Morphing Unmanned Air Vehicle Intelligent Shape and Flight Control

John Valasek*, Amanda Lampton† and Monika Marwaha‡

Texas A&M University, College Station, Texas 77843-3141

This paper develops and demonstrates a complete methodology for the control of a morphing unmanned air vehicle. The shape learning is done with a modified episodic Reinforcement Learning algorithm, which employs an adaptive grid to improve the search performance and accuracy of learning the optimal shape change policy. The shape control, which uses Reinforcement Learning, and the trajectory tracking flight control, which uses Structured Adaptive Model Inversion Control, are combined in a technique called Adaptive-Reinforcement Learning Control. Optimality is addressed by cost functions representing optimal shapes corresponding to specified operating conditions. The shape change dynamics are represented by Shape Memory Alloy material hysteresis input-output mappings which were determined experimentally. The nonlinear, six degree-of-freedom unmanned air vehicle dynamical model simulation is integrated with a constant strength source doublet panel method CFD code. This combined 3-D vehicle model code is capable of simulating the dynamic response and the forces and moments which are generated due to commanded multiple large scale shape changes, consisting of thickness, sweep angle, dihedral angle, and chord length. The methodology is demonstrated by a numerical examples of the morphing air vehicle simultaneously tracking a specified trajectory and autonomously morphing over a set of shapes corresponding to flight conditions along the trajectory. Results presented in the paper demonstrate that this methodology is effective for accurately tracking a flight trajectory in the presence of parametric uncertainties and initial error conditions.

*Associate Professor and Director, Vehicle Systems & Control Laboratory, Aerospace Engineering Department. Associate Fellow AIAA. valasek@tamu.edu.
†Graduate Research Assistant, Vehicle Systems & Control Laboratory, Aerospace Engineering Department. Student Member AIAA. alampton@tamu.edu.
‡Graduate Research Assistant, Vehicle Systems & Control Laboratory, Aerospace Engineering Department. Student Member AIAA. monika.marwaha@tamu.edu
Nomenclature

I. Introduction

Morphing research has led to a series of breakthroughs in a wide variety of disciplines that, when fully realized for aircraft applications, have the potential to produce large increments in aircraft system safety, affordability, and environmental compatibility.\textsuperscript{1} There are many recent examples of vehicles in this class of systems that have the potential to impact the aircraft industry. Hong et al. develop an unmanned combat aerial vehicle that use shape changing to meet the requirements of a low-speed, high-altitude loiter and a supersonic strike capable aircraft.\textsuperscript{2} The designed aircraft changes wing area by 200\%, sweep angle from 20° to 70°, and aspect ratio from 3 to 7. The Daedalon, a design baselined for Mars, transforms itself from a blunt-body entry spacecraft into an airplane through a form of wing morphing to allow for a flexible architecture for unmanned planetary exploration.\textsuperscript{3} References\textsuperscript{4,5} consider the impact of incorporating this class of systems into current commercial aircraft service.

Morphing control has also seen much investigation recently. Refs. 6 and 7 consider the case of variable wing dihedral in which a vortex lattice method is used to calculate the aerodynamics. Desired dynamics for each mission phase are chosen and $H_{\infty}$ model-following controllers are developed for each. In contrast to morphing control approaches reported in the literature which focus on structures and actuation, the Adaptive-Reinforcement Learning Control (A-RLC) technique introduced in Ref. 8 addresses the optimal shape changing of an entire vehicle. A-RLC is able to learn the commands that produce the optimal shape, defined as a function of operating condition, while maintaining accurate trajectory tracking of the vehicle. A-RLC uses Structured Adaptive Model Inversion (SAMI) as the trajectory tracking controller for handling time-varying properties, parametric uncertainties, and disturbances. For learning the optimality relations between the operating conditions and the shape and learning how to produce the optimal shape at every operating condition over the life of the vehicle, A-RLC uses Q-Learning. The authors show that A-RLC is able to control a hypothetical 3-D smart aircraft that has two independent morphing parameters, tracking a specified trajectory, and autonomously morphing over a set of shapes corresponding to flight conditions along the trajectory. The methodology is further improved upon by applying Sequential Function Approximation to generalize the learning from previously experienced quantized states and actions to the continuous state-action space.\textsuperscript{9} The authors show that the approximation scheme resulted in marked improvements in the learning as opposed to the previously employed K-Nearest Neighbor approach.

This paper develops and demonstrates a control system architecture which encompasses realistic aerodynamics for a complex configuration, nonlinear dynamics, nonlinear actuator dynamics with hysteresis effect, and an intelligent learning agent coupled with a nonlinear adaptive controller. The paper first develops the Reinforcement Learning module of A-RLC. This module uses the Q-learning method to learn how to morph into specified shapes. The aerodynamic modeling of the morphing wing is briefly described next. This section
details morphing parameters available to manipulate the shape of the wing. This is followed by development of the SAMI control module, which handles morphing wing parametric uncertainties, and initial error conditions, while tracking a trajectory. The A-RLC methodology is demonstrated with a numerical example of the morphing wing air vehicle autonomously morphing over a set of optimal shapes, corresponding to specified flight conditions, while tracking a specified trajectory.

II. Reinforcement Learning Module

A. Overview of Reinforcement Learning

Reinforcement learning (RL) is a method of learning from interaction between an agent and its environment to achieve a goal. The learner and decision-maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. The agent interacts with its environment at each instance of a sequence of discrete time steps, \( t = 0, 1, 2, 3, \ldots \). At each time step \( t \), the agent receives some representation of the environment’s state, \( s_t \in S \), where \( S \) is a set of possible states, and on that basis it selects an action, \( a_t \in A(s_t) \), where \( A(s_t) \) is a set of actions available in state \( s(t) \). One time step later, partially as a consequence of its action, the agent receives a numerical reward, \( r_{t+1} = R \), and finds itself in a new state, \( s_{t+1} \). The mapping from states to probabilities of selecting each possible action at each time step is denoted by \( \pi \) and is called the agent’s policy. Thus \( \pi_t(s, a) \) indicates the probability that \( a_t = a \) given \( s_t = s \) at time \( t \). Reinforcement learning methods specify how the agent changes its policy as a result of its experiences. Specifically, the agent’s goal is to maximize the total amount of reward it receives over the long run.

Almost all reinforcement learning algorithms are based on estimating value functions. For one policy \( \pi \), there are two types of value functions. One is the state-value function \( V^\pi(s) \), which estimates how good it is, under policy \( \pi \), for the agent to be in state \( s \). It is defined as the expected return starting from \( s \) and thereafter following policy \( \pi \). The generalization of this function is shown in Eq. 1.

\[
V^\pi(s_t) \equiv E \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \right] 
\]

where \( \gamma \) is the discount factor and \( r_{t+k} \) is the sequence of rewards.

The other state-value function is the action-value function \( Q^\pi(s, a) \), which estimates how good it is, under policy \( \pi \), for the agent to perform action \( a \) in state \( s \). It is defined as the expected return starting from \( s \), taking action \( a \), and thereafter following policy \( \pi \). It is related to the state value function by Eq. 2.

\[
Q(s, a) \equiv r(s, a) + \gamma V^\ast(\delta(s, a)) 
\]

The process of computing \( V^\pi(s) \) or \( Q^\pi(s, a) \) is called policy evaluation. \( \pi \) can be improved to a better \( \pi' \) that, given a state, always selects the action, of all possible actions, with the best value based on \( V^\pi(s) \) or
This process is called policy improvement. $V^{\pi'}(s)$ or $Q^{\pi'}(s, a)$ can then be computed to improve $\pi'$ to an even better $\pi''$. The ultimate goal of RL is to find the optimal policy $\pi^*$ that has the optimal state-value function, denoted by $V^*(s)$ and defined as $V^*(s) = max_\pi V^\pi(s)$, or the optimal action-value function, denoted by $Q^*(s, a)$ and defined as $Q^*(s, a) = max_\pi Q^\pi(s, a)$. This recursive way of finding an optimal policy is called policy iteration. As a result $Q^*$ can be written in terms of $V^*$.

$$Q^*(s, a) = E[r(s, a) + \gamma V^*(\delta(s, a))]$$

$$= E[r(s, a)] + E[\gamma V^*(\delta(s, a))]$$

$$= E[r(s, a)] + \gamma \sum_{s'} P(s'|s, a) V^*(s')$$

(3)

where $P(s'|s, a)$ is the probability of taking action $a$ in state $s$ will produce the next state $s'$.

To make this function more manageable, $Q$ can be re-expressed recursively.

$$Q^*(s, a) = E[r(s, a)] + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a')$$

(4)

Eq. 4 can further be modified into a training rule that iteratively updates each $Q(s, a)$ as it is visited and converges to $Q^*(s, a)$. This training rule is defined in Equation 5.

$$Q_n(s, a) \leftarrow (1 - \alpha) Q_{n-1}(s, a) + \alpha \left[r + \gamma \max_{a'} Q_{n-1}(s', a')\right]$$

(5)

There exist three major methods for policy iteration: dynamic programming, Monte Carlo methods, and temporal-difference learning. Dynamic Programming refers to a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a Markov decision process (MDP). The key idea is the use of value functions to organize and structure the search for good policies. Classical Dynamic Programming algorithms\textsuperscript{11–13} are of limited utility in Reinforcement Learning, both because of their assumption of a perfect model and their great computational expense. However, they are very important theoretically.

Monte Carlo methods are employed to estimate functions using an iterative, incremental procedure. The term “Monte Carlo” is sometimes used more broadly for any estimation method whose operation involves a significant random component. For the present context it represents methods which solve the Reinforcement Learning problem based on averaging sample returns. To ensure that well-defined returns are available, they are defined only for episodic tasks, and it is only upon the completion of an episode that value estimates and policies are changed. By comparison with Dynamic Programming, Monte Carlo methods can be used to learn optimal behavior directly from interaction with the environment, with no model of the environment’s dynamics. They can be used with simulation, and it is easy and efficient to focus Monte Carlo methods on a small subset of the states. All Monte Carlo methods for Reinforcement Learning have been developed only recently, and their convergence properties are not well understood.
Temporal-Difference methods can be viewed as an attempt to achieve much the same effect as Dynamic Programming, but with less computation and without assuming a perfect model of the environment. Sutton’s method of Temporal-Differences is a form of the policy evaluation method in Dynamic Programming in which a control policy $\pi_0$ is to be chosen.\textsuperscript{14} The prediction problem becomes that of learning the expected discounted rewards, $V^{\pi}(i)$, for each state $i$ in $S$ using $\pi_0$. With the learned expected discounted rewards, a new policy $\pi_1$ can be determined that improves upon $\pi_0$. The algorithm may eventually converge to some policy under this iterative improvement procedure, as in Howard’s algorithm.\textsuperscript{15}

Q-Learning is a form of the successive approximations technique of Dynamic Programming, first proposed and developed by Watkins.\textsuperscript{16} Q-learning learns the optimal value functions directly, as opposed to fixing a policy and determining the corresponding value functions, like Temporal-Differences. It automatically focuses attention to where it is needed, thereby avoiding the need to sweep over the state-action space. Additionally, it is the first provably convergent direct adaptive optimal control algorithm.

Reinforcement Learning has been applied to a wide variety of physical control tasks, both real and simulated. For example, an acrobat system is a two-link, under-actuated robot roughly analogous to a gymnast swinging on a high bar. Controlling such a system by RL has been studied by many researchers.\textsuperscript{17–19} In many applications of RL to control tasks, the state space is too large to enumerate the value function. Some function approximators must be used to compactly represent the value function. Commonly used approaches include neural networks, clustering, nearest-neighbor methods, tile coding, and cerebellar model articulator controller.

\section*{B. Implementation of Reinforcement Learning Agent}

For the present research, the agent in the morphing airfoil problem is its RL agent. It attempts to independently maneuver from some initial state to a final goal state characterized by the aerodynamic properties of the airfoil. To reach this goal, it endeavors to learn, from its interaction with the environment, the optimal policy that, given the specific aerodynamic requirements, commands the series of actions that changes the morphing airfoil’s thickness or camber toward an optimal one. The environment is the resulting aerodynamics the airfoil is subjected to. It is assumed that the RL agent has no prior knowledge of the relationship between actions and the thickness and camber of the morphing airfoil. However, the RL agent does know all possible actions that can be applied. It has accurate, real-time information of the morphing airfoil shape, the present aerodynamics, and the current reward provided by the environment.

The RL agent uses a 1-step Q-learning method, which is a common off-policy Temporal Difference (TD) control algorithm. In its simplest form it is a modified version of Eq. 5 and is defined by

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right)$$

(6)

The Q-learning algorithm is illustrated as follows.\textsuperscript{20}
Q-Learning()

- 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 $r, s'$
    * $Q(s, a) \leftarrow Q(s, a) + \alpha \left( r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$
    * $s \leftarrow s'$
  - until $s$ is terminal
- return $Q(s, a)$

The agent learns the greedy policy, defined as:

$$
\begin{align*}
\epsilon - \text{greedy policy} \\
\text{if(probability > } 1 - \epsilon) \\
a = \arg \max_{a} Q(s, a) \\
\text{else} \\
a = \text{rand}(a_i)
\end{align*}
$$

(7)

As the learning episodes increase, the learned action-value function $Q(s, a)$ converges asymptotically to the optimal action-value function $Q^*(s, a)$. The method is an off-policy one as it evaluates the target policy (the greedy policy) while following another policy. The policy used in updating $Q(s, a)$ can be a random policy, with each action having the same probability of being selected. The other option is an $\epsilon$-greedy policy, where $\epsilon$ is a small value. The action $a$ with the maximum $Q(s, a)$ is selected with probability $1-\epsilon$, otherwise a random action is selected.

If the number of the states and the actions of a RL problem is a small value, its $Q(s, a)$ can be represented using a table, where the action-value for each state-action pair is stored in one entity of the table. Since the RL problem for the morphing vehicle has states (the shape of the airfoil) on continuous domains, it is impossible to enumerate the action-value for each state-action pair. In essence, there are an infinite number of state-action pairs. One commonly used solution is to artificially quantize the states into discrete sets thereby reducing the number of state-action pairs the agent must visit and learn. The goal in doing this is to reduce the number of state-action pairs while maintaining the integrity of the learned action-value.
function. For a given problem experimentation must be conducted to determine what kind of quantization is appropriate for the states. In this paper several increasingly larger quantizations are considered to determine what the largest allowable step size is. For the problem at hand, this becomes most important for keeping the number of state-action pairs manageable when more state variables are added to the thickness and camber in the form of other morphing parameters.

III. Modeling of the Morphing Air Vehicle

A. Aerodynamic Modeling

To calculate the aerodynamic properties of many different wings in a short period of time, or as a simple wing changes its shape into a more complex shape, a numerical model of the wing is developed. A constant strength doublet-source panel method is used to model the aerodynamics of the wing. The main assumption is that the flow is incompressible; otherwise a much more complex model is necessary. This assumption is valid because current interests lie in the realm of micro air vehicles, which fly at speeds less than Mach 0.3. One other assumption is that the flow is inviscid. Thus the model is only valid for the linear range of angle-of-attack. Despite this versatility there are some limitations to the model. Since the model uses a panel method to calculate the aerodynamics, it is very sensitive to the grid, or location of the panels, and the number of panels created. The model calculates the lift force, drag force, and pitching moment for the given shape. Reference 21 gives a more detailed description of the aerodynamic model used.

B. Dynamic Modeling

The dynamic behavior of the morphing air vehicle is modeled by nonlinear six degree-of-freedom equations of motion. Two co-ordinate systems: an inertial axis system, and a body fixed axis with origin at the center of mass of the vehicle are used to write these equations of motion.

Equations of motion for the morphing vehicle can be written as for a nonlinear dynamic system that is affine in control and can be split into a structured form. It consists of an exactly known kinematic differential equation and a momentum level equation with uncertain parameters. The kinematic level states $d_x, d_y, d_z$, are the positions of the center of mass of the morphing vehicle along the inertial $X_N, Y_N$ and $Z_N$ axes. Rotational states at kinematic level $\phi, \theta$ and $\psi$ are the 3-2-1 Euler angles which give the relative orientation of the body axis with the inertial axis. The acceleration level states are the body-axis linear velocities $u, v, w$ and the body-axis angular velocities $p, q, r$.

The following equations represent the relation between kinematic states and acceleration level states.

\[
\dot{p}_c = J_i \nu_c \\
\dot{\sigma} = J_a \omega
\]
where \( \mathbf{p}_c = \begin{bmatrix} d_x & d_y & d_z \end{bmatrix}^T \), \( \mathbf{v}_c = \begin{bmatrix} u & v & w \end{bmatrix}^T \), \( \mathbf{\sigma} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T \) and \( \mathbf{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^T \)

Matrices \( J_l \) and \( J_a \) are given as

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

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

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

\[
m \ddot{\mathbf{v}}_c + \tilde{\mathbf{\omega}} m \mathbf{v}_c = \mathbf{F} + \mathbf{F}_{aero} \quad (11)
\]

\[
I \ddot{\mathbf{\omega}} + \dot{I} \mathbf{\omega} + \tilde{\mathbf{\omega}} I \mathbf{\omega} = \mathbf{M} + \mathbf{M}_{aero} \quad (12)
\]

where \( \tilde{\mathbf{\omega}} \mathbf{V} \) is the matrix representation of the cross-product between vector \( \mathbf{\omega} \) and vector \( \mathbf{V} \).

\[
\tilde{\mathbf{\omega}} = \begin{bmatrix}
0 & -r & q \\
r & 0 & -p \\
-q & p & 0
\end{bmatrix}
\]

\( m \) is the mass of the morphing air vehicle, \( \mathbf{F} \) is the force generated by control, \( \mathbf{F}_{aero} \) represents aerodynamic forces, \( I \) is the body axis moment of inertia, \( \mathbf{M} \) is the control torque and \( \mathbf{M}_{aero} \) represents aerodynamic moments.

There is an additional term of \( \dot{I} \mathbf{\omega} \) in Eqn. 12, when compared to rigid body equations of motion. As the morphing vehicle changes shape there is a consequent change in moment of inertia. Hence it is important to retain this term in equations of motion in case of morphing air vehicle. This change is responsible for speeding up or slowing down the rotation of the vehicle due to the time rate of change in the moment of inertia about a particular axis. For this paper only longitudinal dynamics is considered for the simulations.

C. **Reference Trajectory**

The objective of this work is to show that while changing shape the wing can track reference trajectories which normally includes different types of flight conditions and requirements. Some of the common maneuvers are climbing up, diving down, turning etc. Motivated by this idea a reference trajectory is divided in three segments. In the first segment wing is at cruise condition and is flying under straight and level flight. After this it tries to dive down to a desired altitude in the second segment and then it climbs up to the
desired altitude in the third segment. Pitch angle $\theta$ and altitude are given as discrete commands and hence polynomials are used to make the reference trajectory smooth.

IV. Control Laws Using Structured Adaptive Model Inversion

A. Structured Adaptive Model Inversion (SAMI)

The Structure Adaptive Model Inversion (SAMI) control algorithm, developed at Texas A&M University, has been applied to various spacecraft problems. SAMI has been shown to be effective for tracking spacecraft and aggressive aircraft maneuvers. The SAMI approach has been extended to handle actuator failures and to facilitate correct adaptation in presence of actuator saturation. It is based on the concepts of Feedback Linearization, Dynamic Inversion and Structured Model Reference Adaptive control. The control objective is calculate commanded moments to track the reference trajectory even when the dynamic properties of the vehicle change, due to the morphing. In SAMI, dynamic inversion is used to solve for the control. The dynamic inversion is approximate, as the system parameters are not modeled accurately. An adaptive control structure is wrapped around the dynamic inverter to account for the uncertainties in the system parameters. 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 dynamics of most systems can be represented in the form of a second order differential equation. The second order differential equation can be separated into a kinematic part and a dynamic part. The kinematic equations are accurately known. The uncertainties exist only in the momentum level dynamic equations. This restricts the adaptation to a subset of the state-space, enabling efficient adaptation.

B. SAMI Controller for Attitude Control

Eqn.9 and Eqn.12 represent the attitude equations of motion for the morphing vehicle. Without the aero-dynamic forces and the $\dot{I}\omega$ term, Eqn.9 and Eqn.12 can be manipulated to obtain the following form

$$I_a^*(\sigma)\ddot{\sigma} + C_a^*(\sigma, \dot{\sigma})\dot{\sigma} = P_a^T(\sigma)M \quad (14)$$

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

$$P_a(\sigma) \triangleq J_a^{-1}(\sigma) \quad (15)$$
$$I_a(\sigma) \triangleq P_a^TIP_a \quad (16)$$
$$C_a^*(\sigma, \dot{\sigma}) \triangleq -I_a^*J_aP_a + P_a^T[\tilde{P}_a\dot{\sigma}]IP_a \quad (17)$$

One can linearly parametrized the the left hand side of Eqn.14

$$I_a^*(\sigma)\ddot{\sigma} + C_a^*(\sigma, \dot{\sigma})\dot{\sigma} = Y_a(\sigma, \dot{\sigma}, \ddot{\sigma})\theta \quad (18)$$
where $\theta$ is the constant inertia parameter vector defined as $\theta \triangleq \begin{bmatrix} I_{11} & I_{22} & I_{33} & I_{12} & I_{13} & I_{23} \end{bmatrix}^T$ and $Y_a(\sigma, \dot{\sigma}, \ddot{\sigma})$ is a regression matrix.

The product of the inertia matrix and a vector can be written as

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

(19)

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

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

(20)

The terms on the left hand side of Eqn.14 can be written as

$$I_a^*\ddot{\sigma} = P_a^T IP_a \ddot{\sigma} = P_a^T \Lambda(P_a \ddot{\sigma}) \theta$$

(21)

$$C_a^*\dot{\sigma} = -P_a^T IP_a \dot{\sigma} + P_a^T [P_a \dot{\sigma} \theta]$$

$$= P_a^T \{-\Lambda(P_a \dot{\sigma}) + [P_a \dot{\sigma} \theta] \} \theta$$

(22)

Combining Eqns.21 and 22 we have the linear minimal parameterization for the Inertia Matrix.

$$I_a^*\ddot{\sigma} + C_a^*(\sigma, \dot{\sigma})\dot{\sigma} = P_a^T \{\Lambda(P_a \ddot{\sigma}) - \Lambda(P_a \dot{\sigma}) + [P_a \dot{\sigma} \theta] \} \theta$$

(23)

The attitude tracking problem can be formulated as follows. The objective is to make the error between reference and plant to go to zero. The morphing air vehicle will track an attitude trajectory in terms of the 3-2-1 Euler angles. It is assumed that the desired reference trajectory is twice differentiable with respect to time. Let $\epsilon \triangleq \sigma - \sigma_r$ be the tracking error. Differentiating twice and multiplying by $I_a^*$ throughout

$$I_a^* \ddot{\epsilon} = I_a^* \ddot{\sigma} - I_a^* \ddot{\sigma}_r$$

(24)

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

$$I_a^* \ddot{\epsilon} + (C_{da} + C_a^*(\sigma, \dot{\sigma}))\dot{\epsilon} + K_{da} \epsilon$$

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

(25)

The RHS of Eqn.25 can be written as

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

(26)
From Eqn.14 and the construction of Y similar to Eqn.23, the RHS of Eqn.25 can be further written as

\[ P_T^a M - Y_a(\sigma, \dot{\sigma}, \dot{\sigma}_r, \ddot{\sigma}_r)\theta + C_{da} \dot{\epsilon} + K_{da} \epsilon \]  \hspace{1cm} (27)

From above equation it is clear that the control law is

\[ M = P_a^{-T} \{ Y_a(\sigma, \dot{\sigma}, \dot{\sigma}_r, \ddot{\sigma}_r)\theta - C_{da} \dot{\epsilon} - K_{da} \epsilon \} \]  \hspace{1cm} (28)

Now \( \theta \) may not be known accurately in actual practice but this control law requires that the inertia parameters \( \theta \) be known accurately. So by using the certainty equivalence principle, adaptive estimates for the inertia parameters \( \hat{\theta} \) will be used for calculating the control.

\[ M = P_a^{-T} \{ Y_a(\sigma, \dot{\sigma}, \dot{\sigma}_r, \ddot{\sigma}_r)\hat{\theta} - C_{da} \dot{\epsilon} - K_{da} \epsilon \} \]  \hspace{1cm} (29)

Substituting Eqn.29 in equation 14 the closed loop dynamics becomes

\[ I_a^* \ddot{\epsilon} + (C_{da} + C_{a^*}(\sigma, \dot{\sigma}))\dot{\epsilon} + K_{da} \epsilon = Y_a(\sigma, \dot{\sigma}, \dot{\sigma}_r)\tilde{\theta} \]  \hspace{1cm} (30)

where \( \tilde{\theta} = \hat{\theta} - \theta \).

C. Update laws

To find the update law for \( \theta \) and ensure the stability of the system following Lyapunov function is selected.

\[ V = \frac{1}{2} \dot{\epsilon}^T I_a^* \dot{\epsilon} + \frac{1}{2} \dot{\epsilon}^T K_{da} \dot{\epsilon} + \frac{1}{2} \ddot{\theta}^T \Gamma^{-1} \ddot{\theta} \]

where \( \Gamma^{-1} \) is a symmetric positive definite matrix. Taking the time derivative of above Lyapunov function

\[ \dot{V} = \frac{1}{2} \dot{\epsilon}^T I_a^* \dot{\epsilon} + \dot{\epsilon}^T K_{da} \dot{\epsilon} + \ddot{\theta}^T \Gamma^{-1} \ddot{\theta} \]

\[ \dot{V} = \frac{1}{2} \dot{\epsilon}^T I_a^* \dot{\epsilon} + \dot{\epsilon}^T K_{da} \dot{\epsilon} + \ddot{\theta}^T \Gamma^{-1} \ddot{\theta} \]

Substituting the expression for \( \ddot{\epsilon} \)

\[ \dot{V} = \dot{\epsilon}^T (\frac{1}{2} I_a^* - C_{a^*}) \dot{\epsilon} - \dot{\epsilon}^T C_{da} \dot{\epsilon} + (\dot{\epsilon}^T Y(\sigma, \dot{\sigma}, \dot{\sigma}_r, \ddot{\sigma}_r) + \dot{\theta}^T \Gamma^{-1} \dot{\theta} \]

The first term is zero from Reference. To make the last term zero adaptive laws for \( \theta \) should be selected as

\[ \dot{\theta} = -\Gamma Y(\sigma, \dot{\sigma}, \dot{\sigma}_r, \ddot{\sigma}_r) \dot{\epsilon} \]

which implies

\[ \dot{\theta} = -\Gamma Y(\sigma, \dot{\sigma}, \dot{\sigma}_r, \ddot{\sigma}_r) \dot{\epsilon} \]

Using this adaptive law for \( \theta \) reduces Lyapunov function \( V \) to

\[ \dot{V} = -\dot{\epsilon}^T C_{da} \dot{\epsilon} \leq 0 \]
D. Stability Analysis

From the above equation it is concluded that $V > 0$ and $\dot{V} \leq 0$. Lyapunov candidate is function of $\dot{\theta}, \epsilon, \dot{\epsilon}$ and $\dot{V}$ is a function of $\dot{\epsilon}$ and hence $\dot{\epsilon} \in L_2 \cap L_\infty$. Also $\dot{\theta}, \epsilon, \dot{\epsilon}$ are all bounded. Since $\sigma_r, \dot{\sigma}_r, \epsilon$ and $\dot{\epsilon}$ are bounded, $\sigma, \dot{\sigma}$ and regression matrix are also bounded and $\in L_\infty$. Taking the higher derivatives of Lyapunov function and using Barbalat’s Lemma it can be proved that the tracking error dynamics is asymptotically stable and hence $\epsilon \to 0$ as $t \to \infty$

E. Controllers on Morphing wing

Presently, the morphing wing is considered as a free body that can fly. It does not have any of the conventional controllers, e.g. elevators or flaps, to track the reference trajectory. It is assumed that the thrust is applied through a jet engine and only affects translational velocities $u$ and $w$. For tracking rotational states, pseudo controls are used, i.e. a control effectiveness matrix is selected to represent the effectiveness of these pseudo controls. Work is continuing on the design of non-conventional controls, e.g. camber or location of camber, to use in place of pseudo controls.

V. A-RLC Architecture Functionality

The Adaptive-Reinforcement Learning Control Architecture is composed of two sub-systems: Reinforcement Learning and Structured Adaptive Model Inversion (SAMI) (Fig. 1). The two sub-systems interact significantly 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. For either type of stage, the system functions as follows.

Considering the Reinforcement Learning sub-system at the top of Fig. 1 and moving counterclockwise, the Reinforcement Learning module initially commands an arbitrary action from the set of admissible actions. This action is sent to the plant, which produces a shape change. The cost associated with the resultant shape change in terms of system states, parameters, and user defined performance measures, is evaluated with the cost function and then passed to the agent. The agent modifies its action-value function according to Eqn. 6. For the next episode, the agent chooses a new action based on the current policy and its updated action-value function, and the sequence repeats itself.

Considering the SAMI sub-system at the bottom of Fig. 1, shape changes in the plant due to actions generated by the agent cause the plant dynamics to change. The SAMI controller maintains trajectory tracking irrespective of the changing dynamics of the plant due to these shape changes.
Figure 1. Adaptive-Reinforcement Learning Control Architecture.
VI. Numerical Examples

A. Purpose and Scope

The purpose of the numerical examples is to demonstrate the learning performance and the trajectory tracking performance of the A-RLC architecture. The morphing wing in both examples is tasked with following a set reference trajectory and changing shape to meet preset aerodynamic demands.

B. Example 1

This example demonstrates the A_RLC architecture using two sets of learned shape data. Prior to running this simulation example, the agent, in this case the morphing wing, is set up to learn how to change from any shape in the state space to a shape that meets an aerodynamic goal. The state space consists of the shape parameters tip chord $c_t$, root chord $c_r$, span $b$, and leading edge sweep angle $\Lambda$. The two goals the agent learns is a maximum lift of $0.3 \pm 0.05$ and a minimum lift of $0.09 \pm 0.05$. These goals can be thought of as the nominal goals. During the simulation, the morphing wing must change shape to meet the nominal aerodynamic goals listed in Table 1 while following the trajectory.

<table>
<thead>
<tr>
<th>Time</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t &lt; 40\text{sec}$</td>
<td>$0.3 \pm 0.05$</td>
</tr>
<tr>
<td>$40\text{sec} \leq t &lt; 80\text{sec}$</td>
<td>$0.09 \pm 0.05$</td>
</tr>
<tr>
<td>$t \geq 80\text{sec}$</td>
<td>$0.3 \pm 0.05$</td>
</tr>
</tbody>
</table>

The simulation time histories are shown in Figures 2, 3, 4, 5, 6, and 7. Figure 2 shows the translational time histories. The reference trajectory indicates a dive and climb corresponding to the times that the commanded aerodynamic goals for the shape are changed. As only longitudinal dynamics is considered only $u$, $w$ and $\theta$ are considered. Despite the changes in shape, the adaptive controller maintains good tracking for all translational states. Figure 3 also shows good tracking of the rotational states with only small tracking error during the duration of shape change. This error is due to the effect of inertia changes, but the controller is able to handle these changes and regulate the tracking error to zero. It is also noted that the rate of shape change is high, and hence during the shape change, the dynamics of the system change and the error increases during the transient response. The simulation shows that error decreases with the decrease in rate of shape change. Figure 4 illustrates the shape changes the morphing wing makes. For the initial maximum nominal lift command, the shape parameters settle to a tip chord of 0.8 ft, root chord of 2.4 ft, span of 11 ft, and leading edge sweep angle of 0 degrees. The wing chooses this shape by consulting the learning data stored in the $Q(s,a)$ for $c_L = 0.3 \pm 0.05$. After 40 seconds, the wing consults the learning data stored in the $Q(s,a)$ for $c_L = 0.09 \pm 0.05$ and chooses the shape of tip chord of 0.8 ft, root chord of 3.2 ft, span of 5
ft, and leading edge sweep angle of 0 degrees. Finally, after 80 seconds the wing again chooses a shape for maximum lift of tip chord of 0.5 ft, root chord of 3 ft, span of 12 ft, and leading edge sweep angle of 0.05 degrees. This shape is different from that of the first 40 seconds because the starting state is different. The learning data tells the wing how to get from any state in the state-space to the closest shape in the space corresponding to the goal range. Since the wing configuration was different at 0 seconds and 80 seconds, the final shape chosen to meet the goal is not the same. The effects of these shape changes can be seen in Figures 5, 6, and 7. Most notable is the axial force in Figure 5 that shows a dip when the wing goes to a minimum lift configuration and then a raise as the wing returns to a maximum lift configuration.

Figure 2. Example 1: Translational Time Histories

Figure 5 shows the forces and moments acting on the wing. Axial force and normal force change during in congruence with the shape change. Pitching moment is also shown in Figure 5. These forces are calculated using a table look up of aerodynamic coefficients at different flight conditions.

Figure 6 shows the mass of the wing. An uncertainty of 20% is introduced in the mass and moment of inertia of the wing for the simulation. Using the adaptive laws, estimated parameters are calculated. These parameters settle down to constants that may not be their true values. Moment of inertia for the wing is defined as a function of \( c_t \), \( c_r \), and \( \Lambda \) for the simulation. It varies with all the wing geometrical parameters. It is also observed that though the moment of inertia is updated, the changes are extremely small. Tracking
Figure 3. Example 1: Rotational Time Histories
Figure 4. Example 1: Morphing/Shape Time Histories
Figure 5. Example 1: Force Time Histories
control of the wing is calculated using this estimated mass and moment of inertia to track the reference trajectory.

![Mass Time History](image)

**Figure 6. Example 1: Mass Time History**

![Moment of Inertia Time History](image)

**Figure 7. Example 1: Moment of Inertia Time History**

C. Example 1

This example further exercises the A-RLC structure by using the two sets of learning data to achieve several different goals without the need for additional learning. Since maximum lift and minimum lift are already learned, it is possible to use these data achieve intermediate goals and still maintain good trajectory tracking. The series of aerodynamic goals for this example is listed in Table 2. The agent achieves these intermediate goals by judiciously choosing which set of learning data to consult. For example, initially the wing must find a maximum lift configuration so it just uses the maximum lift learning data. At the next time increment, the agent must find a configuration to achieve a lift coefficient of 0.2. Since the wing is starting from a
maximum lift configuration, it can use the minimum lift learning data and just stop when the lift goal is achieved. A similar selection can be done when the wing needs to change from a minimum lift configuration to a 0.2 configuration.

<table>
<thead>
<tr>
<th>Time</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t &lt; 20\text{sec}$</td>
<td>$0.3 \pm 0.05$</td>
</tr>
<tr>
<td>$20\text{sec} \leq t &lt; 40\text{sec}$</td>
<td>$0.2 \pm 0.05$</td>
</tr>
<tr>
<td>$40\text{sec} \leq t &lt; 60\text{sec}$</td>
<td>$0.09 \pm 0.05$</td>
</tr>
<tr>
<td>$60\text{sec} \leq t &lt; 80\text{sec}$</td>
<td>$0.2 \pm 0.05$</td>
</tr>
<tr>
<td>$t \geq 80\text{sec}$</td>
<td>$0.3 \pm 0.05$</td>
</tr>
</tbody>
</table>

The simulation time histories are shown in Figures 8, 9, 10, 11, 12, and 13. Figure 8 shows the translational time histories. The reference trajectory indicates a dive and climb in the same manner as Example 1. Despite the changes in shape the adaptive controller maintains good tracking for all translational states except for small steady state errors during the shape change while tracking the altitude. Figure 3 also shows good tracking of the rotational states. A pitch angle is commanded to 30 degrees and $-30$ degrees during the climb and dive, respectively. During the same time increment, the wing changes shape. Figure 10 illustrates the shape changes the morphing wing makes. Table 3 lists the shape parameters for the shape chosen by the wing for each time segment. There is the most activity in root chord and span. The differences in shape for the 0.2 configuration can again be explained by the difference in starting configuration. The effects of these shape changes can be seen in Figures 11 and 12. Again, most notable is the axial force in Figure 11 that reflects the change in lift generated as the wing changes shape. It should be noted that the reference trajectory selected for tracking is arbitrary. It just demonstrates three flight conditions of cruise, climb and dive. This trajectory is not optimally calculated for these flight conditions. Extension of this work will include more realistic reference trajectories, which will give more insight into the maximum limits of rate of change of shape of the morphing wing.

VII. Conclusions and Future Research

This paper describes the development and application of Adaptive-Reinforcement Learning Control to the problem of the morphing wing. The tracking performance of the adaptive controller when the wing is subjected to shape changes is investigated. In addition, the reinforcement learning is exercised to show that learning data for preset goals can be used to achieve other goals without the need to learn from scratch.

The results show that the A-RLC architecture is able to command the shape of the wing whilst main-
Table 3. Aerodynamic Goals

<table>
<thead>
<tr>
<th>Time</th>
<th>$c_t$ (ft)</th>
<th>$c_r$ (ft)</th>
<th>$b$ (ft)</th>
<th>$\Lambda$ (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t &lt; 20\text{sec}$</td>
<td>0.8</td>
<td>2.4</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>$20\text{sec} \leq t &lt; 40\text{sec}$</td>
<td>0.8</td>
<td>2.6</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>$40\text{sec} \leq t &lt; 60\text{sec}$</td>
<td>0.8</td>
<td>3.2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>$60\text{sec} \leq t &lt; 80\text{sec}$</td>
<td>0.8</td>
<td>3.2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>$t \geq 80\text{sec}$</td>
<td>0.5</td>
<td>3.0</td>
<td>12</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Figure 8. Example 2: Translational Time Histories
Figure 9. Example 2: Rotational Time Histories
Figure 10. Example 2: Morphing/Shape Time Histories
Figure 11. Example 2: Force Time Histories

Figure 12. Example 2: Mass Time History
The first example shows that the adaptive controller can transition between two sets of learned shape with only minimal tracking error. The second example shows that intermediate goals can be achieved using the two extrema sets of learning data all while maintaining good tracking of the trajectory despite changes in forces, mass, and inertia. These two examples show that the A-RLC architecture is a good candidate for controlling both the shape changing of a morphing aircraft and the vehicle itself.

VIII. Acknowledgment

This work was sponsored (in part) by the Air Force Office of Scientific Research, USAF, under grant/contract number FA9550-08-1-0038. The technical monitor is Dr. William M. McEneaney. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the National Aeronautics and Space Administration or the Air Force Office of Scientific Research, or the U.S. Government.

References


