{u}_{pq}(t_k) = D[x_{pq}(t_k), u_{pq}(t_k)] \cdot \tilde{u}_{pq}(t_k) \quad (14)$$

where $D[x_{pq}(t_k), u_{pq}(t_k)]$ is the control sensitivity matrix. Let $\tilde{x}^d_{pq}(t_{k+1}) = x^d_{pq}(t_{k+1}) - x_{pq}(t_{k+1})$ and $\bar{x}_{pq}(t_{k+1}) = x^d_{pq}(t_{k+1}) - x'_{pq}(t_{k+1})$. Therefore, $\tilde{x}^d_{pq}(t_{k+1}) = D \cdot \tilde{u}_{pq}(t_k) + \bar{x}_{pq}(t_{k+1})$.

The optimal control input increments $\tilde{u}_{\mathrm{pq}}(t_k)$ are determined such that the following performance index is minimized:

$$J = \tfrac{1}{2}\bar{x}_{pq}(t_{k+1}) \cdot Q \cdot \bar{x}_{pq}(t_{k+1}), \qquad Q > 0 \quad (15)$$

Therefore,

$$\frac{\partial J}{\partial \tilde{u}_{pq}(t_k)} = \frac{1}{2} \cdot \left[\frac{\partial \bar{x}_{pq}(t_{k+1})}{\partial \tilde{u}_{pq}(t_k)}\right]^T \cdot \frac{\partial J}{\partial \bar{x}_{pq}(t_{k+1})} = 0 \quad (16)$$

where

$$\frac{\partial J}{\partial \bar{x}_{pq}(t_{k+1})} = 2 \cdot Q \cdot \bar{x}_{pq}(t_{k+1}), \qquad \frac{\partial \bar{x}_{pq}(t_{k+1})}{\partial \tilde{u}_{pq}(t_k)} = -D$$

Using Eq. (16) and $\tilde{x}^d_{pq}(t_{k+1}) = D \cdot \tilde{u}_{pq}(t_k) + \bar{x}_{pq}(t_{k+1})$,

$$\frac{\partial J}{\partial \tilde{u}_{pq}(t_k)} = -D^T \cdot Q \cdot \bar{x}_{pq}(t_{k+1}) = 0$$

$$\Rightarrow D^T \cdot Q\left\{\tilde{x}^d_{pq}(t_{k+1}) - D\tilde{u}_{pq}(t_k)\right\} = 0 \quad (17)$$

Solving for $\tilde{u}_{\mathrm{pq}}(t_k)$ in Eq. (17) yields

$$\tilde{u}_{pq}(t_k) = [D^T \cdot Q \cdot D]^{-1}D^T \cdot Q \cdot \tilde{x}^d_{pq}(t_{k+1}) \quad (18)$$

Therefore, the WLS optimal control value is determined to be

$$u'_{pq}(t_k) = u_{pq}(t_k) + [D^T \cdot Q \cdot D]^{-1}D^T \cdot Q \cdot \tilde{x}^d_{pq}(t_{k+1}) \quad (19)$$

The steps of the controller adaptation mechanism algorithm are

1) Apply the ACM to $x_{pq}(t_k)$ and produce the current initial estimate of the control input to $u_{pq}(t_k)$. Because it is possible for the fuzzy neural model of the blending weights not to be sufficiently activated by $x_{pq}(t_k)$, structure learning with local model information is performed at this stage.

2) Input $u_{pq}(t_k)$ and $x_{pq}(t_k)$ to the APM and produce $\hat{x}_{pq}(t_{k+1})$. Calculate $\tilde{x}^d_{pq}(t_{k+1})$ using the predictive one-step-ahead output $\hat{x}_{pq}(t_{k+1})$ in place of the unavailable output $x_{pq}(t_{k+1})$.

3) The true control sensitivity matrix $D[x_{pq}(t_k), u_{pg}(t_k)]$ is approximated via the APM's incremental control matrix $\hat{D}$, which can be calculated from Eq. (19). When the APM is not sufficiently activated by $(x_{pq}(t_k), u_{pq}(t_k))$, the control sensitivity information contained in the strongest fired rule's consequence is used to determine $\hat{D}$.

4) Compute the adjusted control law, $u'_{pq}(t_k) = u_{pq}(t_k) + [\hat{D}^T \cdot Q \cdot \hat{D}]^{-1}\hat{D}^T \cdot Q \cdot \tilde{x}^d_{pq}(t_{k+1})$. Afterward, calculate the desired blending weights $k'_{pq}(t_k)$.

5) Train the ACM to capture the desired blending weights $k'_{pq}(t_k)$ given current input $x_{pq}(t_k)$. Note that parameter learning with local model information is used to train the ACM.

6) Put $t_k \leftarrow t_{k+1}$ and perform the same procedure at the next time t_{k+1}.

V. Hover-to-Forward-Flight Example

A. Model of Helicopter's Forward Dynamics

The offline design approach described in Sec. III.B is used to design a hover-to-forward-flight (FF) transition controller for the following model representing the longitudinal channel dynamics of an Apache helicopter[34] constrained to have no vertical motion; only longitudinal and pitch rotation motions are allowed:

$$X = X_{\mathrm{trim}} + X_{\dot{x}}(\dot{x} - \dot{x}_{\mathrm{trim}}) + X_{\dot{\theta}}(\dot{\theta} - \dot{\theta}_{\mathrm{trim}}) + X_{\delta_e}(\delta_e - \delta_{e,\mathrm{trim}})$$

$$M = M_{\mathrm{trim}} + M_{\dot{x}}(\dot{x} - \dot{x}_{\mathrm{trim}}) + M_{\dot{\theta}}(\dot{\theta} - \dot{\theta}_{\mathrm{trim}}) + M_{\delta_e}(\delta_e - \delta_{e,\mathrm{trim}})$$

$$\ddot{x} = X/[m \cdot \cos(\theta)] - g \cdot \tan(\theta), \qquad \ddot{\theta} = M/I_Y$$

where $\ddot{x}$, $\ddot{\theta}$, and δ_e represent the forward acceleration (ft/s^2), pitch angle acceleration (rad/s^2), and longitudinal cyclic input (deg), respectively. X represents the aerodynamic force along the x axis and M represents the pitching moment about the y axis. Figure 4 shows the axis system of the helicopter with respect to the sideview. Table 1 describes the aerodynamic and physical parameters of the longitudinal channel dynamics model. The parameters X_{trim}, $X_{\dot{x}}$, $X_{\dot{\theta}}$, X_{δ_e}, M_{trim}, $M_{\dot{x}}$, $M_{\dot{\theta}}$, M_{δ_e}, $\dot{x}_{\mathrm{trim}}$, $\dot{\theta}_{\mathrm{trim}}$, $\delta_{e,\mathrm{trim}}$ are functions of $\dot{x}$. The physical constants m and I_Y have values of 4.5528×10^2 and 3.7409×10^4, respectively. The state vector of the helicopter model is $[x_1\ x_2\ x_3\ x_4]^T = [\dot{x}\ \ddot{x}\ \theta\ \dot{\theta}]^T$. It is assumed that the output vector of the model is the same as the state vector.

B. Hover-to-Forward-Flight Mode Controller

1. *BLMCs Approach*

The following control law was used for the hover-to-FF controller:

$$\delta_e = \delta_{e,\mathrm{hov}}(\dot{x}, \ddot{x}, \theta, \dot{\theta}) \cdot K_{\mathrm{hov}}(\dot{x}, \ddot{x}, \theta, \dot{\theta})$$

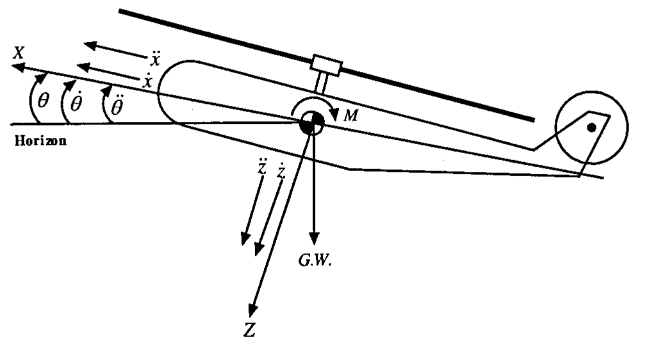

Fig. 4 Side view of helicopter's axis system.

Table 1 Description of aerodynamic and physical parameters

Parameters	Description
$X_{\dot{x}}$	Partial derivative of X w.r.t. $\dot{x}$
$X_{\dot{\theta}}$	Partial derivative of X w.r.t. $\dot{\theta}$
X_{δ_e}	Partial derivative of X w.r.t. δ_e
X_{trim}	Trim value of aerodynamic force X
$M_{\dot{x}}$	Partial derivative of M w.r.t. $\dot{x}$
$M_{\dot{\theta}}$	Partial derivative of M w.r.t. $\dot{\theta}$
M_{δ_e}	Partial derivative of M w.r.t. δ_e
M_{trim}	Trim value of aerodynamic moment M
$\dot{x}_{\text{trim}}$	Trim value of forward velocity $\dot{x}$
$\dot{\theta}_{\text{trim}}$	Trim value of pitch angle rate $\dot{\theta}$
$\delta_{e,\text{trim}}$	Trim value of longitudinal input δ_e
m	Mass of the helicopter
I_Y	Moment of inertia along y axis

$$+\,\delta_{e,\text{FF}}(\dot{x},\ddot{x},\theta,\dot{\theta})\cdot K_{\text{FF}}(\dot{x},\ddot{x},\theta,\dot{\theta})$$

where $\delta_{e,\text{hov}}(\cdot)$ and $\delta_{e,\text{FF}}(\cdot)$ are linear quadratic regulators that regulate about the operating points

$$[\dot{x} \quad \ddot{x} \quad \theta \quad \dot{\theta}]^T = [0.0000 \quad 0.0000 \quad 0.1008 \quad 0.0000]^T$$

and

$$[\dot{x} \quad \ddot{x} \quad \theta \quad \dot{\theta}]^T = [92.8278 \quad 0.0000 \quad 0.0402 \quad 0.0000]^T$$

respectively. The scalar gains $K_{\text{hov}}(\dot{x},\ddot{x},\theta,\dot{\theta})$ and $K_{\text{FF}}(\dot{x},\ddot{x},\theta,\dot{\theta})$ are determined such that the closed-loop system transitions from $[0.0000\ 0.0000\ 0.1008\ 0.0000]^T$ to $[92.8278\ 0.0000\ 0.0402\ 0.0000]^T$ in minimum time with the following constraints:

$$-1.0000 \le \dot{x} \le 94.0000, \qquad -2.5000 \le \ddot{x} \le 20.0000$$

$$-0.7000 \le \theta \le 0.7000, \qquad -0.6000 \le \dot{\theta} \le 0.6000$$

$$-6.5000 \le \delta_e \le 4.5000$$

Afterward, the blending gains $K_{\text{hov}}(\dot{x},\ddot{x},\theta,\dot{\theta})$ and $K_{\text{FF}}(\dot{x},\ddot{x},\theta,\dot{\theta})$ are realized via an FNN construct described in Sec. II.

2. Gain-Scheduling Approach

A gain-scheduled controller is designed via dynamic linearization about the minimum time trajectory determined in Sec. V.B.1:

1) Sixty-seven equidistant frozen times $t_1,\ldots,t_{67}$ are chosen, where $t_{k+1}=(t_f-t_0)/66$.

2) Sixty-seven linear autonomous open-loop systems are obtained via Lyapunov linearization of the helicopter model about the minimum time state and control trajectories at frozen times $t_1,\ldots,t_{67}$.

3) Sixty-seven linear quadratic regulators are designed for each time-frozen linear model of the helicopter created in the previous step.

4) The 67 linear control laws are blended according to how near the current state is to each of the frozen operating states determined in the first step. The interpolated control law is applied to the system to be controlled.

3. Least-Squares Adaptation Scheme

The sample time of $T_S=0.05$ s is chosen for the adaptation scheme. The desired minimum time trajectory and control are resampled such that they occur every T_S: $\boldsymbol{x}^d(t_k)$ and $\boldsymbol{u}^d(t_k)$ for $k=0,\ldots,N$, where $\boldsymbol{x}^d(t_k)=[\dot{x}^d(t_k)\ \ddot{x}^d(t_k)\ \theta^d(t_k)\ \dot{\theta}^d(t_k)]^T$, $\boldsymbol{u}^d(t_k)=[\delta_e^d(t_k)]$ and $t_{k+1}-t_k=T_S$. $t_N\ge t_f$ and $t_N-t_f<T_S$. Afterward, the desired transition model of the following mapping is determined offline:

$$\boldsymbol{x}^d(t_k)\rightarrow \boldsymbol{x}^d(t_{k+1}), \qquad k=0,\ldots,N$$

The APM is initially determined offline for the following mapping:

$$\left\{\left[\boldsymbol{x}^d(t_k),\,\boldsymbol{u}^d(t_k)\right]\rightarrow \boldsymbol{x}^d(t_{k+1})\right\} \qquad \text{for} \qquad k=0,\ldots,N$$

Also, the linear model information defined at $[\boldsymbol{x}^d(t_k),\,\boldsymbol{u}^d(t_k)]$ for $k=0,\ldots,N$

$$\frac{\partial \boldsymbol{x}(t_{k+1})}{\partial \boldsymbol{x}(t_k)}, \qquad \frac{\partial \boldsymbol{x}(t_{k+1})}{\partial \boldsymbol{u}(t_k)}$$

is incorporated into the consequent part of the APM.

The plant adaptation mechanism adapts the APM with the following parameters:

$$\delta=0.2, \qquad \beta=0.5$$

$$\boldsymbol{\sigma}^U=\left[\sigma_1^U \quad \ldots \quad \sigma_5^U\right]=[0.20 \quad 0.20 \quad 0.05 \quad 0.05 \quad 0.20]$$

where δ, β, and σ^U are the lower threshold for membership value, the desired overlap degree between membership functions, and the upper limit of the width of each membership function, respectively.

The ACM is the hover-to-FF mode controller determined in Sec. V.B.1. The controller adaptation mechanism adapts the blending weights of the active controller model with the following parameters:

$$\delta=0.2, \qquad \beta=0.5$$

$$\boldsymbol{\sigma}^U=\left[\sigma_1^U \quad \cdots \quad \sigma_4^U\right]=[0.20 \quad 0.20 \quad 0.05 \quad 0.05]$$

C. Simulation Results

Figures 5–8 show the $\dot{x}$, $\ddot{x}$, θ, and $\dot{\theta}$ trajectories with respect to nominal parameters and no wind disturbance for the controllers designed in Sec. V.B.1 and V.B.2. The controller designed via the

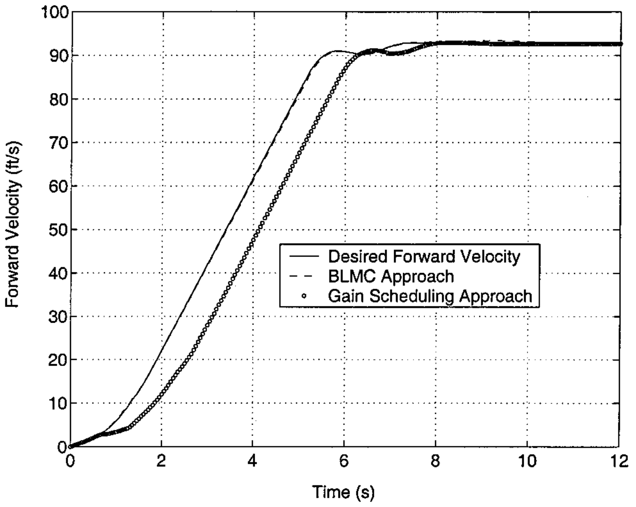

Fig. 5 Plots of $\dot{x}$ nominal trajectories.

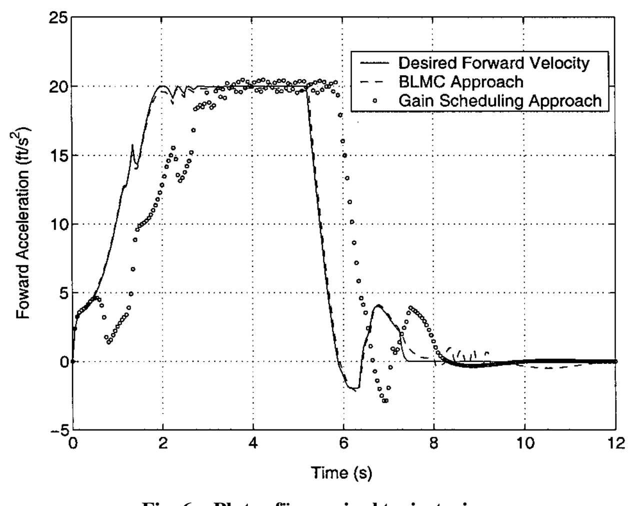

Fig. 6 Plots of $\ddot{x}$ nominal trajectories.

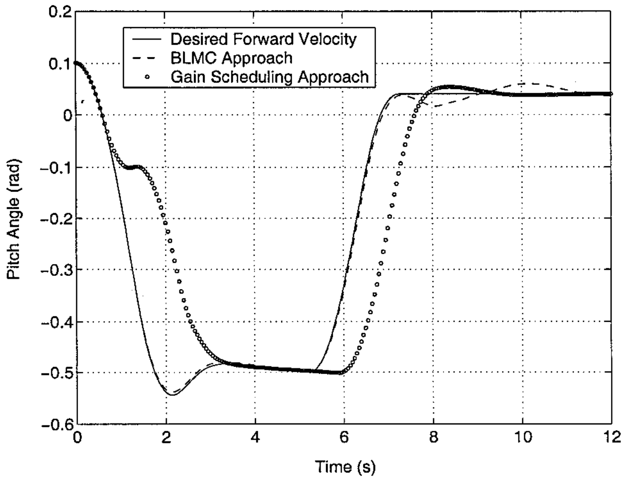

Fig. 7 Plots of θ nominal trajectories.

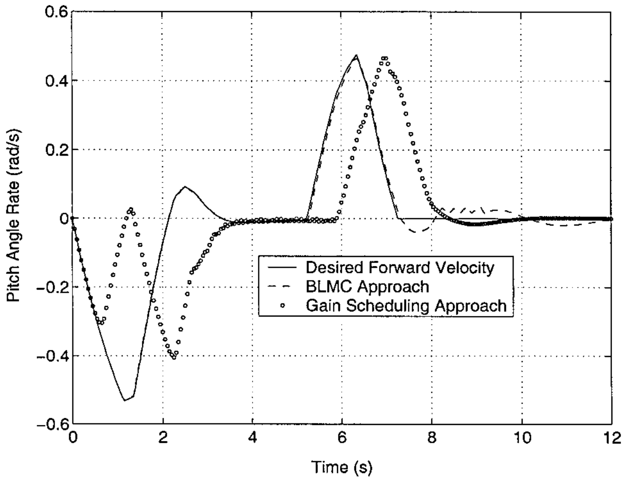

Fig. 8 Plots of $\dot{\theta}$ nominal trajectories.

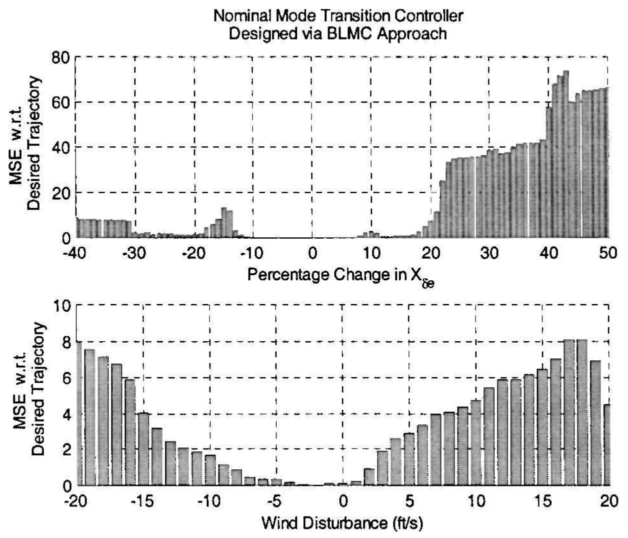

Fig. 9 Plots of MSEs for controller designed via BLMC approach.

BLMC approach exhibits good tracking performance of the desired transition trajectory for the nominal system, but the gain-scheduling controller has poor tracking performance. This result is not unusual, because the first controller is designed to track the desired trajectory for the nominal system, and the gain-scheduling controller design does not necessarily guarantee good transient behavior when tracking a desired trajectory.

For the controllers designed in Sec. V.B.1 and V.B.2, Figs. 9 and 10 show the mean squared error (MSE) from the desired transition

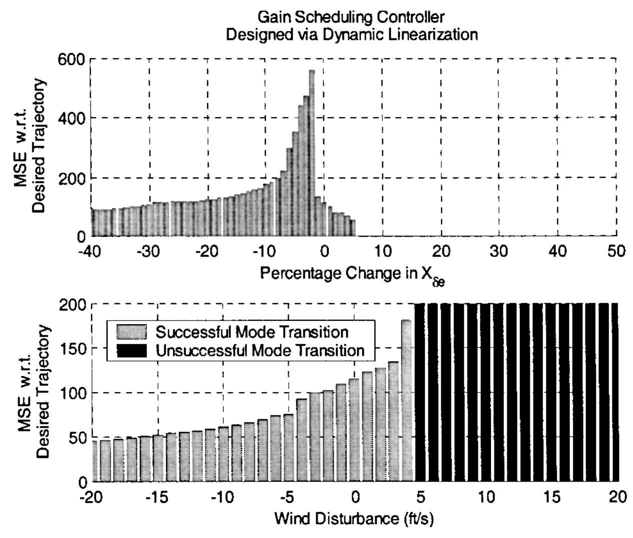

Fig. 10 Plots of MSEs for gain-scheduling controller.

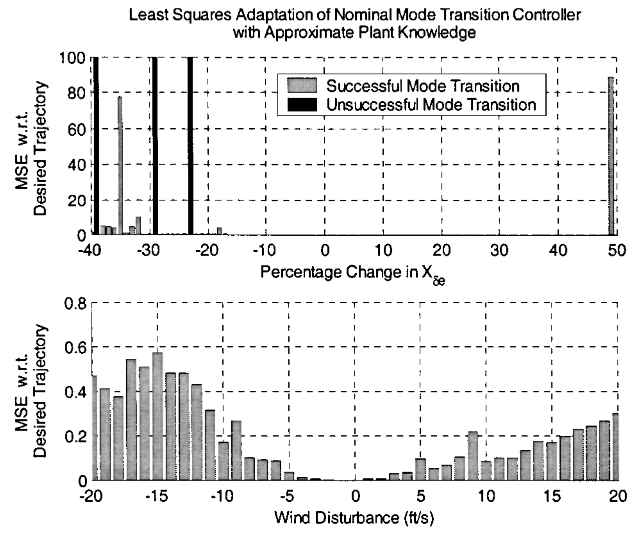

Fig. 11 Plots of MSEs for adaptation with approximate plant information.

trajectory for wind disturbances and parametric changes of X_{δ_e}. Although the gain-scheduling controller exhibits good tracking performance for percentage changes of X_{δ_e} between 6 and 50%, the controller designed via the BLMC approach possesses a more robust tracking performance over the range of parameter changes of X_{δ_e}. In particular, the average MSE due to parametric changes of X_{δ_e} is approximately 17.5 and 78.3 for the controllers designed via the BLMC approach and the gain-scheduling approach, respectively. Also, for the case of −20–20 ft/s wind disturbance, the controller designed via the BLMC approach outperformed the gain-scheduling controller. Although the controller designed in Sec. V.B.1 exhibited no controller faults, the gain-scheduling controller did not transition the system from $[0.0000\ 0.0000\ 0.1008\ 0.0000]^T$ to $[92.8278\ 0.0000\ 0.0402\ 0.0000]^T$ for wind disturbances between 5 and 20 ft/s.

For the adaptation of the controller designed in Sec. V.B.1, Figs. 11 and 12 show the MSE from the desired transition trajectory for wind disturbances and parametric changes of X_{δ_e}. If the approximate plant accurately captures the local model information and the input/output behavior of the system to be controlled, the adapted controller exhibits excellent tracking performance when encountering parametric changes and wind disturbances. Because the adaptation scheme adapts the controller of Sec. V.B.1 such that the future error is erased, the aggressiveness of the scheme can lead to controller faults when the approximate plant poorly models the system to be controlled, as shown in Fig. 11. Tables 2 and 3 show

Table 2 Results for various controllers encountering parametric changes of X_{δ_e}

Controllers	Average MSE	Minimum MSE	Maximum MSE	Controller faults
BLMC approach	17.4822	0.0136	73.8291	0
Gain-scheduling approach	78.3444	0.1158	560.3075	0
Adaptation with approx. plant info	2.4625	0.0021	89.1941	3
Adaptation with perfect plant info	0.2477	0.0001	1.3411	0

Table 3 Results for various controllers encountering wind disturbances

Controllers	Average MSE	Minimum MSE	Maximum MSE	Controller faults
BLMC approach	3.6106	0.0164	8.1026	0
Gain-scheduling approach	80.2857	45.4705	181.1169	16
Adaptation with approx. plant info	0.1957	0.0014	0.5747	0
Adaptation with perfect plant info	0.1219	0.0001	0.4452	0

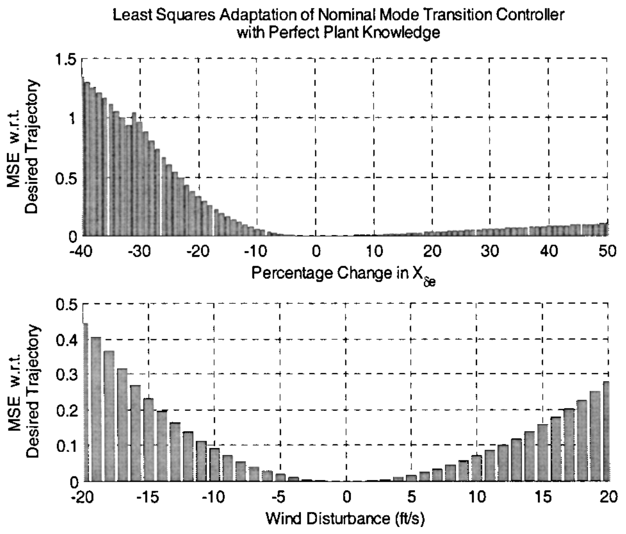

Fig. 12 Plots of MSEs for adaptation with accurate plant information.

that the adaptation scheme can lead to smaller tracking errors in the presence of parametric changes and wind disturbances. Note that the average, minimum, and maximum MSEs are calculated from successful mode transitions.

VI. Conclusions

An adaptation scheme is proposed for the real-time adaptation of mode transition controllers designed via blending local mode controllers. When the APM of the adaptation scheme is a good approximation of the system to be controlled, then it is expected that the adapted controller will track the desired trajectory very well in the presence of parametric changes and disturbances. However, if the APM does not capture the local model and the input/output behavior of the system to be controlled, poor tracking performance to unstable tracking can result. The incremental control law to remove the future error needs to be modified to take into account the previously predicted output errors and errors due to approximating the control sensitivity matrix. Another way of preventing controller faults will be to decrease the aggressiveness of the scheme when approximation errors are significant and increase the aggressiveness when the errors are small.

Acknowledgment

This work has been supported by Defense Advanced Research Projects Agency/Air Force Research Laboratory under Contract F33615-98-C1341. We gratefully acknowledge their continued encouragement and support.

References

[1]Shamma, J. S., and Athans, M., "Analysis of Gain Scheduled Control for Nonlinear Plants," *IEEE Transactions in Automatic Control*, Vol. 35, No. 8, 1990, pp. 898–907.

[2]Shamma, J. S., and Athans, M., "Gain Scheduling: Potential Hazards and Possible Remedies," *IEEE Control Systems Magazine*, Vol. 12, No. 3, 1992, pp. 101–107.

[3]Gonsalves, P. G., and Zacharias, G. L., "Fuzzy Logic Gain Scheduling for Flight Control," *IEEE International Conference Fuzzy Systems*, Vol. 2, Inst. of Electrical and Electronics Engineers, New York, 1994, pp. 952–957.

[4]Rugh, W. J., "Analytical Framework for Gain Scheduling," *IEEE Control Systems Magazine*, Vol. 11, No. 1, 1991, pp. 79–84.

[5]Nichols, R. A., Reichert, R. T., and Rugh, W. J., "Gain-Scheduling for H_∞ Controllers: A Flight Control Example," *IEEE Transactions on Control System Technology*, Vol. 1, No. 2, 1993, pp. 69–78.

[6]Hyde, R. A., and Glover, K., "The Application of Scheduled H_∞ Controllers to a VSTOL Aircraft," *IEEE Transactions in Automatic Control*, Vol. 38, No. 7, 1993, pp. 1021–1039.

[7]Leith, D. J., and Leithead, W. E., "Appropriate Realization of Gain Scheduled Controllers with Application to Wind Turbine Regulation," *International Journal of Control*, Vol. 65, No. 2, 1996, pp. 223–248.

[8]Ravi, R., Nagpal, K., and Khargonekar, P., "H_∞ Control of Linear Time-Varying Systems: A State Space Approach," *SIAM Journal on Control and Optimization*, Vol. 29, No. 6, 1991, pp. 1394–1413.

[9]Voulgaris, P. G., and Dahleh, M. A., "On l^∞ to l^∞ Performance of Slowly Varying Systems," *Systems and Control Letters*, Vol. 24, No. 4, 1995, pp. 243–249.

[10]Scherpen, J. M., and Verhaegen, M. H., "H^∞ Output Feedback Control for Linear Discrete Time-Varying Systems via the Bounded Real Lemma," *International Journal of Control*, Vol. 65, No. 6, 1996, pp. 963–993.

[11]Dimiter, D., Palm, R., and Rehfuess, U., "A Takagi-Sugeno Fuzzy Gain-Scheduler," *IEEE International Conference on Fuzzy Systems*, Inst. of Electrical and Electronics Engineers, New York, Vol. 2, 1996, pp. 1053–1059.

[12]Leith, D. J., and Leithead, W. E., "Gain-Scheduled and Nonlinear Systems: Dynamic Analysis by Velocity-Based Linearization Families," *International Journal of Control*, Vol. 70, No. 2, 1998, pp. 289–317.

[13]Leith, D. J., and Leithead, W. E., "Gain-Scheduled Controller Design: An Analytic Framework Directly Incorporating Non-Equilibrium Plant Dynamics," *International Journal of Control*, Vol. 70, No. 2, 1998, pp. 249–269.

[14]Ng, K. C., Li, Y., Murray-Smith, D. J., and Sharman, K. C., "Genetic Algorithms Applied to Fuzzy Sliding Mode Controller Design," CSC-95016 Center for Systems and Control, Univ. of Glasgow, Scotland, March 1994.

[15]Bashi, A. S., "A Comparison Between Linear Quadratic Control and Sliding Mode Control," URL: http://www.uno.edu/~SAGES/publications/SlidingModeControl.PDF [cited 9 April 2001].

[16]Jalili, N., and Olgac, N., "Time-Optimal/Sliding Mode Control Implementation for Robust Tracking of Uncertain Flexible Structures," *Mechatronics*, Vol. 8, No. 2, 1998, pp. 121–142.

[17]Slotine, J. E., "Sliding Controller Design for Nonlinear Systems," *International Journal of Control*, Vol. 40, No. 2, 1984, pp. 421–434.

[18]Wang, W., and Lin, H., "Fuzzy Control Design for the Trajectory Tracking on Uncertain Nonlinear Systems," *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 1, 1999, pp. 53–62.

[19]Jagannathan, S., "Adaptive Fuzzy Logic Control of Feedback Linearizable Discrete-Time Dynamical Systems Under Persistence of Excitation," *Automatica*, Vol. 34, No. 11, 1998, pp. 1295–1310.

[20]Zhang, T., Ge, S. S., and Hang, C. C., "Neural-Based Direct Adaptive Control for a Class of General Nonlinear Systems," *International Journal of Systems Science*, Vol. 28, No. 10, 1997, pp. 1011–1020.

[21]Marino, R., and Tomei, P., "Adaptive Output Feedback Tracking Control for Nonlinear Systems with Time-Varying Parameters," *Proceedings of the IEEE Conference on Decision and Control*, Vol. 3, Inst. of Electrical and Electronics Engineers, New York, 1997, pp. 2483–2488.

[22]Ma, X., and Loh, N. K., "One-Step-Ahead Controller Design Using Neural Networks," *Proceedings of the American Control Conference*, Vol. 2, American Automatic Control Council, New York, 1992, pp. 958–962.

[23]Ma, X., and Loh, N. K., "Neural Network-Based Successive One-Step-Ahead Control of Nonlinear Systems," *Proceedings of the American Control Conference*, Vol. 4, American Automatic Control Council, New York, 1994, pp. 2129–2133.

[24]Tan, Y., and Cauwenberghe, A. V., "Nonlinear One-Step-Ahead Control Using Neural Networks: Control Strategy and Stability Design," *Automatica*, Vol. 32, No. 12, 1996, pp. 1701–1706.

[25]Song, J. B., and Hardt, D. E., "Application of Adaptive Control to Arc Welding Processes," *Proceedings of the American Control Conference*, American Automatic Control Council, New York, 1993, pp. 1751–1754.

[26]Mingzhong, L., and Fuli, W., "Adaptive Control of Black-Box Nonlinear Systems Using Recurrent Neural Networks," *Proceedings of the IEEE Conference on Decision and Control*, Vol. 5, Inst. of Electrical and Electronics Engineers, New York, 1997, pp. 4165–4170.

[27]Theocharis, J., and Vachtsevanos, G., "Adaptive Fuzzy Neural Networks as Identifiers of Discrete-Time Nonlinear Dynamic Systems," *Journal of Intelligent and Robotic Systems*, Vol. 17, No. 2, 1996, pp. 119–168.

[28]Jang, J.-S. R., "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, 1993, pp. 665–685.

[29]Jang, J.-S. R., "Self Learning Fuzzy Controllers Based on Temporal Back Propagation," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, 1992, pp. 714–723.

[30]Lin, C. T., and Lee, C. S. G., "Real-Time Supervised Structure/ Parameter Learning for Fuzzy Neural Network," *IEEE Conference on Fuzzy Systems*, Inst. of Electrical and Electronics Engineers, New York, 1992, pp. 1283–1291.

[31]Lin, C. T., and Lee, C. S. G., "Reinforcement Structure/Parameter Learning for Neural-Network-Based Fuzzy Logic Control Systems," *IEEE Transactions on Fuzzy Systems*, Vol. 2, No. 1, 1995, pp. 46–63.

[32]Hunt, K. J., Irwin, G. R., and Warwick, K., *Neural Network Engineering in Dynamic Control Systems*, Springer-Verlag, New York, 1991, pp. 42–45.

[33]Rufus, F., Clements, S., Sander, S., Heck, B., Wills, L., and Vachtsevanos, G., "Software-Enabled Control Technologies for Autonomous Aerial Vehicles," *18th Digital Avionics System Conference*, Vol. 2, Inst. of Electrical and Electronics Engineers, New York, 1999, pp. 6.A.5-1–6.A.5-8.

[34]Prasad, J. V. R., and Lipp, A. M., "Modeling and Nonlinear Controller Development for the Apache Helicopter using the GTNONCON2 Program," E-16-690, General Electric Aerospace Company, Binghamton, New York, pp. 1–30, Jan. 1992.