

# Improved Adaptive–Reinforcement Learning Control for Morphing Unmanned Air Vehicles

John Valasek, *Senior Member, IEEE*, James Doebbler, Monish D. Tandale, and Andrew J. Meade

**Abstract**—This paper presents an improved Adaptive–Reinforcement Learning Control methodology for the problem of unmanned air vehicle morphing control. The reinforcement learning morphing control function that learns the optimal shape change policy is integrated with an adaptive dynamic inversion control trajectory tracking function. An episodic unsupervised learning simulation using the Q-learning method is developed to replace an earlier and less accurate Actor-Critic algorithm. Sequential Function Approximation, a Galerkin-based scattered data approximation scheme, replaces a K-Nearest Neighbors (KNN) method and is used to generalize the learning from previously experienced quantized states and actions to the continuous state-action space, all of which may not have been experienced before. The improved method showed smaller errors and improved learning of the optimal shape compared to the KNN.

**Index Terms**—Adaptive control, approximation methods, learning control systems, shape control, unmanned air vehicles.

## I. INTRODUCTION

MORPHING research has led to a series of breakthroughs in a wide variety of disciplines that, when fully realized for air vehicle applications, have the potential to produce large increments in aviation safety, affordability, and environmental compatibility. Valasek *et al.* developed an Adaptive–Reinforcement Learning Control (A-RLC) methodology to the morphing air vehicle control problem [1]. Structured Adaptive Model Inversion (SAMI) was used as the controller for tracking trajectories and handling time-varying properties, parametric uncertainties, unmodeled dynamics, and disturbances. A reinforcement learning (RL) module using an Actor-Critic algorithm was used to learn how to produce the optimal shape at every flight condition. While the A-RLC methodology worked well, the learning was found to be dependent on the performance of the function approximator. The

learning itself was found to be decoupled from the actual approximation method used; the data to be approximated is sparse in some regions. The K-Nearest Neighbors (KNN) is not accurate in those regions because it is an “averager”-type approximator, and at times, this produced significant errors in the achievable morphed shapes.

This paper extends and improves upon the methods and results in [1] by using improved nonlinear Shape Memory Alloy (SMA) dynamics, a Q-learning algorithm in place of the Actor-Critic algorithm, and a Galerkin Sequential Function Approximation (SFA) in place of the KNN function approximation.

## II. MORPHING AIR VEHICLE MODEL AND SIMULATION

A simplified morphing air vehicle model in the shape of an ellipsoid is used to obtain the results in this paper. The morphing used in this paper involves a change in the dimensions of the ellipsoid axes while maintaining a constant total volume. The RL module specifies the  $y$ - and  $z$ -axis dimensions, corresponding to the current flight condition, and the  $x$  dimension is calculated by enforcing the constant volume condition  $x = 6Vol/\pi yz$ . The ellipsoidal air vehicle is composed of an SMA whose shape can be modulated by applying voltage. The morphing dynamics are nonlinear differential equations given by

$$\ddot{y} + 2.5\dot{y} + 2.5y + 0.4 \sin(\pi(y - 2)) - 5 = Volt_y \quad (1)$$

$$\ddot{z} + 1.8\dot{z} + 2z + 0.6(z - 2)(z - 4) - 4 = Volt_z. \quad (2)$$

The model for relating the  $y$  and  $z$  dimensions to the applied voltages is given by (1) and (2). Note that the coefficients are arbitrarily selected to form a conceptual model for the morphing dynamics. The optimal  $y$  and  $z$  dimensions, respectively, are arbitrarily selected to be nonlinear functions of the flight condition  $F$ , i.e.,

$$S_y(F) = 3 + \cos\left(\frac{\pi}{5}F\right) \quad (3)$$

$$S_z(F) = 2 + 2e^{-0.5F}. \quad (4)$$

With the optimal  $y$  and  $z$  dimensions given by (3) and (4), the costs associated with the  $y$  and  $z$  dimensions are summed to give the total cost

$$J = J_y + J_z = (y - S_y(F))^2 + (z - S_z(F))^2 \quad (5)$$

where  $y$  and  $z$  are any arbitrary dimensions. For the simulation, flight conditions are specified at various locations along a predesignated flight path. For this simplified example, the optimal shapes are not correlated to the flight path but depend only on the flight condition.

Manuscript received July 31, 2007. The morphing research was supported by the National Aeronautics and Space Administration (NASA) under Award NCC-1-02038, and the sequential function approximation work was provided by the NASA Ames Research Center under Grant NCC-2-8077 and NASA Cooperative Agreement NCC-1-02038. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration. This paper was recommended by Guest Editor F. Lewis.

J. Valasek and J. Doebbler are with the Department of Aerospace Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: valasek@tamu.edu; james.doebbler@tamu.edu).

M. D. Tandale is with Optimal Synthesis, Palo Alto, CA 94303 USA (e-mail: monish@optisyn.com).

A. J. Meade is with the Department of Mechanical Engineering and Materials Science, William Marsh Rice University, Houston, TX 77251-1892 USA (e-mail: meade@rice.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSMCB.2008.922018

### A. Dynamical Model of Morphing Air Vehicle

The dynamic behavior of the morphing air vehicle is modeled by nonlinear six degree-of-freedom equations written in two coordinate systems: an inertial axis system, and a body fixed axis with origin at the center of mass of the air vehicle. The states  $d_x$ ,  $d_y$ , and  $d_z$  are the positions of the center of mass of the morphing air vehicle along the inertial  $X_N$ ,  $Y_N$ , and  $Z_N$  axes, and  $\phi$ ,  $\theta$ , and  $\psi$  are the 3–2–1 Euler angles that give the relative orientation of the body axis with the inertial axis. The acceleration level states are the body axis linear velocities  $u$ ,  $v$ , and  $w$ , and the body axis angular velocities  $p$ ,  $q$ , and  $r$ .

Let  $\mathbf{p}_c = [d_x \ d_y \ d_z]^T$ ,  $\mathbf{v}_c = [u \ v \ w]^T$ ,  $\boldsymbol{\sigma} = [\phi \ \theta \ \psi]^T$ , and  $\boldsymbol{\omega} = [p \ q \ r]^T$

The kinematic states and the acceleration states are related by the differential equations

$$\dot{\mathbf{p}}_c = J_l \mathbf{v}_c \quad (6)$$

$$\dot{\boldsymbol{\sigma}} = J_a \boldsymbol{\omega} \quad (7)$$

where

$$J_l = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix}$$

$$J_a = \begin{bmatrix} 1 & S_\phi \tan(\theta) & C_\phi \tan(\theta) \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi \sec(\theta) & C_\phi \sec(\theta) \end{bmatrix} \quad (8)$$

and  $C_\phi = \cos(\phi)$ ,  $S_\theta = \sin(\theta)$ , and so on. The acceleration level differential equations are

$$m\dot{\mathbf{v}}_c + \tilde{\boldsymbol{\omega}} m \mathbf{v}_c = \mathbf{F} + \mathbf{F}_d \quad (9)$$

$$I\dot{\boldsymbol{\omega}} + \dot{I}\boldsymbol{\omega} + \tilde{\boldsymbol{\omega}} I \boldsymbol{\omega} = \mathbf{M} + \mathbf{M}_d \quad (10)$$

where  $m$  is the mass of the morphing air vehicle,  $\mathbf{F}$  is the control force,  $\mathbf{F}_d$  is the drag force,  $I$  is the body axis moment of inertia,  $\mathbf{M}$  is the control torque,  $\mathbf{M}_d$  is the drag moment, and  $\tilde{\boldsymbol{\omega}} \mathbf{V}$  is the matrix representation of the cross product between vector  $\boldsymbol{\omega}$  and vector  $\mathbf{V}$ , where

$$\tilde{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}. \quad (11)$$

For this paper, the motion of the morphing air vehicle is simulated in the absence of gravity. Equation (10) has an additional term  $\dot{I}\boldsymbol{\omega}$  that is a consequence of the shape change and is responsible for speeding up or slowing down the rotation of the air vehicle due to the time rate of change in the moment of inertia about a particular axis.

The drag force  $\mathbf{F}_d$  and the drag moment  $\mathbf{M}_d$  are modeled as functions of the air density  $\rho$ , the square of the velocity along the axis, and the projected area of the ellipsoidal air vehicle perpendicular and parallel to the axis, respectively. The dimensions of the ellipsoid axes along the body axis are  $x$ ,  $y$ , and  $z$ , respectively. We have

$$\mathbf{F}_d = \frac{-\rho\pi}{8} \begin{bmatrix} u^2 \text{sgn}(u)yz \\ v^2 \text{sgn}(v)xz \\ w^2 \text{sgn}(w)xy \end{bmatrix} \quad (12)$$

$$\mathbf{M}_d = \frac{-\rho\pi}{8} \begin{bmatrix} p^2 \text{sgn}(p)x(y+z) \\ q^2 \text{sgn}(q)y(x+z) \\ r^2 \text{sgn}(r)z(x+y) \end{bmatrix}. \quad (13)$$

### B. Trajectory Generation

The reference trajectory is arbitrary and is generated as a combination of straight lines and sinusoidal curves. The total flight path is divided into an odd number of segments of variable lengths. During every odd-numbered segment, the  $d_y$  and  $d_z$  locations remain constant while their values are generated by a random function. The even-numbered segments connect the trajectories in two adjacent sections with smooth sinusoidal curves. The morphing air vehicle flies along the flight trajectory with a constant inertial velocity along the  $X_N$  inertial axis. For the attitude reference, it tracks prescribed sinusoidal roll oscillations along the  $X_N$  inertial axis.

## III. RL MODULE

The RL module's objective is to learn the optimal control policy for a specific flight condition, which commands the optimal voltage to achieve  $S_y$  and  $S_z$  over the entire flight trajectory. The mapping from state  $s_t \in S$ , where  $S$  is a set of possible states, to the probabilities of selecting each possible action  $a_t \in A(s_t)$ , where  $A(s_t)$  is a set of actions available in state  $s(t)$  at time  $t$ , is the agent's policy  $\pi_t(s, a)$ . It indicates the probability that  $a_t = a$  given  $s_t = s$  at  $t$ . The agent's goal is to find the optimal policy  $\pi^*$  that has the optimal state-value function  $V^*(s) = \max_{\pi} V^{\pi}(s)$  and the optimal action-value function  $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$ . In many applications of RL to control tasks, the state space is too large to enumerate the value function, so function approximators must be used to compactly represent the value function. For the current problem, the RL module has no prior knowledge of the relationship between voltages, as defined by the voltage functions on the right-hand side (RHS) of (1) and (2), and the dimensions of the morphing air vehicle. Neither does it know the relationship between the flight conditions, costs, and optimal shapes. However, it does know all possible voltages that can be applied and has accurate real-time information of the vehicle shape, present flight condition, and the current cost provided by a variety of sensors.

The RL algorithm used here is the one-step Q-learning method, which is a common off-policy temporal difference control algorithm [2].

- Initialize  $Q(s, a)$  arbitrarily
- Repeat (for each episode)
  - Initialize  $s$
  - Repeat (for each step of the episode)
    - \* Choose  $a$  from  $s$  using policy derived from  $Q(s, a)$  (e.g.,  $\epsilon$ -Greedy Policy)
    - \* Take action  $a$ , observe  $r, s'$
    - \*  $Q(s, a) \leftarrow Q(s, a) + \alpha\{r + \gamma \max_{a'} Q(s', a') - Q(s, a)\}$
    - \*  $s \leftarrow s'$
    - until  $s$  is terminal
- return  $Q(s, a)$

In this algorithm,  $\gamma$  is the discount rate parameter. As the learning episodes increase, the learned action-value function  $Q(s, a)$  asymptotically converges to the optimal action-value function

$Q^*(s, a)$ . The method is an off-policy one as it evaluates the target policy following another policy. Here, the  $\epsilon$ -greedy policy is used, in which the action  $a$  with the maximum  $Q(s, a)$  is selected with probability  $1-\epsilon$ ; otherwise, a random action is selected. In the current approach,  $\epsilon$  is gradually decreased as the learning goes on, which allows the agent to exploit its learning result as the number of episodes increase. The RL problem is decoupled into two independent problems that correspond to the  $y$  and  $z$  dimensions, respectively. For both dimensions,  $S = [2, 4] \times [0, 5]$  is a 2-D continuous Cartesian space consisting of all the possible states. For each state,  $A(s) = \{-1.0, -0.5, 0.0, 0.5, 1.0\}$  are all possible voltages that can be applied. The training is conducted at six discrete flight conditions  $F = \{0, 1, 2, 3, 4, 5\}$ , so the cost  $J : [2, 4] \times [2, 4] \times [0, 5] \rightarrow [0, \infty]$ . For each flight condition, the number of samples for approximating the action-value function is 2000. The RL agent runs ten iterations on these samples to update their  $Q$  values.

#### IV. FUNCTION APPROXIMATION

The Q-learning algorithm introduced above can be used for either continuous or discrete action sets. The action-value function  $Q(s, a)$  determined by the RL module is sampled at discrete states, not discretized, but the shape of the morphing air vehicle is on a continuous domain with inputs that are discrete voltages. The continuous space cannot be accurately implemented and learned, so for computational purposes, the discrete implementation was chosen. Thus, the action-value function must be interpolated and approximated in order to apply the learned information. The authors' earlier work used the KNN method, which computes a weighted average of the KNN in the sampled space [1]. As will be shown in Section VII, the KNN performed mediocre at best for this application since the method is susceptible to the absence of accurate information near the desired point. To obtain improved function approximation, the Galerkin SFA method is used because it is an adaptive and matrix-free scheme that is useful for interpolating and approximating sparse multidimensional scattered data.

##### A. SFA

The development of this method was motivated by the similarities between iterative optimization procedures and the method of weighted residuals, specifically the Galerkin method [3]. To begin, we can write the function residual as

$$\begin{aligned} r(c_n, \boldsymbol{\xi}, \boldsymbol{\beta}_n) &= r_n \\ &= u(\boldsymbol{\xi}) - u_n^a(\boldsymbol{\xi}) \\ &= u(\boldsymbol{\xi}) - u_{(n-1)}^a(\boldsymbol{\xi}) - c_n \phi(\eta(\boldsymbol{\xi}, \boldsymbol{\beta}_n)) \\ &= r_{(n-1)} - c_n \phi(\eta(\boldsymbol{\xi}, \boldsymbol{\beta}_n)) \\ &= r_{(n-1)} - c_n \phi_n. \end{aligned}$$

Utilizing the Petrov–Galerkin approach, we select a  $c_n$  that will force the function residual to be orthogonal to the basis function

$$\langle r_n, \phi_n \rangle = - \left\langle r_n, \frac{\partial r_n}{\partial c_n} \right\rangle = - \frac{1}{2} \frac{\partial \langle r_n, r_n \rangle}{\partial c_n} = 0$$

which is equivalent to selecting a value of  $c_n$  that will minimize  $\langle r_n, r_n \rangle$  or

$$c_n = \frac{\langle \phi_n, r_{(n-1)} \rangle}{\langle \phi_n, \phi_n \rangle}. \quad (14)$$

Using (14), the values of our remaining variables ( $\boldsymbol{\beta}_n$ ) must minimize as

$$\langle r_n, r_n \rangle = \langle r_{(n-1)}, r_{(n-1)} \rangle \left( 1 - \frac{\langle \phi_n, r_{(n-1)} \rangle^2}{\langle \phi_n, \phi_n \rangle \langle r_{(n-1)}, r_{(n-1)} \rangle} \right). \quad (15)$$

Recalling the definition of the cosine using arbitrary functions  $f$  and  $v$

$$\cos(\theta) = \frac{\langle f, v \rangle}{\langle f, f \rangle^{1/2} \langle v, v \rangle^{1/2}}$$

and the equivalence between the inner product and the square of the  $L_2$  norm, (15) can be written as

$$\|r_n\|_2^2 = \|r_{(n-1)}\|_2^2 \sin^2(\theta_n) \quad (16)$$

where  $\theta_n$  is the angle between  $\phi_n$  and  $r_{(n-1)}$ . With (16),  $\|r_n\|_2 < \|r_{(n-1)}\|_2$  as long as  $\phi_n$  is not orthogonal to the previous equation residual  $r_{(n-1)}$ , that is,  $\theta_n \neq \pi/2$ . To force  $\|r_n\|_2 \rightarrow 0$ , a low-dimensional function approximation problem must be solved at each stage  $n$ . This involves an unconstrained nonlinear optimization in the determination  $\boldsymbol{\beta}_N$ . The dimensionality of the nonlinear optimization problem is kept low since we are solving for only one basis at a time.

In this paper, we assumed that the underlying function was smooth but could only be measured as a positive integer between 0 and 5. For example, 0.49 was measured as 0, and 2.5 was measured as 3. Consequently, we used radial basis functions (RBFs) to approximate the underlying function since it is smooth and has the fewest parameters to optimize, i.e., the center and the width. When using RBFs in this implementation of the SFA method, then the upper bound of the residual should be

$$\langle r_n, r_n \rangle = \|r_n\|_2^2 \leq C a^{-n} \quad (17)$$

as per the results of [4], where  $0 \leq a \leq 1$ . In this implementation, (17) indicates that we should observe an exponential convergence that does not explicitly depend on the dimensionality of the approximation. It sidesteps the ‘‘curse of dimensionality’’ [5]. A combination of three criteria can be used to terminate the SFA algorithm:  $\|r_n\|_2^2$ , or  $|c_n|$  falls below a user-specified tolerance ( $\tau$ ), or the number of bases  $n$  exceeds the user-specified maximum. In this paper, we terminate the calculations when either  $|c_n| \leq \tau$  or  $n \geq s$ , and we have measured our approximation  $u_n^a$  to the nearest integer value.

#### V. SAMI CONTROL

The SAMI controller [6] is used here to track the reference trajectories, even when the dynamic properties of the air vehicle change due to morphing. This controller is designed to drive the error between the output of the actual plant and the reference trajectories to zero, with prescribed error dynamics. The SAMI approach has also been extended to handle actuator failures

and to facilitate correct adaptation in the presence of actuator saturation [7], [8]. Without the angular drag and the  $\dot{I}\omega$  term, (8) and (10) can be manipulated to obtain

$$I_a^*(\sigma)\ddot{\sigma} + C_a^*(\sigma, \dot{\sigma})\dot{\sigma} = P_a^T(\sigma)\mathbf{M} \quad (18)$$

where the matrices  $I_a^*(\sigma)$ ,  $C_a^*(\sigma, \dot{\sigma})$ , and  $P(\sigma)$  are defined as

$$P_a(\sigma) \triangleq J_a^{-1}(\sigma) \quad (19)$$

$$I_a^*(\sigma) \triangleq P_a^T I P_a \quad (20)$$

$$C_a^*(\sigma, \dot{\sigma}) \triangleq -I_a^* \dot{J}_a P_a + P_a^T [\widetilde{P}_a \dot{\sigma}] I P_a. \quad (21)$$

The left-hand side of (18) can be linearly parameterized as

$$I_a^*(\sigma)\ddot{\sigma} + C_a^*(\sigma, \dot{\sigma})\dot{\sigma} = Y_a(\sigma, \dot{\sigma}, \ddot{\sigma})\theta \quad (22)$$

where  $Y_a(\sigma, \dot{\sigma}, \ddot{\sigma})$  is a regression matrix, and  $\theta$  is the constant inertia parameter vector defined as  $\theta \triangleq [I_{11} \ I_{22} \ I_{33} \ I_{12} \ I_{13} \ I_{23}]^T$ . It can be seen that the product of the inertia matrix and a vector can be written as

$$I\nu = \Lambda(\nu)\theta \quad \forall \nu \in \mathbb{R}^3 \quad (23)$$

where  $\Lambda \in \mathbb{R}^{3 \times 6}$  is defined as

$$\Lambda(\nu) \triangleq \begin{bmatrix} \nu_1 & 0 & 0 & \nu_2 & \nu_3 & 0 \\ 0 & \nu_2 & 0 & \nu_1 & 0 & \nu_3 \\ 0 & 0 & \nu_3 & 0 & \nu_1 & \nu_2 \end{bmatrix}. \quad (24)$$

The terms on the left-hand side of (18) can be written as

$$\begin{aligned} I_a^* \ddot{\sigma} &= P_a^T I P_a \ddot{\sigma} \\ &= P_a^T \Lambda(P_a \ddot{\sigma}) \end{aligned} \quad (25)$$

$$\begin{aligned} C_a^* \dot{\sigma} &= -P_a^T I P_a \dot{J}_a P_a \dot{\sigma} + P_a^T [\widetilde{P}_a \dot{\sigma}] I P_a \dot{\sigma} \\ &= P_a^T \left\{ -\Lambda(P_a \dot{J}_a P_a \dot{\sigma}) + [\widetilde{P}_a \dot{\sigma}] \Lambda(P_a \dot{\sigma}) \right\} \theta. \end{aligned} \quad (26)$$

Combining (25) and (26), we have the linear minimal parameterization for the inertia matrix [9]

$$\begin{aligned} I_a^*(\sigma)\ddot{\sigma} + C_a^*(\sigma, \dot{\sigma})\dot{\sigma} &= P_a^T \left\{ \Lambda(P_a \ddot{\sigma}) - \Lambda(P_a \dot{J}_a P_a \dot{\sigma}) \right. \\ &\quad \left. + [\widetilde{P}_a \dot{\sigma}] \Lambda(P_a \dot{\sigma}) \right\} \theta \\ &= Y_a(\sigma, \dot{\sigma}, \ddot{\sigma})\theta. \end{aligned} \quad (27)$$

The attitude tracking problem can be formulated as follows. The control objective is to track an attitude trajectory in terms of the 3–2–1 Euler angles. The desired reference trajectory is assumed to be twice differentiable with respect to time. Let  $\epsilon \triangleq \sigma - \sigma_r$  be the tracking error. Differentiating twice and multiplying by  $I_a^*$  throughout, we have

$$I_a^* \ddot{\epsilon} = I_a^* \ddot{\sigma} - I_a^* \ddot{\sigma}_r. \quad (28)$$

Adding  $(C_{da} + C_a^*(\sigma, \dot{\sigma}))\dot{\epsilon} + K_{da}\epsilon$  on both sides, where  $C_{da}$  and  $K_{da}$  are the design matrices, we have

$$\begin{aligned} I_a^* \ddot{\epsilon} + (C_{da} + C_a^*(\sigma, \dot{\sigma}))\dot{\epsilon} + K_{da}\epsilon \\ = I_a^* \ddot{\sigma} - I_a^* \ddot{\sigma}_r + (C_{da} + C_a^*(\sigma, \dot{\sigma}))\dot{\epsilon} + K_{da}\epsilon. \end{aligned} \quad (29)$$

The RHS of (29) can be written as

$$(I_a^* \ddot{\sigma} + C_a^*(\sigma, \dot{\sigma})\dot{\sigma}) - (I_a^* \ddot{\sigma}_r + C_a^*(\sigma, \dot{\sigma})\dot{\sigma}_r) + C_{da}\dot{\epsilon} + K_{da}\epsilon. \quad (30)$$

From (18) and the construction of  $Y$  similar to (27), the RHS of (29) can be further written as

$$P_a^T \mathbf{M} - Y_a(\sigma, \dot{\sigma}, \ddot{\sigma}, \ddot{\sigma}_r)\theta + C_{da}\dot{\epsilon} + K_{da}\epsilon. \quad (31)$$

So the control law can now be chosen as

$$\mathbf{M} = P_a^{-T} \{ Y_a(\sigma, \dot{\sigma}, \ddot{\sigma}, \ddot{\sigma}_r)\theta - C_{da}\dot{\epsilon} - K_{da}\epsilon \}. \quad (32)$$

The above control law requires that the inertia parameters  $\theta$  be accurately known, but they may not be accurately known in actual practice. So by using the certainty equivalence principle [10], adaptive estimates for the inertia parameters  $\hat{\theta}$  will be used for calculating the control

$$\mathbf{M} = P_a^{-T} \left\{ Y_a(\sigma, \dot{\sigma}, \ddot{\sigma}, \ddot{\sigma}_r)\hat{\theta} - C_{da}\dot{\epsilon} - K_{da}\epsilon \right\}. \quad (33)$$

With the control law given in (33), the closed-loop dynamics takes the following form:

$$I_a^* \ddot{\epsilon} + (C_{da} + C_a^*(\sigma, \dot{\sigma}))\dot{\epsilon} + K_{da}\epsilon = Y_a(\sigma, \dot{\sigma}, \ddot{\sigma})\tilde{\theta} \quad (34)$$

where  $\tilde{\theta} = \hat{\theta} - \theta$ . Now consider the candidate Lyapunov function

$$V = \frac{1}{2} \dot{\epsilon}^T I_a^* \dot{\epsilon} + \frac{1}{2} \epsilon^T K_{da} \epsilon + \frac{1}{2} \tilde{\theta}^T \tau^{-1} \tilde{\theta} \quad (35)$$

where  $\tau^{-1}$  is a symmetric positive definite gain matrix. Taking the derivative of the Lyapunov function along the closed-loop trajectories given by (34), and on further simplification, we have

$$\begin{aligned} \dot{V} &= \dot{\epsilon}^T \left[ \frac{1}{2} \dot{I}_a^* - C_a^* \right] \dot{\epsilon} - \dot{\epsilon}^T C_{da} \dot{\epsilon} \\ &\quad + \left( \dot{\epsilon}^T Y_a(\sigma, \dot{\sigma}, \ddot{\sigma}, \ddot{\sigma}_r) + \dot{\tilde{\theta}}^T \tau^{-1} \right) \tilde{\theta}. \end{aligned} \quad (36)$$

The first term vanishes because  $[(1/2)\dot{I}_a^* - C_a^*]$  is skew symmetric. Setting the coefficient of  $\tilde{\theta}$  to 0, we obtain the adaptive laws

$$\dot{\tilde{\theta}} = -\tau Y_a(\sigma, \dot{\sigma}, \ddot{\sigma}, \ddot{\sigma}_r)^T \dot{\epsilon}. \quad (37)$$

From the definition of  $\tilde{\theta}$  and assuming that the true parameter  $\theta$  remains constant, we have

$$\dot{\tilde{\theta}} = -\tau Y_a(\sigma, \dot{\sigma}, \ddot{\sigma}, \ddot{\sigma}_r)^T \dot{\epsilon}. \quad (38)$$

This update law renders the derivative of the Lyapunov function

$$\dot{V} = -\dot{\epsilon}^T C_{da} \dot{\epsilon}. \quad (39)$$

Thus, the derivative is negative semidefinite. From (35) and (39), it can be concluded that  $\epsilon$ ,  $\dot{\epsilon}$ , and  $\theta$  are bounded. Using the standard procedure of application of Barbalat's lemma [10], asymptotic stability of the tracking error dynamics can be concluded for bounded reference trajectories. Following similar

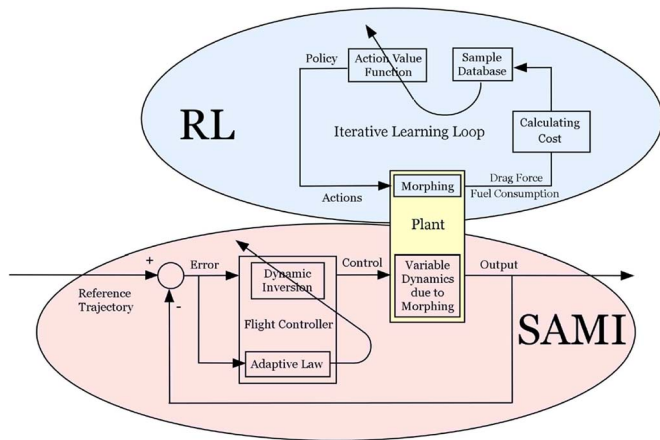


Fig. 1. A-RLC architecture.

lines as the attitude controller, a SAMI controller for the control of the linear states of the air vehicle can also be implemented.

## VI. A-RLC ARCHITECTURE FUNCTIONALITY

The A-RLC architecture is composed of two subsystems, i.e., RL and SAMI (Fig. 1). The two subsystems significantly interact during both the episodic learning stage, when the optimal shape change policy is learned, and the operational stage, when the plant morphs and tracks a trajectory. The RL module initially commands an arbitrary action from the set of admissible actions. This action is sent to the plant, which produces a shape change, which also causes the plant dynamics to change. The SAMI controller maintains trajectory tracking irrespective of the changing dynamics of the plant due to these shape changes. The cost associated with the resultant shape change, in terms of drag force and fuel consumption, is evaluated with the cost function. For the next step, a new action is generated based on the current policy and the current action-value function, and the sequence repeats itself.

## VII. NUMERICAL SIMULATION EXAMPLE

The numerical example demonstrates the realistic dynamics and improved function approximation scheme used with the A-RLC architecture. The measures of merit are tracking performance and comparison of the total rms error between the actual shape and the learned shape. The learning takes place over 200 unsupervised learning episodes, each consisting of a single 200-s transit through a 100-m-long path. The sequence of the flight conditions is random and changes twice during each episode. For learning purposes, new shape change commands are issued at 10-s intervals. The reference trajectory to be tracked in each episode is arbitrarily generated. The trajectory tracking performance is demonstrated with a single pass through the path, with a randomly generated reference trajectory and an arbitrary flight condition change at approximately 50-s intervals.

### A. Learning Performance

Fig. 2 compares the optimal shape with the actual shape achieved after learning for both the KNN method and the SFA

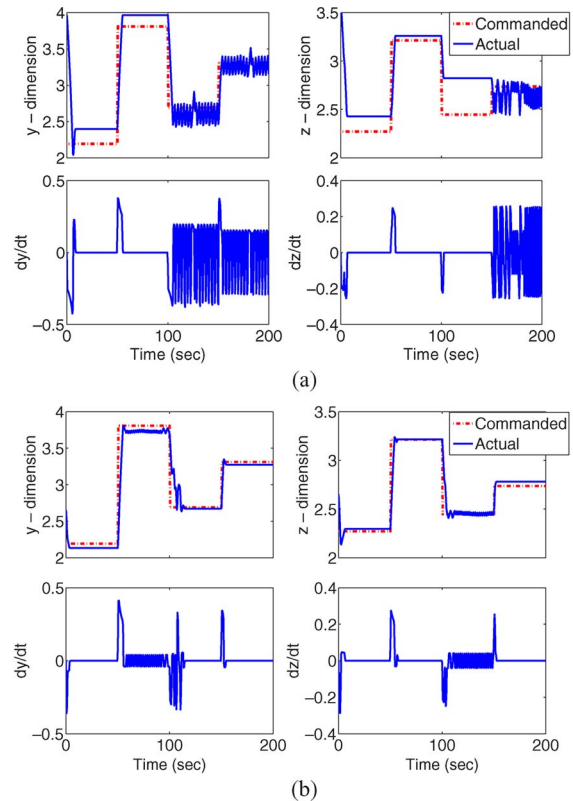


Fig. 2. Comparison of function approximation techniques. (a) Optimal and learned shapes using the KNN method. (b) Optimal and learned shapes using the SFA method.

TABLE I  
FUNCTION APPROXIMATION NORMALIZED RMS ERRORS

Method	Y dimension	Z dimension
KNN	1.42	0.821
SFA	1.27	0.661
Reduction	10%	20%

method, showing that the SFA method more closely matches the optimal shape in both dimensions and at all flight conditions. The optimal action-value function for the continuous domain is not known, so the performance of the two function approximation methods cannot be directly compared. However, the sampled data points are known, and each method learns these points very well. The difference in the quality of approximation lies in the data points that are not near the sampled data points, so the two methods are compared by examining errors between the obtained shape and the true shape. Using normalized error values for the entire shape time histories, Table I shows that the numerical simulation using the KNN method had a total rms error of 1.42 in the  $y$  dimension and 0.821 in the  $z$  dimension. The example run using the SFA method had total rms errors of 1.27 and 0.661 in the  $y$  and  $z$  dimensions, respectively. This shows a reduction of the rms error by slightly more than 10% in the  $y$  dimension and nearly 20% in the  $z$  dimension. It is important to note that all of these error values include the transient errors between flight condition changes due to the morphing dynamics, not just the steady-state errors. Since these transient errors are clearly present in the time history

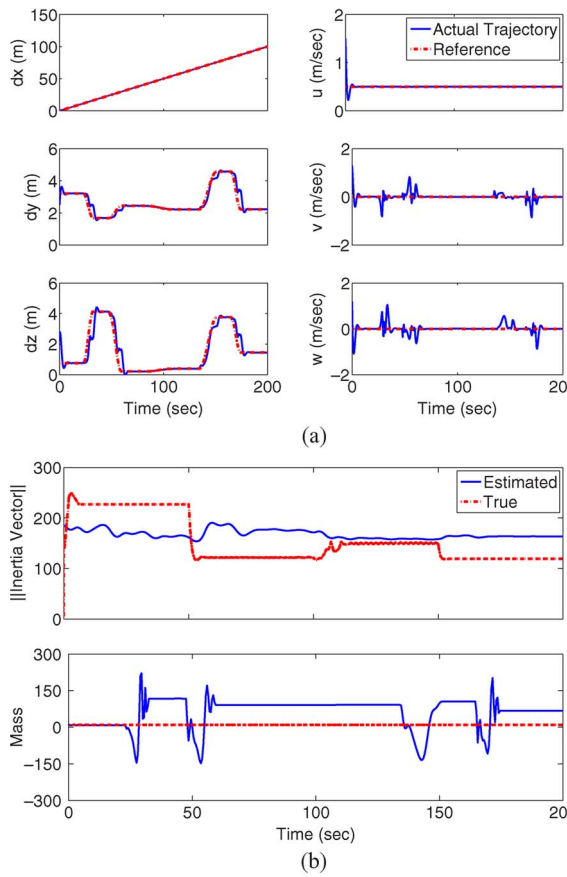


Fig. 3. Trajectory tracking performance time histories. (a) Linear states. (b) Adaptive parameters.

of the obtained shape, but are not dependent on the function approximation method used, the actual percent improvement in the function approximation provided by the SFA method is actually higher than the numbers in Table I indicate.

### B. Tracking Performance

The control objective for the A-RLC controller is to track the reference trajectories irrespective of the initial condition errors, parametric uncertainties, and changes in the dynamic behavior of the air vehicle due to morphing. The deviations from the reference trajectory seen in Fig. 3(a) are due to rapid changes in the morphing dimensions. The adaptive control has to undergo rapid adaptation during a shape change, as seen in Fig. 3(b). The adaptive controller formulation assumes that the true parameters are constant, hence the shape change causes disturbances in the tracking behavior. In the current simulation, the true parameters that are learned are inertia and mass. Mass remains constant, but the inertia changes as the air vehicle morphs into different shapes. The A-RLC controller does not implement the term  $\dot{I}\omega$  in (10), so  $\dot{I}\omega$  acts as an unmodeled dynamics and perturbs the tracking. However, the A-RLC controller is able to maintain adequate tracking performance. Although the A-RLC controller does not explicitly account for the external disturbance due to the drag force, it still provides excellent tracking.

## VIII. CONCLUSION

The method of function approximation used strongly affects the reliability of applying the learned data at points other than those that were sampled during the learning process. Comparing normalized error values, the Galerkin-based SFA reduced the rms error by slightly more than 10% in the  $y$  dimension and nearly 20% in the  $z$  dimension compared to the KNN method. The stability proof of the A-RLC controller guarantees asymptotic stability of the tracking error only for constant true parameters. However, due to the time scale separation and the time interval between two successive shape changes, it behaves like a piecewise constant parametric change for the adaptive controller and is effectively handled.

## REFERENCES

- [1] J. Valasek, M. Tandale, and J. Rong, "A reinforcement learning-adaptive control architecture for morphing," *J. Aerosp. Comput., Inf., Commun.*, vol. 2, no. 4, pp. 174–195, Apr. 2005.
- [2] R. Sutton and A. Barto, *Reinforcement Learning—An Introduction*. Cambridge, MA: MIT Press, 1998.
- [3] C. A. J. Fletcher, *Computational Galerkin Methods*. New York: Springer-Verlag, 1984.
- [4] A. J. Meade and B. Zeldin, "Approximation properties of local bases assembled from neural network transfer functions," *Math. Comput. Model.*, vol. 28, no. 9, pp. 43–62, Nov. 1998.
- [5] D. L. Thomson, "Sequential function approximation of the radiative transfer equation," Ph.D. dissertation, Rice Univ., Houston, TX, 1999.
- [6] K. Subbarao, "Structured adaptive model inversion: Theory and applications to trajectory tracking for non-linear dynamical systems," Ph.D. dissertation, Aerosp. Eng. Dept., Texas A&M Univ., College Station, TX, 2001.
- [7] M. D. Tandale and J. Valasek, "Fault tolerant structured adaptive model inversion control," *J. Guid. Control Dyn.*, vol. 29, no. 3, pp. 635–642, May/Jun. 2006.
- [8] M. D. Tandale and J. Valasek, "Adaptive dynamic inversion control with actuator saturation constraints applied to tracking spacecraft maneuvers," *J. Astronaut. Sci.*, vol. 54, no. 4, pp. 517–530, Oct.–Dec. 2004.
- [9] J. Ahmed, V. T. Coppola, and D. S. Bernstein, "Adaptive asymptotic tracking of spacecraft attitude motion with inertia matrix identification," *J. Guid. Control Dyn.*, vol. 21, no. 5, pp. 684–691, Sep./Oct. 1998.
- [10] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Upper Saddle River, NJ: Prentice-Hall, 1996, ch. 1, pp. 10–11.



**John Valasek** (S'89–M'95–SM'98) earned the B.S. degree in aerospace engineering from California State Polytechnic University, Pomona, in 1986 and the M.S. degree (with honors) and the Ph.D. degree in aerospace engineering from the University of Kansas, Lawrence, in 1991 and 1995, respectively.

From 1985 to 1988, he was a Flight Control Engineer with the Flight Controls Research Group, Aircraft Division, Northrop Corporation, where he worked on the AGM-137 Tri-Services Standoff Attack Missile (TSSAM) Program. He previously held an academic appointment at Western Michigan University during 1995–1997 and was a Summer Faculty Researcher with NASA Langley in 1996 and an AFOSR Summer Faculty Research Fellow with the Air Force Research Laboratory in 1997. Since then, he has been with the Department of Aerospace Engineering, Texas A&M University, College Station, where he is currently an Associate Professor of aerospace engineering and the Director of the Vehicle Systems and Control Laboratory. He teaches courses on digital control, nonlinear systems, vehicle management systems, cockpit systems and displays, and flight mechanics. His current research interests include machine learning and multiagent systems, intelligent autonomous control, vision-based navigation systems, and fault-tolerant adaptive control.

Prof. Valasek is a member of the Control Systems Society and the Education Society. He was an Associate Editor for Dynamics and Control of the IEEE TRANSACTIONS ON EDUCATION (1998–2001).



**James Doebbler** received the B.S. (*magna cum laude*) and M.S. degrees from Texas A&M University, College Station, in 2004 and 2007, respectively, both in aerospace engineering. He is currently working toward the Ph.D. degree in aerospace engineering at Texas A&M University.

Since 2003, he has been a Researcher with the Vehicle Systems and Control Laboratory, Department of Aerospace Engineering, Texas A&M University, and has been an Undergraduate Research Assistant and a Graduate Research Assistant. His research work has included dynamics and control of robotic systems, distributed real-time flight simulation systems, control systems for autonomous aerial refueling, intelligent and computationally efficient path planning methods, and artificial intelligence techniques applied to the control of morphing air vehicles.

Mr. Doebbler is a member of Sigma Gamma Tau, the national aerospace engineering honor society.



**Monish D. Tandale** was born in Mumbai, India. He received the B.E. degree (with distinction) from Victoria Jubilee Technical Institute, University of Mumbai, Mumbai, in 2000 and the M.S. and Ph.D. degrees from Texas A&M University, College Station, in 2002 and 2006, respectively.

He is currently a Research Scientist with Optimal Synthesis, Palo Alto, CA. His research interests include fault-tolerant adaptive control in the presence of actuator saturation, and intelligent control.

Dr. Tandale was a recipient of the Regents Graduate Fellowship from the Texas A&M University and was one of the finalists for the Eppright Outstanding International Student Award in 2003.



**Andrew J. Meade** received the B.S. degree from William Marsh Rice University, Houston, TX, and the M.S. and Ph.D. degrees in mechanical engineering from the University of California, Berkeley.

He is currently a Professor of mechanical engineering with the Department of Mechanical Engineering and Materials Science, William Marsh Rice University. His research primarily focuses on machine learning and its application to experimental and computational fluid dynamics. His current work includes pattern recognition and the fusion of exper-

imental and numerical aerodynamics data.