SOFTWARE-ENABLED ADAPTIVE MODE TRANSITION CONTROL FOR AUTONOMOUS UNMANNED VEHICLES

F. Rufus, B. Heck and G. Vachtsevanos
School of Electrical & Computer Engineering,
Georgia Institute of Technology
Atlanta, Georgia 30332-0250

Abstract

Typically, complex large-scale systems such as unmanned aerial vehicles are required to operate in a finite number of operational modes that necessitate robust, stable and smooth transitions between them. In this work, an online adaptation scheme is proposed for adapting the parameters of mode transition controllers designed off-line via the method of blending local mode controllers. The adaptive mode transition control algorithm is enabled via a software architecture that accommodates functionalities such as dynamic reconfigurability, plug and play extensibility, interoperability and openness. Through its reconfiguration management, virtual resource network and real-time distributed computing, it allows the UAV to execute agile and extreme performance maneuvers.

Introduction

Complex large-scale systems such as unmanned aerial vehicles and industrial processes are demanded to 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 require robust, stable and smooth transitions between them. A local operational mode is considered to be a region in the system's state space in which the system exhibits quasi steady-state behavior. And 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 solved by non-adaptive techniques such as gain scheduling [1,2], sliding mode control [3,4] and the method of blending local mode controllers [5]. However, when the system to be controlled differs significantly from the nominal system used in the design methods above, degraded tracking performance of the desired transition trajectory is to be expected.

In this work, an online adaptation scheme is proposed for adapting the parameters of mode transition controllers designed off-line via the method of blending local mode controllers (BLMC). 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 to be adapted. 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 discussed in [6]. All three fuzzy neural models are trained off-line while 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 the mode transition controller are determined from the control sensitivity matrix and the predicted output of the active plant model. A software substrate used for integrating adaptive mode transition controllers with other system components is discussed in the last section of this paper.

Adaptive Mode Transition Control

Consider a large-scale dynamical system that 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 i^{th} subsystem is given by:

$$\dot{x}_i = f_i(x_i, u_i), \qquad x_i \in R^{n_i},\ u_i \in R^{m_i}$$

0-7803-6395-7/00/$10.00 ©2000 IEEE

Let $mode_p$ and $mode_q$ denote the p^{th} and q^{th} subsystem, respectively. How do we design a controller that stably and smoothly transitions a system from $mode_p$ to $mode_q$?

Off-line Control Design

In [5], an off-line design methodology known as the BLMC approach was developed to design mode transition controllers. This approach for designing mode transition controller uses the aggregated states of the start and goal modes, while the output vector of the mode transition controller is determined by blending the individual output vector of the start and goal mode controllers. The following is an outline of the BLMC approach:

- Step 1: Design regulators for the start and goal modes such that initial states are driven to the equilibrium of the respective modes.
- Step 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.
- Step 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.
- Step 4: Determine the desired blending gains using the desired state and control trajectory determined from step 3.
- Step 5: Realize the blending gains via a fuzzy neural network construct proposed in [6].

The structure of a mode transition controller designed via the BLMC method is shown Figure 1, where x_{pq} is the aggregated state vector of x_p and x_q; u_{pq} is the aggregated control vector of u_p and u_q; x_p and x_q denote the state vectors of $mode_p$ and $mode_q$, respectively; u_p and u_q denote the control input vectors of $mode_p$ and $mode_q$, respectively; x_p^* and x_q^* denote the equilibrium of $mode_p$ and $mode_q$, respectively; K_p and K_q are the blending matrices which are functions of x_{pq}.

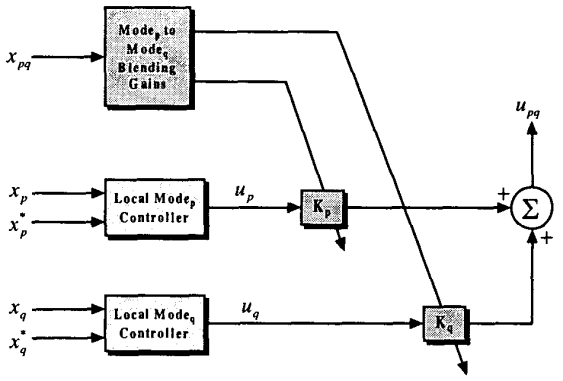

Figure 1. Mode transition controller structure.

Online Adaptation Scheme

In this section, an adaptation scheme is proposed for the online adaptation of the mode transition controllers designed off-line via the BLMC approach. The control objective is to adapt the blending matrices such that the plant output vector tracks the output vector of a desired transition model. In order to apply the discrete-time controller scheme to the continuous-time system, it is assumed that the sample rate has been appropriately selected. Figure 2 shows the configuration for indirect adaptive mode transition control. The adaptation scheme is composed of five components: a *desired transition model,* an *active plant model*, a *plant adaptation mechanism*, an *active controller model* and a *controller adaptation mechanism.*

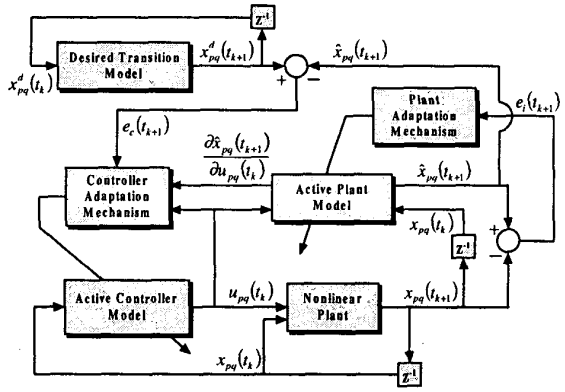

Figure 2. Configuration for indirect adaptive mode transition control

Desired Transition Model

A fuzzy neural model representation of the desired transition trajectory is determined off-line.

Active Plant Model

A fuzzy neural model of the input/output relationship of the nonlinear plant along the desired transition is determined by off-line training. Also, the linear model information along the desired trajectory is incorporated into the consequent part of the fuzzy neural model. Afterwards, the *active plant model* is adapted online via the plant adaptation mechanism.

Plant Adaptation Mechanism

The active plant model is adapted online to account for plant variations on a real-time basis. 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 of the system to be controlled.

Active Controller Model

The active controller model is the mode transition controller determined off-line via the BLMC approach. The blending gains of the active controller model are adapted online using the *controller adaptation mechanism.*

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 which corresponds to $x_{pq}(t_k)$. Suppose that $x^d_{pq}(t_{k+1})$ represents the desired trajectory at t_{k+1} provided by the desired transition model. The steps of the controller adaptation mechanism algorithm are:

- Step 1: Apply ACM to $x_{pq}(t_k)$ and produce the current initial estimate of the control input $u_{pq}(t_k)$.
- Step 2: Input $u_{pq}(t_k)$ and $x_{pq}(t_k)$ to APM and produce $\hat{x}_{pq}(t_{k+1})$. Calculate $\tilde{x}^d_{pq}(t_{k+1}) = x^d_{pq}(t_{k+1}) - x_{pq}(t_{k+1})$ using the predictive one-step-ahead output $\hat{x}_{pq}(t_{k+1})$ in place of the unavailable output $x_{pq}(t_{k+1})$.
- Step 3: The true control sensitivity matrix $D\left(x_{pq}(t_k), u_{pq}(t_k)\right)$ is approximated via the APM's incremental control matrix $\hat{D}$. When the APM is not sufficiently activated by the input $\left(x_{pq}(t_k), u_{pq}(t_k)\right)$, the control sensitivity information contained in the strongest fired rule's consequent parameters is used to determine $\hat{D}$.
- Step 4: Compute the weighted least squares optimal control law,
$$u'_{pq}(t_k) = u_{pq}(t_k) + \left[D^T \cdot Q \cdot D\right]^{-1} D^T \cdot Q \cdot \tilde{x}^d_{pq}(t_{k+1})$$
Afterwards, calculate the desired blending weights $k'_{pq}(t_k)$.
- Step 5: Train ACM to capture 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.
- Step 6: Put $t_k \leftarrow t_{k+1}$ and perform the same procedure at the next time t_{k+1}.

Hover to Forward Flight Example

Model of Helicopter's Forward Dynamics

The proposed adaptation scheme will be illustrated on the following model representing the longitudinal channel dynamics of an Apache helicopter constrained to have no vertical motion; only longitudinal and pitch rotation motions are allowed [5]:

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

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

$$\ddot{x} = \frac{X}{m \cdot \cos(\theta)} - g \cdot \tan(\theta)$$

$$\ddot{\theta} = \frac{M}{I_Y}$$

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 represent the aerodynamic force along the "X axis" and M represent the pitching moment about the "Y axis". Figure 3 shows the axis system of the helicopter with respect to the sideview. The parameters X_{trim}, $X_{\dot{x}}$, $X_{\dot{\theta}}$, X_{δ_e}, M_{trim}, $M_{\dot{x}}$, $M_{\dot{\theta}}$, M_{δ_e}, $\dot{x}_{trim}$, $\dot{\theta}_{trim}$, $\delta_{e,trim}$ are functions of $\dot{x}$. X_{trim} and M_{trim} are the trim values of the aerodynamic force X and the pitching moment M, respectively. The variables $X_{\dot{x}}$, $X_{\dot{\theta}}$, X_{δ_e}, M_{trim}, $M_{\dot{x}}$ and $M_{\dot{\theta}}$ are the partial derivatives of X and M with respect to $\dot{x}$, $\dot{\theta}$ and δ_e, respectively. The physical constants m and I_Y are the mass of the helicopter and the moment of inertia along the Y axis. The state vector of the helicopter model is $[x_1 \quad x_2 \quad x_3 \quad x_4]^T = [\dot{x} \quad \ddot{x} \quad \theta \quad \dot{\theta}]^T$. It is assumed that the output vector of the model is the same as the state vector.

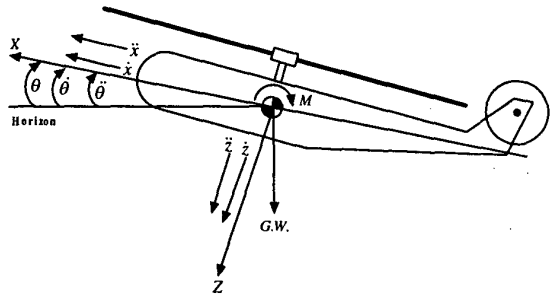

Figure 3. Side view of Helicopter's Axis System

Hover to Forward Flight Mode Controller

In [5], a hover to forward flight mode controller was designed via the BLMC approach. The controller was designed such that the closed-loop system transitions from $[0.0000 \quad 0.0000 \quad 0.1008 \quad 0.0000]^T$ to $[92.8278 \quad 0.0000 \quad 0.0402 \quad 0.0000]^T$ in minimum time with the following constraints:

$$-1.0000 \le \dot{x} \le 94.0000$$
$$-2.5000 \le \ddot{x} \le 20.0000$$
$$-0.7000 \le \theta \le 0.7000$$
$$-0.6000 \le \dot{\theta} \le 0.6000$$
$$-6.5000 \le \delta_e \le 4.5000.$$

The proposed adaptation scheme proposed is applied to the mode transition controller mentioned above. The sample time of $T_S = 0.05s$ is chosen for the adaptation scheme. The desired minimum time trajectory and control are resampled such that they occur every T_S:

$$\vec{x}^d(t_k) \text{ and } \vec{u}^d(t_k) \text{ for } k = 0,\ldots,N$$

where $\vec{x}^d(t_k) = [\dot{x} \quad \ddot{x} \quad \theta \quad \dot{\theta}]^T$, $\vec{u}^d(t_k) = [\delta_e]$, $t_{k+1} - t_k = T_S$, $t_N \ge t_f$ and $t_N - t_f < T_S$.

Afterwards, the *desired transition model* of the following mapping is determined off-line:

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

The *active plant model* is initially determined off-line for the following mapping:

$$\left\{\left(\vec{x}^d(t_k), \vec{u}^d(t_k)\right) \rightarrow \vec{x}^d(t_{k+1})\right\}$$
$$k = 0,\ldots,N$$

Also, the linear model information defined at $\left(\vec{x}^d_{pq}(t_k), \vec{u}^d_{pq}(t_k)\right)$ for $k = 0,\ldots,N$,

$$\frac{\partial \vec{x}(t_{k+1})}{\partial \vec{x}(t_k)} \text{ and } \frac{\partial \vec{x}(t_{k+1})}{\partial \vec{u}(t_k)}$$

are incorporated into the consequent part of *active plant model*. The *plant adaptation mechanism* adapts the active plant model with the following structure learning parameters:

$$\delta = 0.2,\ \beta = 0.5 \text{ and}$$
$$\sigma^U = [\sigma_1^U \quad \cdots \quad \sigma_5^U]$$
$$= [0.20 \quad 0.20 \quad 0.05 \quad 0.05 \quad 0.20]$$

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 *active controller model* is the hover to forward flight mode controller determined previously. The *controller adaptation mechanism* adapts the blending weights of the active controller model with the following structure learning parameters:

$$\delta = 0.2,\ \beta = 0.5 \text{ and}$$

$$\sigma^U = \begin{bmatrix} \sigma_1^U & \cdots & \sigma_4^U \end{bmatrix} = \begin{bmatrix} 0.20 & 0.20 & 0.05 & 0.05 \end{bmatrix}$$

Simulation Results

Figures 4-5 show the desired $\dot{x}$, $\ddot{x}$, θ and $\dot{\theta}$ trajectories. For the nominal mode transition controller, Figure 6 show the mean squared error from the desired transition trajectory for wind disturbances and parametric changes of X_{δ_e}. For small parametric changes and wind disturbances, the controller exhibits good tracking performance of the desired transition trajectory. However, as the magnitude of the parametric changes and wind disturbances increase the tracking performance of the controller degrades. Figure 7 show mean squared error from the desired transition trajectory for wind disturbances and parametric changes of X_{δ_e}, for the least squares adaptation of the nominal mode transition controller. As expected, 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.

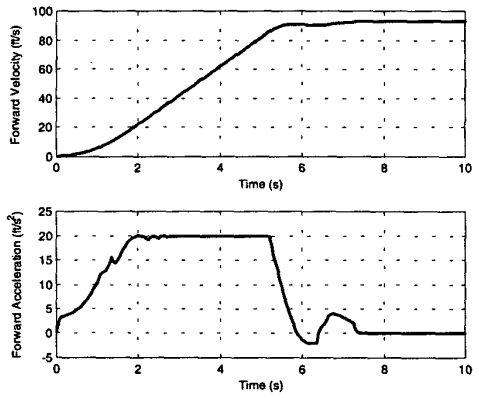

Figure 4. Plots of desired $\dot{x}$ and $\ddot{x}$ trajectories

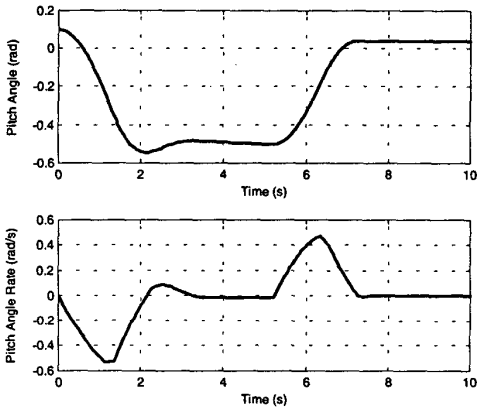

Figure 5. Plots of desired θ and $\dot{\theta}$ trajectories

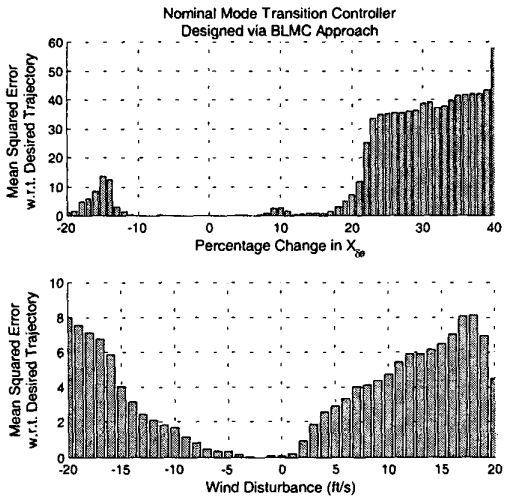

Figure 6. Plots of mean squared errors for controller designed via BLMC approach

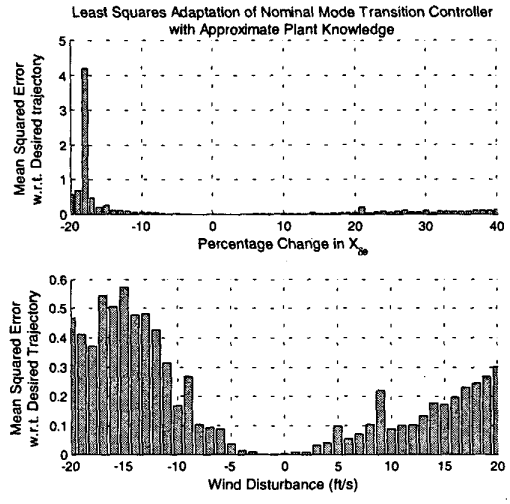

Figure 7. Plots of mean squared errors for adaptation with approximate plant knowledge

The Enabling Software Technology

The adaptive mode transition control algorithm described in this paper is being enabled by an emerging software architecture, called the *Open Control Platform* (OCP). The OCP will be used to integrate the adaptive mode transition control algorithm with a wide variety of components needed to achieve autonomous flight, such as the mode selector module, the mission planning module, the situation awareness module and the fault tolerant control module. The OCP is being designed to support the following capabilities that are required to accommodate the operational and environmental changes inherent in autonomous, extreme-performance flight:

- Plug and play extensibility: it should be easy to insert new technology, such as new control algorithms or sensor technology into the system architecture without redesigning the components already in the system.
- Interoperability in distributed, heterogeneous environments: control algorithms and other components may be running on different processors, using different programming languages, hardware platforms, and network protocols, most likely over wireless links. We need to provide real-time communication among these distributed components while dealing with tight constraints on bandwidth, response time, and reliability.
- Dynamic reconfiguration: We need to support on-line switching of algorithmic components and rapid redirection of the interconnections among them, as well as changing the priorities at which information is flowing.
- Real-time quality of service (QoS) guarantees are needed to enable predictable local and distributed communication for control and data acquisition. Adaptive scheduling for optimum resource utilization is essential.

Real-time Distributed Computing

The OCP uses the Common Object Request Broker Architecture (CORBA) [7] to achieve seamless distributed communication between components running on different processors, using different programming languages, hardware platforms and network protocols while dealing with tight constraints on bandwidth, response time and reliability. Components interact through an Object Request Broker (ORB). Using an ORB, a client can transparently invoke a method on a server object which can be on the same machine or across

a network. The ORB intercepts the call and is responsible for finding an object that can implement the request, pass it the parameters, invoke its method and return the results. The client does not have to be aware of where the object is located, its programming language, its operating system or any other system aspects that are not part of an object's interface. Therefore, the ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

The OCP uses an event service [8] that enables components to interact without being tightly coupled. This eases architectural evolution and facilitates on-line reconfiguration. The event service provides a communication abstraction, called an event channel, similar to a bus. Components that generate data ("suppliers") or use data ("consumers") connect to the event channel and the suppliers "publish" certain data event types while the consumers "subscribe" to certain event types.

The event channel acts as a Mediator [9] between components so that their interconnections are flexible. The event channel provides the level of abstraction needed by mediating information flow between suppliers and consumers. When a new type of sensor is added to the system or replaces another type of sensor, for example, it can be connected to the event channel to publish its type of data and all consumers subscribed to that data will receive it. The event channel helps to minimize the architectural impact of switching components by localizing the changes needed.

The OCP also allows control system developers to specify real-time QoS requirements for components (such as event frequency, period and priority). The event service used by the OCP provides QoS specification and enforcement through preemptive real-time scheduling of all system components, including processors, memory units, network interfaces, and network bandwidth. This capability is essential in pushing the limits of controlling UAVs while maintaining efficient resource utilization.

Virtual Resource Network

The virtual resource network (VRN) is an abstraction used to simplify configuration and reconfiguration of system components during a mission. The overall objective in configuring control software for a mission is to define the interactions between the various components in the system in a systematic fashion, where the components of the system are treated as network resources at multiple levels of granularity. For example, in the UAV application, reconfiguration could be at the mission level, the sensor level or anywhere in between.

The ability of these resources to be flexibly interchanged without changing other components in the system comes from the standardization of the interfaces for each component. The standardization of the interfaces are achieved using the CORBA Interface Definition Language [7]. It is also possible for any other resource on the network to use a resource address to connect directly and interact with the resource, irrespective of where it exists on the network. Thus, all distributed computing specifics are abstracted out via the VRN. Also, the VRN is used to define how the resources interact, by defining the input/output relationships among the components.

Reconfiguration Management

The software infrastructure supports a loose coupling between control system components to provide flexibility, extensibility and reuse. The VRN provides a layer of abstraction for easily specifying configurations and reconfigurations with localized changes to the VRN representation. However, a configuration management mechanism is needed to ensure that the configurations, specified in the VRN, are valid and consistent with functional and nonfunctional requirements (such as performance, security and reliability). Moreover, it is critical that changes to the configuration maintain overall system integrity by being globally coordinated and consistent.

The OCP takes an architecture-oriented approach to reconfiguration management by identifying and exploiting system configuration patterns and reconfiguration strategies specific to

the real-time controls domain. Reconfiguration of control systems will follow standard strategies for making changes without violating reliability, safety and consistency constraints. These strategies may dictate how quickly one algorithm can be switched for another or whether a redundant component needs to work concurrently with the component is it replacing before the swap occurs to allow the new component to "come up to speed". For example, the adaptive mode transition control algorithm discussed in this paper may be the primary mechanism used in a control system to gracefully transition between modes. However, it has a certain operating range that on rare occasions may be exceeded. In these cases, we may want the control system to fall back on a traditional gain scheduling control algorithm which transitions through a fixed trajectory of local controllers. It may not be possible for the original mode transitioning algorithm to be switched abruptly to the gain scheduling one. Instead, we may need a gradual switch from the primary control algorithm to the other.

A beneficial synergy exists in the concurrent development of the mode control algorithms and of the OCP. While the adaptive mode transition control algorithm is enabled by the OCP, it is also driving the underlying component-based software technology forward by demanding new OCP capabilities to support on-line customization and extreme performance.

Conclusions

An adaptation scheme is proposed for the online adaptation of mode transition controllers designed via the blending local mode controllers approach. And, a software architecture used for integrating the control algorithm with other system components is discussed.

Acknowledgements

The authors would like to thank Dr. Helen Gill, DARPA, and Mr. Bill Koenig, AFRL, for the sponsorship of this research under DARPA contract number F33615-98-C-1341. In addition, the authors would like to thank their Co-P.I.'s on this research project: Drs. Daniel Schrage and J.V.R. Prasad, School of Aerospace Engineering, Georgia Tech; Dr. Linda Wills, School of Electrical and Computer Engineering, Georgia Tech; graduate students Suresh Kannan, Sam Sander, and Ilkay Yavrucuk; and Mr. Bryan Doerr and Mr. Brian Mendel, Boeing Phantom Works.

References

[1] Dimiter, D., R. Palm, U. Rehfuess, 1996, *A Takagi-Sugeno Fuzzy Gain-Scheduler*, Vol. 2, IEEE International Conference on Fuzzy Systems, pp. 1053-1059.

[2] Rugh, W.J., 1991, *Analytical Framework for Gain Scheduling*, Vol. 11, IEEE Control Systems Magazine, pp. 79-84.

[3] Slotine, J.E., 1984, *Sliding Controller Design for Nonlinear Systems*, Vol. 40, International Journal of Control, pp. 421-434.

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

[5] Rufus, F., S. Clements, S. Sander, B. Heck, L. Wills, G. Vachtsevanos, 1999, *Software-Enabled Control Technologies for Autonomous Aerial Vehicles*, Vol. 2, 18th Digital Avionics System Conference, pp. 6.A.5-1 - 6.A.5-8.

[6] Theocharis, J., G. Vachtsevanos, 1996, *Adaptive Fuzzy Neural Networks as Identifiers of Discrete-Time Nonlinear Dynamic Systems*, Vol. 17, Journal of Intelligent and Robotic Systems, pp. 119-168.

[7] Object Management Group, December 1998, *CORBA 2.2 Common Object Services Specification*, http://www.omg.org.

[8] Levine, D., Sumedh Mungee, Doug Schmidt, April 1998, *The design and performance of real-time object request brokers*, Vol. 21, Computer Communications.

[9] Johnson, R., E. Gamma, R. Helm, J. Vlissides, 1998, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.