AIAA <i>Infotech@Aerospace</i> 2007 Conference and Exhibit
7 - 10 May 2007, Rohnert Park, California

AIAA 2007-2805

Reinforcement Learning of Morphing Airfoils with Aerodynamic and Structural Effects

Amanda Lampton[*], Adam Niksch[†], and John Valasek[‡]

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

This paper applies a Reinforcement Learning methodology to the problem of airfoil morphing. The reinforcement learning as it is applied to morphing is integrated with a computational model of an airfoil. The computational model utilizes a doublet panel method whose end yield is airfoil lift, drag, and moment coefficients. An episodic unsupervised learning simulation using the Q-Learning method is developed to learn the optimal shape and shape change policy. Optimality is addressed by reward functions based on airfoil properties such as lift coefficient, drag coefficient, and moment coefficient about the leading edge representing optimal shapes for specified flight conditions. The methodology is demonstrated with numerical examples of a NACA type airfoil that autonomously morphs in two degrees of freedom, thickness and camber, to a shape that corresponds to specified goal requirements. Given the nature of the problem and the possibility of there being many shapes that satisfy the lift, drag, or moment coefficient requirements, the results presented in this paper show that this methodology is capable of learning the range of acceptable shapes for a given set of requirementes and morphing into one.

Nomenclature

A	Plane of the Airfoil
a	Action
$A(s_t)$	Set of actions available in state s_t
C	Force Coefficient

[*]Graduate Research Assistant, Flight Simulation Laboratory, Aerospace Engineering Department, Student Member AIAA, alampton@tamu.edu.

[†]Undergraduate Research Assistant, Flight Simulation Laboratory, Aerospace Engineering Department, Student Member AIAA, adam45611@tamu.edu.

[‡]Associate Professor and Director, Flight Simulation Laboratory, Aerospace Engineering Department, Associate Fellow AIAA, valasek@tamu.edu.

c	Chord Length
C_p	Pressure Coefficient
I	Moment of Inertia
M	Bending Moment
$Q^{\pi}(s,a)$	action-Value Function for Policy π
R	Reward
S	Set of Possible States for Reinforcement Learning
s	State
t	Time Step
u	Tangential Velocity
w	Normal Velocity
V	Velocity
$V^{\pi}(s)$	State Value Function for Policy π
Z	Plane of the Circle
μ	Doublet Strength
α	Angle of Attack
θ_i	Angle between individual panels
σ_{xx}	Bending Stress

Subscript

a	Airfoil Axial Force
d	Airfoil Drag Force
l	Airfoil Lift Force
$lower$	Lower Surface
n	Normal Force
p	Panel Coordinates
t	Time
$upper$	Upper Surface
0	Initial
∞	Freestream

I. Introduction

Current interest in morphing vehicles has been fuelled by advances in smart technologies, including materials, sensors, actuators, and their associated support hardware and microelectronics. 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.[1] Although there are several definitions and interpretations of the term morphing, it is generally agreed that the concept refers to large scale shape changes or transfigurations. Various organizations that are researching morphing technologies for both air and space vehicles have adopted their own definitions according to their needs. The National Aeronautics and Space Administration's (NASA) Morphing Project defines morphing as an efficient, multi-point adaptability that includes macro, micro, structural and/or fluidic approaches.[2] The Defense Advanced Research Projects Agency (DARPA) uses the definition of a platform that is able to change its state substantially (on the order of 50%) to adapt to changing mission environments, thereby providing a superior system capability that is not possible without reconfiguration. Such a design integrates innovative combinations of advanced materials, actuators, flow controllers, and mechanisms to achieve the state change.[3]

In spite of their relevance, these definitions do not adequately address or describe the supervisory and control aspects of morphing. Reference 4 attempts to do so by making a distinction between Morphing for Mission Adaptation, and Morphing for Control. In the context of flight vehicles, Morphing for Mission Adaptation is a large scale, relatively slow, in-flight shape change to enable a single vehicle to perform multiple diverse mission profiles. Conversely, Morphing for Control is an in-flight physical or virtual shape change to achieve multiple control objectives, such as maneuvering, flutter suppression, load alleviation and active separation control. In this paper, the authors consider the problem of Morphing for Mission Adaptation.

In the context of intelligent systems, three essential functionalities of a practical Morphing for Mission Adaptation capability are:

1. When to reconfigure

2. How to reconfigure

3. Learning to reconfigure

When to reconfigure is driven by mission priorities/tasks, and leads to optimal shape being a system parameter. In the context of a reconfigurable vehicle such as an aircraft, each shape results in performance values (speed, range, endurance, etc.) at specific flight conditions (Mach number, altitude, angle-of-attack, and sideslip angle). It is a major issue, as the inability for a given aircraft to perform multiple missions successfully can directly be attributed to shape, at least if aerodynamic performance is the primary consideration. This is because for a given task or mission, there is usually an ideal or optimal vehicle shape, e.g. configuration.[4] However, this optimality criteria may not be known over the entire flight envelope in actual practice, and the mission may be modified or completely changed during operation. *How* to reconfigure is a problem of sensing, actuation, and control.[5] They are important and challenging since large shape changes produce time-varying vehicle properties, and especially, time-varying moments and products of inertia. The controller must therefore be sufficiently robust to handle these potentially wide variations. *Learning* to reconfigure is perhaps the most challenging of the three functionalities, and the one that has received the least attention. Even if optimal shapes are known, the actuation scheme(s) to produce them may be only poorly understood, or not understood at all; Reinforcement Learning is therefore a candidate approach. It is important that learning how to reconfigure is also life-long learning. This will enable the vehicle to be more survivable, operate more safely, and be multi-role.

Morphing for Mission Adaptation has been investigated many times in the past. Reference 6 describes a methodology that combines Structured Adaptive Model Inversion (SAMI) with Reinforcement Learning to address the optimal shape change of an entire vehicle. In this case a smart block. The method learns the commands for two independent morphing parameters that produce the optimal shape. The authors show that the methodology is capable of learning the required shape and changing into it and accurately tracking some reference trajectory.[6] This methodology is further developed in Reference 7. It is extended to an "air vehicle" using Q-learning to learn the optimal shape change policy. The authors show that the methodology is able to handle 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.[7] Finally, 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.[8] The authors showed that the approximation scheme resulted in marked improvements in the learning as opposed to the previously employed K Nearest Neighbor approach.

The problem of a morphing airfoil was investigated by Hubbard in Reference 9 The focus is on the physical shape change of an airfoil modeled by a space/time transform parameterization. The space/time parameterization results in a spatially decoupled system with Fourier coefficients as inputs and orthogonal basis shapes as outputs.[9]

This paper proposes and develops a conceptual architecture that addresses these essential functionalities of Morphing for Mission Adaptation. It combines a computational model of an airfoil that calculates the aerodynamic and structural properties as the airfoil changes shape. Reinforcement learning is wrapped around the model to learn the commands that produce the optimal shape based on lift, drag, and moment, all of which are dependent on flight condition. The goal is to eventually apply this method to an entire vehicle. Reinforcement learning learns the optimality relations between the flight condition, aerodynamic requirements, and the shape. The airfoil can then be subjected to a series of aerodynamics requirements and use the relations learned to choose a good shape for the current set of requirements. This paper first describes the reinforcement learning module. It uses Q-learning to learn how to morph into specified shapes. Then the airfoil model is developed. Flexibility is allowed in the sense that thickness, camber, chord, and angle-of-attack all have the potential of being commanded and used to determine the optimal configuration. The model is validated and verified against published airfoil data for representative NACA airfoils. The methodology is then demonstrated by numerical examples of the airfoil autonomously morphing into optimal shapes corresponding to a specified aerodynamic requirements. The numerical examples entail the agent learning how to meet requirements defined by some combination of the airfoil lift coefficient, drag coefficient, and moment coefficient.

II. Reinforcement Learning Module

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, $t = 0, 1, 2, 3....$ 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, denoted by π is called the agent's policy, where $\pi_t(s, a)$ 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. 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 π, there are two types of value functions. One is the state-value function $V^\pi(s)$, which estimates how good it is, under policy π, for the agent to be in state s. It is defined as the expected return starting from s and thereafter following policy π. The other state-value function is the action-value function $Q^\pi(s, a)$, which estimates how good it is, under policy π, 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 π and shown in Equation 1.

$$Q^\pi\left(s, a\right) = E_\pi\left\{R_t | s_t = s, a_t = a\right\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \tag{1}$$

The process of computing $V^\pi(s)$ or $Q^\pi(s, a)$ is called policy evaluation. π can be improved to a better π' that, given a state, always selects the action, of all possible actions, with the best value based on $V^\pi(s)$ or $Q^\pi(s, a)$. This process is called policy improvement. $V^{\pi'}(s)$ or $Q^{\pi'}(s, a)$ can then be computed to improve π' to an even better π''. The ultimate goal of RL is to find the optimal policy π^* 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^*.

$$Q^*\left(s,a\right) = E_\pi\left\{r_{t+1} + \gamma V^*\left(s_{t+1}\right)|s_t = s, a_t = a\right\} \tag{2}$$

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[10–12] 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 π_0 is to be chosen.[13] The prediction problem becomes that of learning the expected discounted rewards, $V^\pi(i)$, for each state i in S using π_0. With the learned expected discounted rewards, a new policy π_1 can be determined that improves upon π_0. The algorithm may eventually converge to some policy under this iterative improvement procedure, as in Howard's algorithm.[14] Q-Learning is a form of the successive approximations technique of Dynamic Programming, first proposed and developed by Watkins.[15] 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.[16,17,18] 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.

A. Implementation of Reinforcement Learning Module

For the present research, the agent in the morphing airfoil problem is its RL module. 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 towards an optimal one. The environment is the flight conditions and resulting aerodynamics the airfoil is subjected to. It is assumed that the RL module has no prior knowledge of the relationship between actions and the thickness and camber of the morphing airfoil. However, the RL module 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 module uses a 1-step Q-learning method, which is a common off-policy Temporal Difference (TD) control algorithm. In its simplest form it is defined by

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma max_{a'} Q(s',a') - Q(s,a)\} \tag{3}$$

The Q-learning algorithm is illustrated as follows:[19]

```
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 $r$, $s'$
        ∗ $Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma max_{a'} Q(s',a') - Q(s,a)\}$
        ∗ $s \leftarrow s'$
    – until $s$ is terminal
```

- return $Q(s, a)$

The agent learns the greedy policy, defined as:

$$\begin{aligned}
&\varepsilon - greedypolicy \\
&\text{if(probability} > 1{-}\varepsilon) \\
&a = \arg\max_a Q(s, a) \\
&\text{else} \\
&a = rand(a_i)
\end{aligned} \tag{4}$$

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 ϵ-greedy policy, where ϵ is a small value. The action a with the maximum $Q(s, a)$ is selected with probability 1-ϵ, 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 ellipsoidal vehicle) on continuous domains, it is impossible to enumerate the action-value for each state-action pair. In this case, the K-nearest neighbors method to approximate the $Q(s, a)$ is implemented. The action-value of each state-action pair is computed using the Q-learning method illustrated above. The action-value of any state-action pair not already recorded in $Q(s, a)$ is calculated using the interpolation of those of its K nearest neighbors already stored in $Q(s, a)$.

When selecting the next action, one typical problem the agent has to face is the exploration-exploitation dilemma.[19] If the agent selects a greedy action that has the highest value, then it is exploiting its knowledge obtained so far about the values of the actions. If instead it selects one of the non-greedy actions, then it is exploring to improve its estimate of the non-greedy actions' values. Naturally, the RL agent is more explorative in the early stage of the learning and more exploitative in the late stage. The change from exploration to exploitation is realized by changing the value of ϵ in the ϵ-greedy policy. At first, ϵ is large, so that each action has equal probability of being selected and the agent can explore as many state-action pairs as possible. The value of ϵ is decreased gradually as the learning goes on, which allows the agent to exploit its learning result.

III. Airfoil Model Representation

To calculate the aerodynamic properties of many different airfoils in a short period of time, or as a single airfoil changes shape, a numerical model of the airfoil is developed. A constant strength doublet panel method is used to model the aerodynamics of the airfoil. 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. Other assumptions are that both the upper and lower surfaces of the airfoil are pinned at the leading and trailing edge rendering the structural moment at these points to be zero. These boundary conditions become important in later sections in the calculation of M_y and σ_{xx}. One final assumption is that the flow is inviscid. Thus the model is only valid for the linear range of angle-of-attack.

The flexibility of this type of model allows the reinforcement learning algorithm developed to manipulate three degrees of freedom, one flight condition parameter, and one material parameter. The degrees of freedom, flight condition parameter, and material parameter are

- Airfoil thickness
- Camber
- Chord
- Airfoil angle-of-attack
- Elastic Modulus

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 grid must be a sinusoidal spaced grid in the x direction, which puts more points at the trailing edge of the airfoil. This type of grid is necessary because many aerodynamic changes occur near the trailing edge. If the number of panels were to decrease, the accuracy of the model would also decrease. However, as the number of panels generated is increased, the computational time of the model increases as well. Thus, a balance is needed between accuracy and computational time. This balance can be achieved by finding a set number of panels for which any increase from that number of panels yields a minimal accuracy increase. For example, assume 50 panels are chosen initially. If the number of panels were changed to 100 and the accuracy of the model increased by 10 per cent, this increase in the number of panels would be deemed necessary. If the number of panels were changed from 100 to 150 and the accuracy of the model increased by less than 1 per cent, then this increase in the number of panels would be deemed unescessary, and 100 panels is chosen as the correct number of panels to use.

A. Airfoil Model Development

The aerodynamics of the airfoil is modeled by a constant strength doublet panel method. For each panel consider first the velocities in the local panel coordinate system

$$u_p = \frac{\mu}{2\pi}\left[\frac{z}{(x-x_1)^2+z^2} - \frac{z}{(x-x_2)^2+z^2}\right] \tag{5}$$

$$w_p = \frac{\mu}{2\pi}\left[\frac{x-x_1}{(x-x_1)^2+z^2} - \frac{x-x_2}{(x-x_2)^2+z^2}\right] \tag{6}$$

Since these equations require panel coordinates, a transformation from the global coordinate system to the local panel coordinate system must be made. The following equation represents the coordinate transformation used[20]

$$\begin{pmatrix} x \\ z \end{pmatrix}_p = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{pmatrix} \begin{pmatrix} x - x_0 \\ z - z_0 \end{pmatrix} \tag{7}$$

The panel method is based on the no penetration condition, which states that the flow cannot cross the solid boundary of the airfoil, thus the velocity normal to the surface is 0 in the global coordinate system. Eq. 8 transforms the velocities from Eqs. 5 and 6 into the global coordinate system.[20]

$$\begin{pmatrix} u \\ w \end{pmatrix} = \begin{pmatrix} \cos(\theta_i) & \sin(\theta_i) \\ -\sin(\theta_i) & \cos(\theta_i) \end{pmatrix} \begin{pmatrix} u_p \\ w_p \end{pmatrix} \tag{8}$$

Now it is possible to solve for the doublet strengths using Eqs. 5-8. These doublet strengths are then used to find the tangential velocities at each point. Once the tangential velocities are calculated, the pressure coefficient can be calculated using Bernoulli's equation which, when modified, produces Eq. 9.[21]

$$C_p = 1 - \frac{u^2+w^2}{V_\infty^2} \tag{9}$$

The pressure coefficient can be broken up into normal and axial forces using simple integration. These forces can also be further broken up into lift and drag using simple trigonometry.[21]

$$C_n = \frac{1}{c}\int_0^c \left(C_{p_{lower}} - C_{p_{upper}}\right) dx \tag{10}$$

$$C_a = \frac{1}{c}\int_0^c \left(C_{p_{upper}}\frac{dy_{upper}}{dx} - C_{p_{lower}}\frac{dy_{lower}}{dx}\right) dx \tag{11}$$

$$C_l = C_n\cos(\alpha) - C_a\sin(\alpha) \tag{12}$$

$$C_d = C_n\sin(\alpha) + C_a\cos(\alpha) \tag{13}$$

Once the pressure distribution has been calculated, the pressures found are assumed to be the loads on the airfoil, which are needed to apply basic beam theory. The pressure distribution is approximated using a curve fit function so that the stress values can be easily calculated. One disadvantage of using a curve fit function is the resulting curve is not the exact pressure distribution and thus an error will be present. In order to decrease this error, the order of the polynomial fitted to the pressure distribution should be increased. As with the panel accuracy case, too large of an increase in the order of the polynomial will result in increased compuational time as well as a minimal accuracy increase. Once again, a balance is necessary in order to achieve a high accuracy with low compuational time. The balance between accuracy and compuational time is found by using the same method as when finding the correct number of panels to be used. The stress values for both the upper and lower surfaces of the airfoil are calculated using Eqs. 14 and 15.[22] Eq. 14 is the basic boundary value problem from Euler-Bernoulli Beam Theory. Once Eq. 14 is solved, Eq. 15 is used to calculate the stress at any point on the surface.

$$\frac{dM}{dx} = P \tag{14}$$

$$\sigma_{xx} = \frac{-M \cdot y}{I_{zz}} \tag{15}$$

IV. Model Validation and Verification

This section validates the mathematics and physics of the airfoil model, and then verifies that it accurately represents the desired airfoil. For the validation, two representative airfoils were examined for correct pressure and lift profile characteristics corresponding to a conventional airfoil. For verification, wind tunnel data from Ref. 23 was compared to the data generated by the doublet panel model.

A. Aerodynamic Validation

The first airfoil modeled was a NACA 0012, representing the class of symmetric airfoils. The constant strength doublet panel method was applied producing the pressure distribution illustrated in Figure 1. The angle-of-attack is held at 0°, so there should be zero calculated lift on the airfoil. The pressure distribution is shown in the following figure:

An asymmetric airfoil was also modeled. The representative airfoil chosen was the NACA 4412. Figure 2 shows the pressure distribution produced by the constant strength doublet panel code for a NACA 4412 held at 0°.

Calculating the lift and drag based on Figure 1 and Figure 2 result in no lift on the symmetric airfoil and positive lift on the asymmetric airfoil. These results are reasonably accurate with respect to the physics involved, and indicate that the airfoil model produces reasonably correct aerodynamics. However, as can be seen from Figure 1 and Figure 2, there are problems associated with the trailing edge in each of these airfoils. This discrepancy is due to the singularity present at the trailing edge. The singularity at the trailing edge is a source of error for constant strength doublet panel methods that must be taken into account when analyzing results.

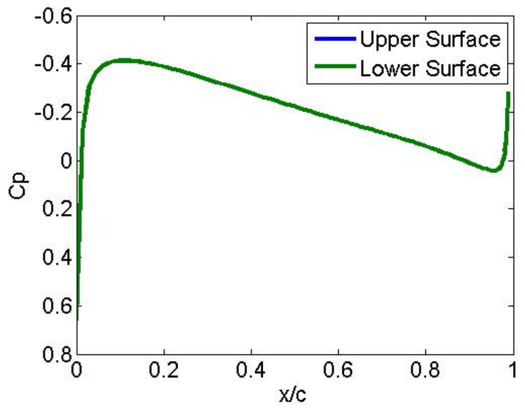

Figure 1. Pressure Distribution Across a NACA 0012 Airfoil

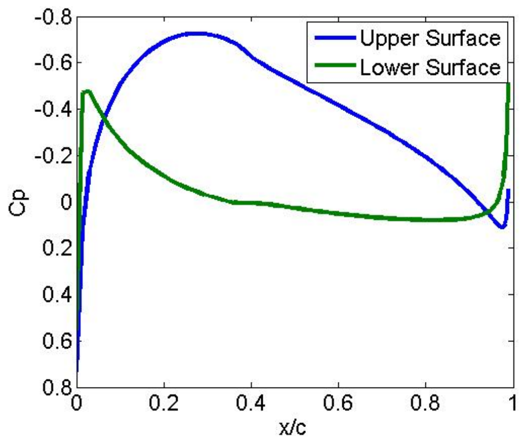

Figure 2. Pressure Distribution Across a NACA 4412 Aifoil.

B. Structural Validation

Structural analysis of the NACA 0012 and NACA 4412 based on the doublet panel model was also conducted. The stress distribution for both the NACA 0012 and NACA 4412 are illustrated in Figure 3 and Figure 4, respectively. Both assume sea level conditions and a speed of about 100 m/s.

These figures show how the stress varies with position along both the upper and the lower surface of the airfoils. The stress values appear to be low and require further investigation. However, the general shape of stress distribution is to be expected. There is greater stress toward the center of the airfoil where there is greater bending moment. Also, the distribution for the symmetric airfoil shows that the stress is equal for both surfaces adhering to the characteristic symmetry of the airfoil, whereas there is a slight difference in the asymmetric airfoil stress distribution. The top and bottom surfaces are not mirrors of each other, which thus results in different stress distributions.

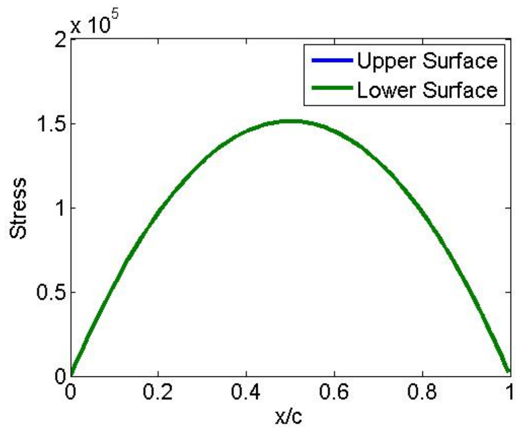

Figure 3. Stres Distribution of a NACA 0012

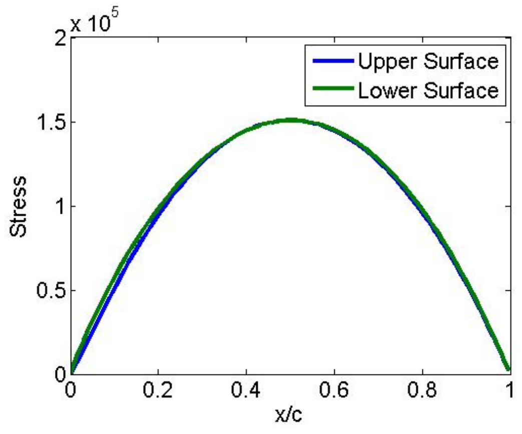

Figure 4. Stress Distribution of a NACA 4412

C. Verification

Several sets of airfoil data were compared to verify the accuracy of the model within the scope of this research. For brevity, only one set of data is presented here. The lift curve produced by the model for the NACA 0012 is compared to wind tunnel data from Ref. 23 and is depicted in Figure 5.

Since the doublet panel model is limited the linear range of the airfoil angle-of-attack, the angles-of-attack only ranged from -5 to 5 degrees. When compared to data from Ref. 23, there were some discrepancies, but they are considered mild and acceptable for this problem. Some of these discrepancies arise from the singularity inherent in this modeling method, while others come from the limitations and assumptions needed to apply the method.

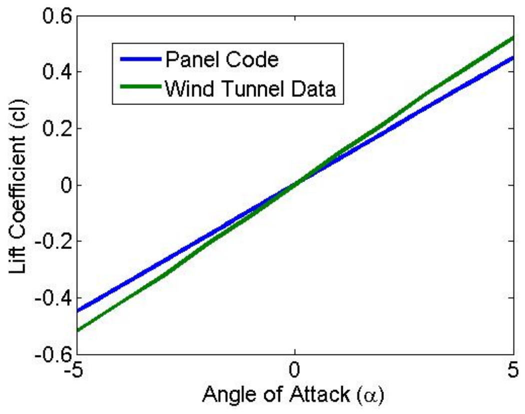

Figure 5. Lift Curve Slope for a NACA 0012

V. Morphing Airfoil Implementation

The morphing of the airfoil entails changing the thickness and camber at this stage in the development of this methodology. The reinforcement learning module specifies these two parameters corresponding to the current flight condition. It is assumed that the airfoil is composed of a smart material whose shape is affected by applying voltage. For simulation purposes, the morphing dynamics are assumed to be simple nonlinear differential equations.

The optimality of the airfoil is defined by some combination of the following characteristics

- Airfoil lift coefficient
- Airfoil drag coefficient
- Airfoil moment coefficient about the leading edge

The aerodynamic model and the reinforcement learning module interact significantly during both the learning stage, when the optimal shape is learned, and the operational stage, when the airfoil morphs from state to state. The purpose of the reinforcement learning module is to learn the optimal series of actions necessary to command both the thickness and camber, and thus the airfoil, into the optimal shape for a given flight condition. The two parts of the system interact as follows. Initially, the reinforcement learning module commands a random action from the set of admissible actions. The admissible actions for this problem are

- -0.10% thickness
- -0.10% thickness
- -0.10% camber
- -0.10% camber

The agent implements this action by submitting it to the plant which produces a shape change. The reward associated with the resultant shape is evaluated. The resulting state, action, and reward set is then stored in a database. Then a new action is chosen, and the sequence repeats itself for some predefined number of episodes or until the agent reaches a goal state. Shape changes in the airfoil due to actions generated by the reinforcement learning module cause the aerodynamics associated with the airfoil to change. The aerodynamic properties of the airfoil define the reward, as stated, and the structural analysis offers a constraint on the limits of the morphing degrees of freedom.

Due to the continuous nature of the states, rather than just a positive reward for a single goal state, a positive reward is given when the airfoil meets some specified range of the aerodynamic properties listed above. The reward is selected to be a combination of the aerodynamic properties of the current airfoil, which is dependent on the current shape and current flight condition. The current flight condition is identified uniquely by altitude, flow velocity, and angle-of-attack. As a result there is a goal region rather than a single goal state that the agent must aspire to reach.

VI. Numerical Examples

The simulation was used to test the integration of the reinforcement learning module with the aerodynamic model by evaluating several reward schemes. For the first example the goal is defined by the lift coefficient only. The agent is looking for a configuration that meets a minimum lift coefficient requirement. The second example adds complexity by having the agent also try to meet a maximum drag requirement. This additional requirement effectively narrows the goal region previously defined by the minimum lift coefficent requirement. The final example adds one more requirement in the form of a maximum moment coefficient about the leading edge of the airfoil, further reducing the goal region. The requirements for each case are enumerated in Table 1.

Table 1. Reward Region for Examples 1-3

	Example #1	Example #2	Example #3
c_l	≥ 0.4	≥ 0.4	≥ 0.4
c_d	N/A	≤ 0.003	≤ 0.003
$c_{m_{le}}$	N/A	N/A	≤ 0.87

For the purpose of direct comparison, the chord, angle-of-attack, number of episodes, boundary reward, and goal reward are the same for each example. The values are listed below.

- Chord $= 1m$
- Angle-of-attack $= 2.0°$

- Number of episodes = 5000
- Boundary reward = −20
- Goal reward = 20

For each of the 5000 episodes, the agent begins in a random initial state that is not classified as a goal state. It explores the state space of thickness-camber combinations until it hits the predefined limit of total number of actions or finds a goal state. Should the agent run into a boundary, that boundary location is noted, and the agent chooses another action.

The boundary is defined as the thickness and camber limits set on the airfoil. The agent is not allowed to venture beyond these limits and is given negative reinforcement whenever it attempts to do so. The limits for these examples are

- Thickness lower limit - 10% chord
- Thickness upper limit - 18% chord
- Camber lower limit - 0% chord
- Camber upper limit - 5% chord

These limits were chosen because they are representative of low speed airfoils that might be found on small UAVs or micro air vehicles.

A. Example #1: Lift Coefficient Goal

This example tests the learning algorithm in which the goal is defined by a minimum lift coefficient the agent must find based on the aerodynamic calculations given the current state's thickness-camber pair. The goal region is therefore all of the pairs that meet the minimum lift coefficient requirement of 0.4.

Figure 6 shows the evolution of $Q(s, a)$. Given the 4-dimensional nature of $Q(s, a)$ (thickness, camber, action, action-value) only one action can be displayed at a time. Figure 6 illustrates the action-value for each thickness-camber pair for the action of an +0.10% increase in camber as the number of episodes increases. Very early on the agent learns that the goal region encompasses higher values of camber as indicated by the positive values of the action-value function. These positive values indicate a positive preference for this action illustrated. If this preference is greater than the preferences for the other three actions, then the agent will choose this action given that it is acting in a greedy manner. This preference becomes more pronounced as the number of episodes increases. It is also apparent that as the number of episodes increases the surface suggested by the values becomes more fleshed out. The action-value for pairs of somewhat lower thickness becomes more positive. This trend indicates that agent is learning that at these lower camber values if it

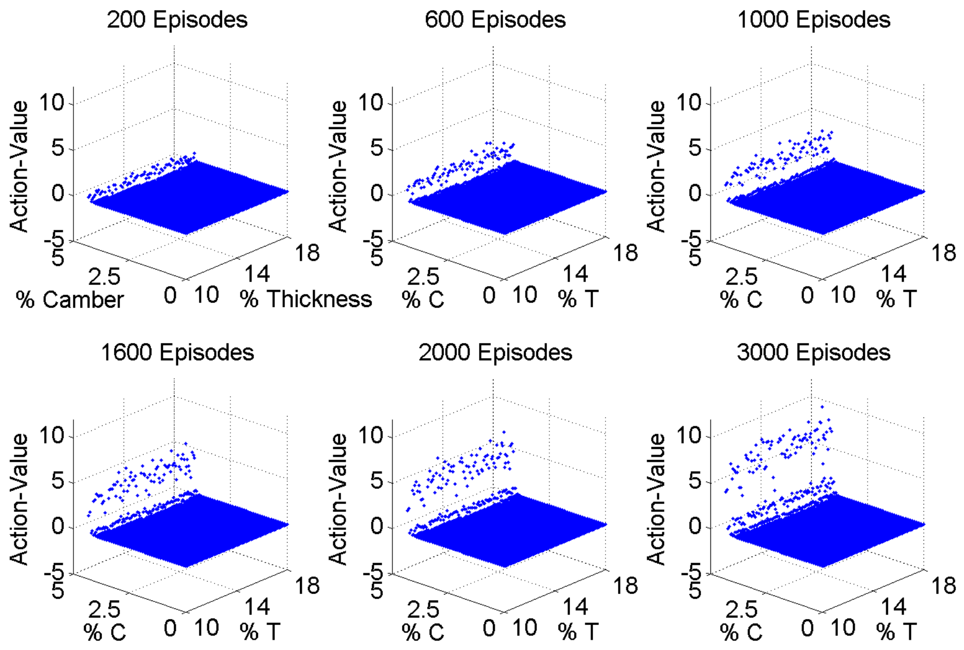

Figure 6. $Q(s, a)$ Evolution for an Increase in Camber for Example #1

chooses to increases camber it will get closer to the goal region. Finally, Figure 6 shows that at any given thickness, there is at least one value of camber that is in the goal region.

A Monte Carlo simulation was conducted using the learned action-value function. The function was recorded every 200 episodes. The simulation entails 500 episodes for each recorded action-value function in which the agent chooses the greedy 95% of the time and chooses a random, non-greedy action 5% of the time. The agent begins in a random initial state and must navigate to the goal region. Every success is recorded, and each boundary encountered resulted in the cessation of that episode. Figure 7 shows the trend of success as the number of episodes increases. The percent success increases rapidly and asymptotically approaches 100% success, meaning that the agent navigated to a goal state in the goal region every episode. Figure 7 indicates that by using this learning algorithm, the agent is successful 90% of the time by 1600 episodes and approximately 100% of the time by 2600 episodes given the definition of the goal for this example. There is a slight decrease in success from 1000 to 1200 episodes. This occurrence is most likely due to the occasionaly random actions taken by agent resulting in the agent encountering boundaries more often than in the preceding and succeeding cases.

Figures 8 and 9 illustrate one successful episode using the final learned action-value function. The agent's initial state is 11% thickness and 0% camber. It uses the learned function to navigate from this initial state to a state in the goal region. The agent chooses an action every 0.5 sec, and either the thickness or camber changes according to arbitrarily chosen simple nonlinear dynamics. Figure 8 shows the airfoil configuration

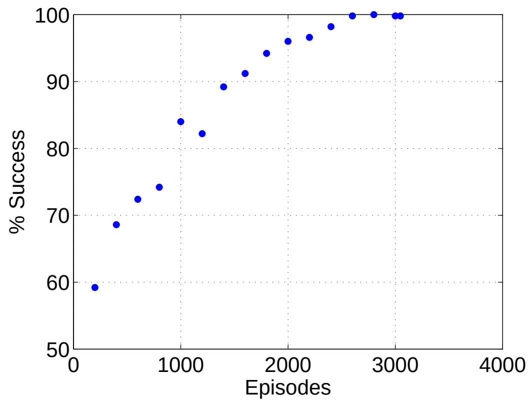

Figure 7. Monte Carlo Simulation Using $Q(s,a)$ for Every 200 Episodes for Example #1

for the initial and final state. Figure 9 shows how the thickness and camber change with time. The resulting lift coefficient in Figure 9 shows that the agent chooses actions that takes it steadily toward the lift coefficient goal of 0.4 until it reaches a thickness and camber combination that corresponds to a calculated lift coefficient greater than 0.4.

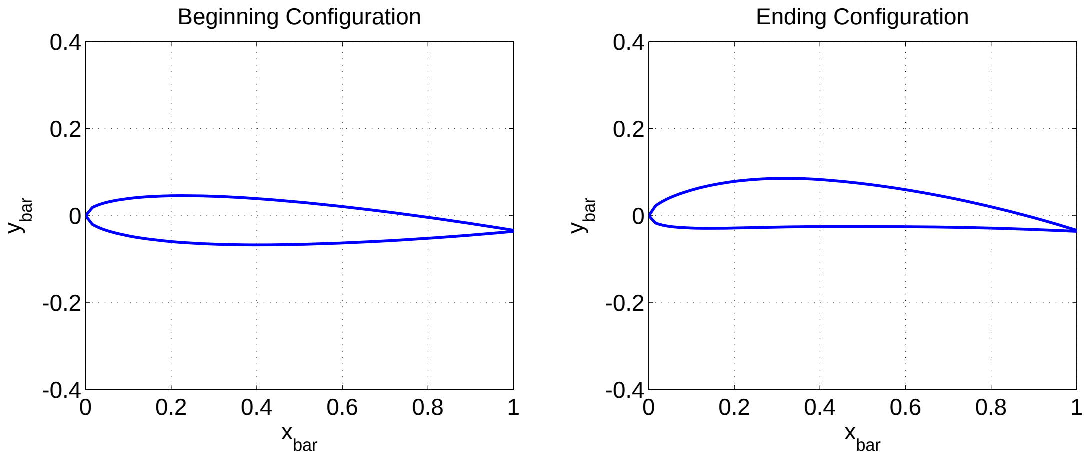

Figure 8. Initial and Final Airfoil Configuration for Example #1

B. Example #2: Lift and Drag Coefficient Goal

This example tests the learning algorithm in which the goal is defined by a minimum lift coefficient and a maximum drag coefficent the agent must find based on the aerodynamic calculations. The goal region is all of the pairs that meet the minimum lift coefficient requirement 0.4 and the maximum drag coefficient requirement of 0.003 as defined in Table 1.

Figure 10 shows the evolution of $Q(s,a)$ for the action of an $+0.1\%$ increase in camber for this goal

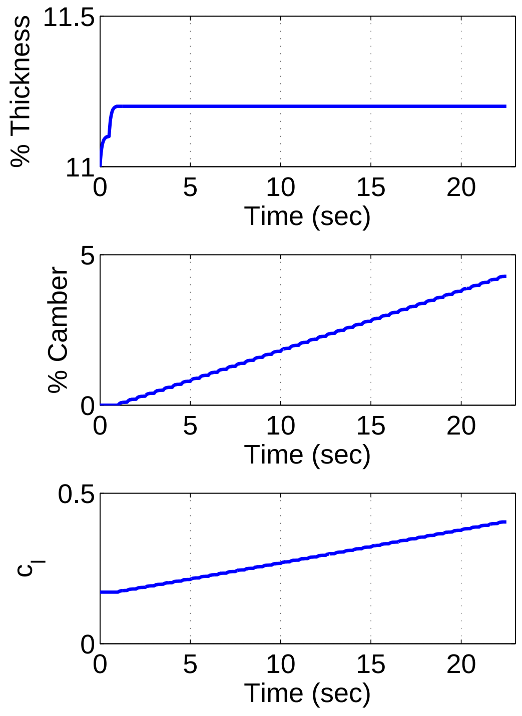

Figure 9. State Progression Using Greedy Policy for Example #1

definition. Again the agent learns that the goal region encompasses higher values of camber. When compared to Figure 6, it can be seen that for this goal definition the goal region is much narrower than previously. The addition of the drag coefficient requirement restricts even further the possible thickness-camber pairs that satisfy the goal requirements. Also, as the number of episodes increases, the surface for this action becomes more pronounced, though not as rapidly as the previous example. Like the previous example, however, Figure 10 shows that at any value of thickness, there is at least one value of camber such that the pair meets the goal requirement.

Figure 11 displays the results of the Monte Carlo simulation for this example. The success rate approaches 100% more rapidly than in the previous example. Figure 11 indicates that using this learning algorithm with the more restricting goal requirement, the agent is successful 90% of the time by 600 episodes approximately 100% successful by 2000 episodes. The small oscillations in success rate as the number episodes increases and the marked decrease in success from 800 to 1000 episodes is most likely due to the occasional random actions taken by the agent that resulted in a cessation of the current episode.

Figures 12 and 13 illustrate one successful episode using the final learned action-value function given the goal requirements for this example. The initial state is 11% thickness and 0% camber. It uses the learned function for this example to navigate to the goal region without encountering a boundary. Figure 12 shows

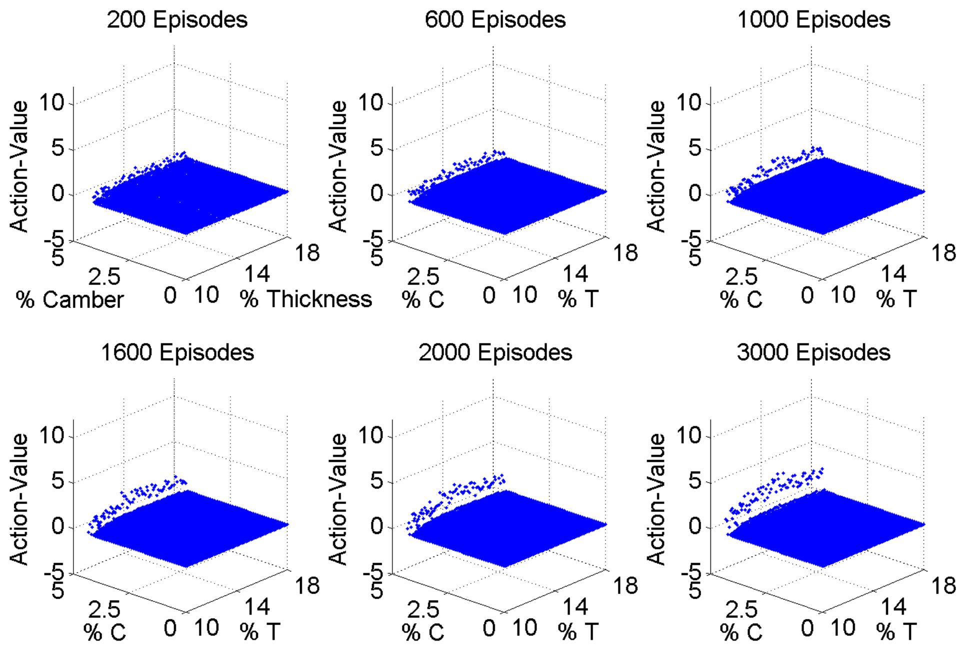

Figure 10. $Q(s,a)$ **Evolution for an Increase in Camber for Example #2**

the initial and final configuration of the airfoil, and Figure 13 shows how the agent navigates from one to the other. The agent chooses actions that takes it directly toward a state that has a lift coefficient greater than 0.4 and a drag coefficient less than 0.003. In this case the agent doesn't choose to change thickness at all. It instead prefers to increase camber steadily until the goal region is reached. The lift coefficient and drag coefficient calculated by the aerodynamic module reflect this as they approach their respective goals as a result of the actions that agent chooses.

C. Example #3: Lift, Drag, and Moment Coefficient Goal

This example further tests the learning algorithm by restricting the goal region even more by adding a maximum moment coefficient about the leading edge the agent may not exceed in addition to the minimum lift coefficient and a maximum drag coefficent the agent must adhere to. The goal region is thus all the pairs that meet the minimum lift coefficient requirement 0.4, the maximum drag coefficient requirement of 0.003, and the maximum moment coefficient about the leading edge as defined in Table 1.

Figure 14 depicts the evolution of $Q(s,a)$ for the action of an +0.1% increase in camber for this goal definition. The contour of the surface for this goal definition is much different than in the previous two examples. There are two small regions that meet all the requirements defined in Table 1. The locations of the two become more apparent as the positive preferences become more pronounced as the number of episodes increases. The smaller region is centered near 12% thickness and 4.7% camber. The larger region

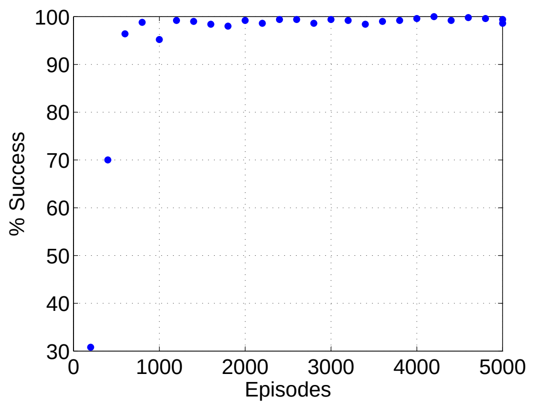

Figure 11. Monte Carlo Simulation Using $Q(s, a)$ for Every 200 Episodes for Example #2

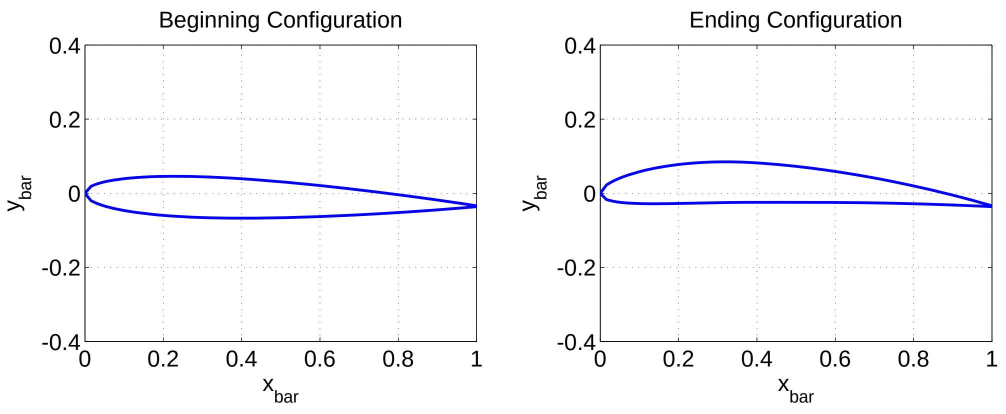

Figure 12. Initial and Final Airfoil Configuration for Example #2

is in the corner of large percent thickness and large percent camber. This figure also shows the result of the agent encountering a boundary - negative action-values. These negative preferences tell the agent that a boundary is near and it should choose some other action.

Despite this smaller goal region, the success rate shown in the Monte Carlo simulation results in Figure 15 approaches 100% asymptotically, similar to the previous examples. The learning algorithm is able to cope with the more restricting goal requirement reaching 90% success rate by 1400 episodes. The agent is almost 100% successful by 1600 episodes. Similar to the previous examples, there are small oscillations just below 100% success as the number of episodes increases. These oscillations are possibly due to the occasional random action the agent takes. Given the smaller qoal region there is more exposed boundary, and therefore the agent is more likely to take a rendom action and encounter that boundary. Figure 15, however, shows that this adversely effect on performanceis within acceptable bounds.

Figures 16 and 17 illustrate one successful episode using the final learned action-value function given the

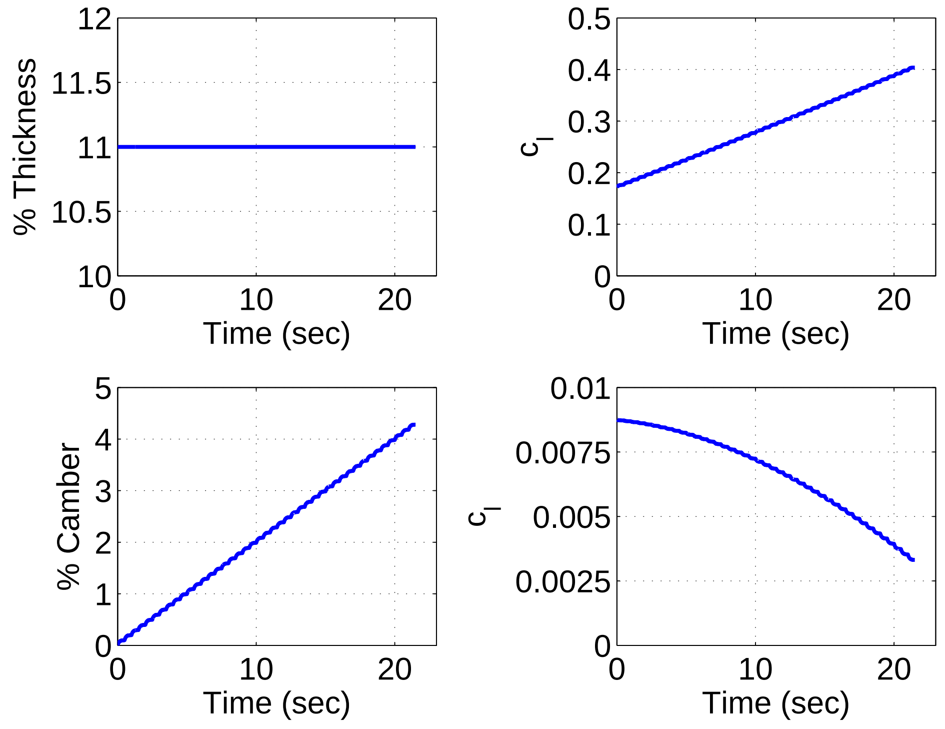

Figure 13. State Progression Using Greedy Policy for Example #2

goal requirements for this example. The initial state is 11% thickness and 0% camber as in the previous two exmples. It uses the learned function for this example to navigate to the much smaller goal region without encountering a boundary. Figure 16 shows the initial and final configuration of the airfoil, and Figure 17 shows how the agent navigates from one to the other. It takes a longer period of time for the agent to reach the larger of the two goal regions, almost twice that of the previous two examples. The reason is that it must choose to change both thickness and camber many times to traverse the distance from the initial state to a goal state. These choices are shown in Figure 17 as the agent alternates between choosing to change thickness and choosing to change camber. This figure also shows the changes in lift, drag, and moment coefficient as the agent navigates to a configuration that meets all of the defined requirements.

VII. Conclusions

This paper develops a methodology for the manipulation of a morphing airfoil, combining machine learning and analytical aerodynamic calculations. For a given set of aerodynamic requirements the Reinforcement Learning learns the policy to morph in two degrees of freedom, thickness and camber, from some initial state to a final state that meets requirements. Development of the aerodynamic model, the Reinforcement Learning module, and the implementation of the learning algorithm were presented. The methodology was

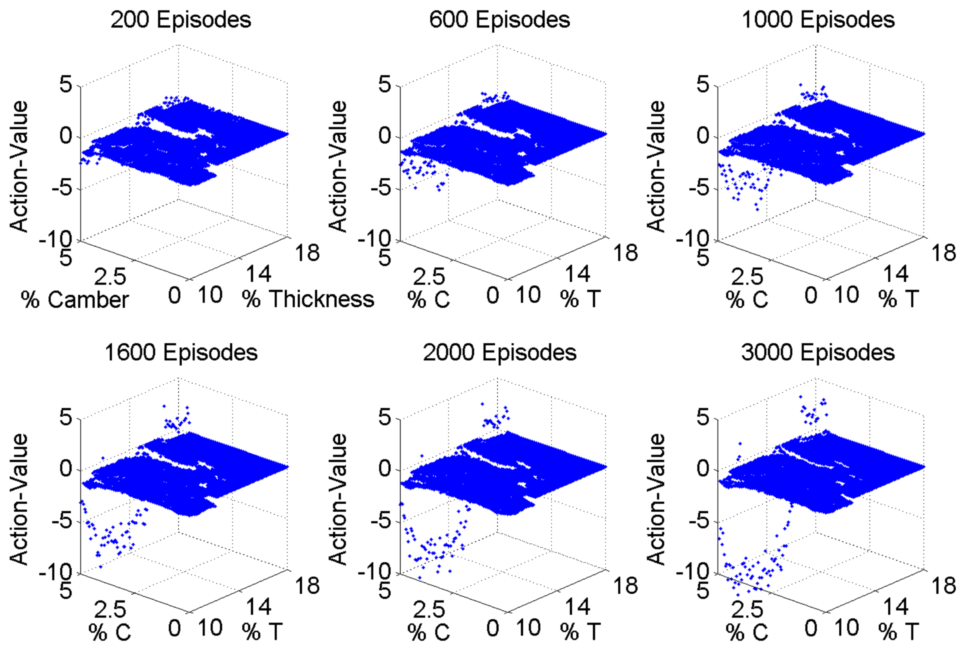

Figure 14. $Q(s,a)$ **Evolution for an Increase in Camber for Example #3**

demonstrated by three numerical examples with increasingly more restrictive aerodynamic requirements.

Based on the results in this paper, it is concluded that:

1. For the numerical examples presented, the Reinforcement Learning Module learns the control policy that results in an approximately 100% success rate in less than 3000 episodes. This was accomplished by allowing the agent the choice to explore during every episode. This control policy is not optimal, but it results in good performance by the agent.

2. Restricting the actions to discrete values is a promising candidate for handling the continuous airfoil shape state defined by thickness and camber. The agent was able to sufficiently learn the action-value function despite the fact that it could not visit each of the infinite possible states. The reason is that it visited enough states to learn general topography of the action-value function.

3. Use of the K-Nearest Neighbor method resulted in adequate approximation of unknown action-values allowing the agent to learn a successful control policy.

4. The Reinforcement Learning agent is sensitive to the discount rate, γ, and learning rate, α. If either is too large or too small, the incremental change in the action-value function becomes either too pronounced or too subtle, resulting in the action-value function not converging. Many iterations resulted in the successful combination presented in this paper.

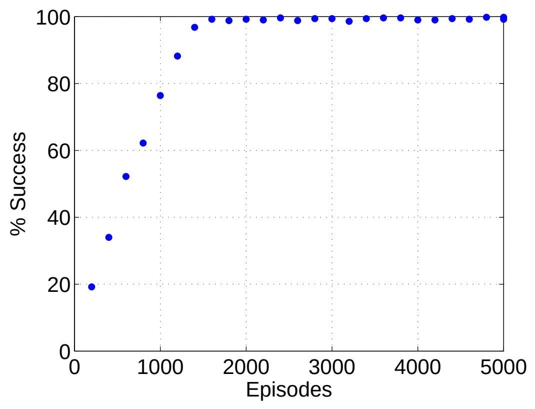

Figure 15. Monte Carlo Simulation Using $Q(s, a)$ for Every 200 Episodes for Example #3

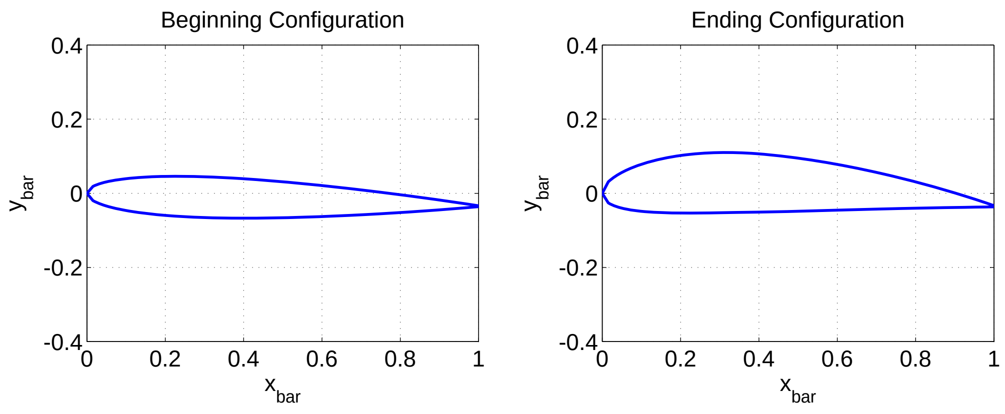

Figure 16. Initial and Final Airfoil Configuration for Example #3

VIII. Acknowledgement

The authors wish to acknowledge the support of the Texas Institute for Intelligent Bio-Nano Materials and Structures for Aerospace Vehicles. The material is based upon work supported by NASA under award no. NCC-1-02038. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

References

[1]Wlezien, R., Horner, G., McGowan, A., Padula, A., Scott, M., Silcox, R., and Simpson, J., “The Aircraft Morphing Program,” , No. AIAA-98-1927.

[2]McGowan, A.-M. R., Washburn, A. E., Horta, L. G., Bryant, R. G., Cox, D. E., Siochi, E. J., Padula, S. L., and Holloway, N. M., “Recent Results from NASA’s Morphing Project,” *Proceedings of the 9th Annual International Symposium on Structures*

and Materials, No. SPIE Paper Number 4698-11, San Diego, CA, 17-21 March 2002.

[3]Wilson, J. R., "Morphing UAVs change the shape of warfare," *Aerospace America*, February 2004, pp. 23–24.

[4]Bowman, J., Weisshaar, T., and Sanders, B., "Evaluating The Impact Of Morphing Technologies On Aircraft Performance," *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, No. AIAA-2002-1631, Denver, CO, 22-25 April 2002.

[5]Scott, M. A., Montgomery, R. C., and Weston, R. P., "Subsonic Maneuvering Effectiveness of High Performance Aircraft Which Employ Quasi-Static Shape Change Devices," *Proceedings of the SPIE 5th Annual International Symposium on Structures and Materials*, San Diego, CA, March 1-6 1998.

[6]Valasek, J., Tandale, M. D., and Rong, J., "A Reinforcement Learning - Adaptive Control Architecture for Morphing," *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, No. AIAA-2004-6220, Chicago, IL, 20-22 September 2004.

[7]Tandale, M., Rong, J., and Valasek, J., "Preliminary Results of Adaptive-Reinforcement Learning Control for Morphing Aircraft," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, No. AIAA-2004-5358, Providence, RI, 16-19 August 2004.

[8]Doebbler, J., Tandale, M., and Valasek, J., "Improved Adaptive-Reinforcement Learning Control for Morphing Unmanned Air Vehicles," *Infotech@Aerospace*, No. AIAA-2005-7159, Arlington, VA, 26-29 September 2005.

[9]Jr, J. E. H., "Dynamic Shape Control of a Morphing Airfoil Using Spatially Distributed Transducers," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 3, 2006, pp. 612–616.

[10]Bellman, R. E., *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[11]Bellman, R. E. and Dreyfus, S. E., *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ, 1962.

[12]Bellman, R. E. and Kalaba, R. E., *Dynamic Programming and Modern Control Theory*, Academic Press, New York, 1965.

[13]Sutton, R. S., "Learning to Predict by the Method of Temporal Differences," *Machine Learning*, Vol. 3, 1998, pp. 9–44.

[14]Williams, R. J. and Baird, L. C., "Analysis of some incremental variants of policy iterations: First Steps toward Understanding Actor-Critic Learning Systems," Tech. Rep. NU-CCS-93-11, Boston, 1993.

[15]Watkins, C. J. C. H. and Dayan, P., *Learning from Delayed Rewards*, Ph.D. thesis, University of Cambridge, Cambridge, UK, 1989.

[16]DeJong, G. and Spong, M. W., "Swinging up the acrobot: An example of intelligent control," *Proceedings of the American Control Conference*, 1994, pp. 2158–2162.

[17]Boone, G., "Minimum-time control of the acrobot," *International Conference on Robotics and Automation*, Albuquerque, NM, 1997.

[18]Sutton, R. S., "Generalization in reinforcement learning: Successful examples using sparse coarse coding," *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, edited by D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, MIT Press, Cambridge MA, pp. 1038–1044.

[19]Sutton, R. and Barto, A., *Reinforcement Learning - An Introduction*, The MIT Press, Cambridge, Massachusetts, 1998.

[20]Katz, J. and Plotkin, A., *Low Speed Aerodynamics: Second Edition*, Cambridge University Press, 2001.

[21]Anderson, J. D., *Fundamentals of Aerodynamics*, McGraw-Hill, 2001.

[22]Allen, D. and Haisler, W., *Introduction to Aerospace Structural Analysis*, John Wiley and Sons, 1985.

[23]Abbot, I. and Van Doenhoff, A., *Theory of Wing Sections*, Dover Publications, 1959.

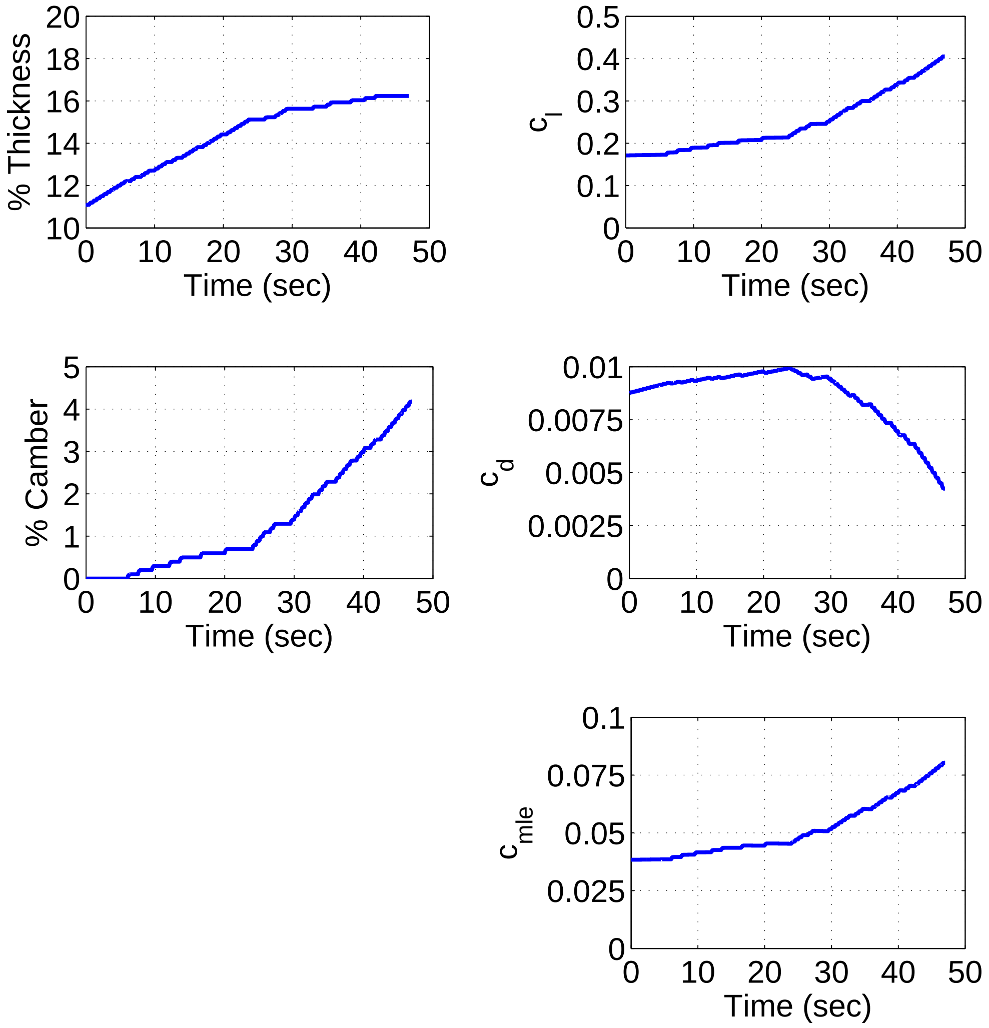

Figure 17. State Progression Using Greedy Policy for Example #3