A Reinforcement Learning – Adaptive Control Architecture for Morphing

John Valasek, Monish D. Tandale, & Jie Rong
Aerospace Engineering Department
Texas A&M University

AIAA 1st Intelligent Systems Technical Conference
20-23 Sep 2004
Chicago, IL
Overview

- Introduction to Morphing
- Simplified Model of a Morphing Vehicle
- Reinforcement Learning Module
- Structured Adaptive Model Inversion Control
- Adaptive-Reinforcement Learning Control Architecture Functionality
- Numerical Example
- Conclusions
- Future Work
Big Picture Research Goals

- **WHEN** to reconfigure
- **HOW** to reconfigure
- **LEARNING** to reconfigure
Which Morphing?

- **Morphing for Mission Adaptation**
  - Large scale, relatively slow, in-flight shape change to enable a single vehicle to perform multiple diverse mission profiles

as opposed to:

- **Morphing for Control**
  - In-flight physical or virtual shape change to achieve multiple control objectives (maneuvering, flutter suppression, load alleviation, active separation control)

John Davidson, NASA Langley, AFRL Morphing Controls Workshop – Feb 2004

Valasek, Tandale, & Rong  AIAA-2004-6220-4
Synthetic Jets for Virtual Shaping and Separation Control

MultiSensor MEMS Arrays for Flow Control Feedback

Adaptive Controller

Sensed Information Aggregation

Control Information Distribution

Reconfiguration Command Generation

System Performance Evaluation

Knowledge Base

Environment

Control Architecture
Adaptive-Reinforcement Learning Control (A-RLC)

Conceptual Control Architecture for Reconfigurable Aircraft

SAMI
Structured Adaptive Model Inversion (Traditional Control)

Flight controller to handle wide variation in dynamic properties due to shape change

RL
Reinforcement Learning (Intelligent Control)

Learn the morphing dynamics and the optimal shape at every flight condition in real-time
Morphing Vehicle Model
Smart Block

- **Smart Block Shape**
  - Rectangular parallelepiped
  - Constant volume \( V \) during change
  - 2 independent variables: \( y \) and \( z \)

\[
x = \frac{V}{yz}
\]

- **Morphing Dynamics**
  - Smart material: carbon nanotubes or shape memory alloy
  - Nonlinear dynamic model relating dimensions to voltage
Morphing Dynamics

Y-morphing

Z-morphing

Time (sec)

y dimension

z dimension
Smart Block Dynamics

mathematical model of dynamic behavior

- **Variables**
  \[ p_c = [d_x, d_y, d_z]^T \quad v_c = [u, v, w]^T \]
  \[ \sigma = [\phi, \theta, \psi]^T \quad \omega = [p, q, r]^T \]

- **Nonlinear 6–DOF Equations**
  
  - Kinematic level: \( \dot{p}_c = J_1 v_c \quad \dot{\sigma} = J_1 \omega \)
  
  - Acceleration level: \( m \ddot{v}_c + \tilde{\omega} m v_c = F + F_d \)
    \[ I \ddot{\omega} + \dot{I} \omega + \tilde{\omega} I \omega = M + M_d \]

- **Drag Force**
  
  - Function of air density, square of velocity along axis, and area of the smart block perpendicular to the axis

  additional dynamics due to morphing
Reinforcement Learning
1. **Actor** takes action based upon states and preference function
2. **Critic** updates state value function, and evaluates action
3. Actor updates preference function

Learning is done repetitively, by subjecting to different scenarios
Reinforcement Learning

self training

- Exploration-exploitation dilemma
  - uniform probability policy $\rightarrow$ Gibbs softmax policy $\rightarrow$ greedy policy

- Limited training examples
  - Only at 6 discrete flight conditions: $\{0, 1, 2, \ldots, 5\}$

- Tile code
  \[
  \Phi_j(s) = \begin{cases} 
  1, & \text{if } (s - c_j)^T R^{-1} (s - c_j) \leq 1 \\
  0, & \text{if } (s - c_j)^T R^{-1} (s - c_j) > 1 
  \end{cases}
  \]

  - Center state vector set: $\{2, 2.125, 2.25, \ldots, 5\} \times \{0, 1, 2, \ldots, 5\}$
  - Weight matrix:

\[
R^{-1} = \begin{bmatrix} r_1^2 & 0 \\ 0 & r_2^2 \end{bmatrix}^{-1}, r_1 = 0.1, r_2 = 0.0625
\]
Structured Adaptive Model Inversion Control
Adaptive Control

model reference adaptive control

Reference model

Controller

Plant

Adaptation Law

Estimated parameters

$y_m$

$r$

$u$

$y$

$e$

$\text{Estimated parameters}$
Dynamics

2nd order differential equations

\[ F = m a \]

\[ \dot{x} = v \]

\[ \dot{v} = a = \frac{F}{m} \]

Exact kinematic relationship between position and velocity

Acceleration level relationships between forces and system parameters
Structured Adaptive Model Inversion
Subbarao, Junkins (Texas A&M)

Features

- Dynamic inversion inner-loop, with an MRAC outer-loop to handle system uncertainties.
- Controls are solved for explicitly:
  
  \[
  \dot{x} = Af(x) + Bu \\
  x_r, \dot{x}_r \\
  u = B^{-1}(\dot{x}_r - Af(x) - \lambda e) \text{ so that the error dynamics becomes } \dot{e} = -\lambda e
  \]
  
  - Undesirable dynamics are cancelled and replaced with user specified desired dynamics.
  
  - Easily applicable to nonlinear systems.
  
  - Error dynamics can be specified.
  
  - Shown to be very effective for a wide variety of systems.
Structured Adaptive Model Inversion Features

Trajectory Tracking for Dynamic Systems

- **Plant:**
  - Nonlinear in states, affine in control, uncertain parameters appear linearly.

- **Control:**
  - Dynamic Inversion and Sliding Mode Control.
  - Dynamic Inversion requires knowledge of system parameters, which are inherently uncertain.

- **Adaptive Learning Parameters:**
  - Updated in real-time, and used for the Dynamic Inversion

- **Adaptation Mechanism:**
  - Driven by the error between the actual plant trajectory and the reference trajectory

- **Stability Analysis:**
  - Guarantees that the plant trajectory, asymptotically converges to the reference trajectory in the presence of Parametric Uncertainties, and initial condition errors.
Structured Model & Minimal Parameterization

Kinematic Level Model
\[ \dot{p}_c = J_l \dot{v}_c \]
\[ \dot{\sigma} = J_a \omega \]

Acceleration level Model
\[ m\dot{v}_c + \tilde{\omega}mv_c = F + F_d \]
\[ I\dot{\omega} + \dot{I}\omega + \tilde{\omega}I\omega = M + M_d \]

Attitude Control
\[ I^*_a(\sigma)\ddot{\sigma} + C^*_a(\sigma, \dot{\sigma})\dot{\sigma} = P^T_a(\sigma)M \]

Minimal Parametrization of the Inertia Matrix
\[
\begin{bmatrix}
I_{11} & I_{12} & I_{13} \\
I_{12} & I_{22} & I_{23} \\
I_{13} & I_{23} & I_{33}
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3
\end{bmatrix}
=
\begin{bmatrix}
a_1 & 0 & 0 & a_2 & a_3 & 0 \\
0 & a_2 & 0 & a_1 & 0 & a_3 \\
0 & 0 & a_3 & 0 & a_1 & a_2
\end{bmatrix}
\begin{bmatrix}
I_{11} \\
I_{22} \\
I_{33} \\
I_{12} \\
I_{13} \\
I_{23}
\end{bmatrix}
\]
Control Law & Update Law

Using Minimal Parametrization of the Inertia Matrix

\[ I_a^*(\sigma)\ddot{\sigma} + C_a^*(\sigma, \dot{\sigma})\dot{\sigma} = Y_a(\sigma, \dot{\sigma}, \ddot{\sigma})\theta \rightarrow \text{unknown parameters} \]

With the control law

\[ M = P_a^{-T}\{Y_a(\sigma, \dot{\sigma}, \dot{\sigma}_r, \ddot{\sigma}_r)\dot{\theta} - C_{da}\dot{e} - K_{da}\varepsilon\} \]

the closed loop dynamics take the form

\[ I_a\dddot{e} + \{C_{da} + C_a^*(\sigma, \dot{\sigma})\dot{e} + K_{da}\varepsilon = Y_a(\sigma, \dot{\sigma}, \ddot{\sigma})\dot{\theta} \]

and, along the adaptive law

\[ \dot{\theta} = -\Gamma Y_a(\sigma, \dot{\sigma}, \dot{\sigma}_r, \ddot{\sigma}_r)^T \dot{e} \]

guarantees asymptotic stability of the tracking errors
Adaptive–Reinforcement Learning Control
A-RLC Architecture
Numerical Example

- **Objective**
  - Demonstrate optimal shape morphing for multiple specified flight conditions

- **Method**
  - For every flight condition, learn optimal policy that commands voltage producing the optimal shape
  - Minimize total cost over the entire flight trajectory
  - Evaluate the learning performance after 200 learning episodes

**RL Module is Completely Ignorant of Optimality Relations and Morphing Control Functions:**
It Must Learn On Its Own, From Scratch
Smart Block Example

reinforcement learning definitions

- **Agent:** Smart Block Reinforcement Learning Module
- **Environment:** Various flight conditions
- **Goal:** Fly in optimal shape that minimizes cost
- **States:** Flight condition; shape of smart block $S = \{(y, F) \mid [2, 4] \times [0, 5]\} \cup \{(z, F) \mid [2, 4] \times [0, 5]\}$
- **Actions:** Discrete voltages applied to change shape of block
  - Action set: $A = \{(V_y, V_z) \mid [0, 0.5, 1, \ldots, 5] ; [0, 0.5, 1, \ldots, 4]\}$
- **Rewards:** Determined by cost functions
- **Optimal control policy:** Mapping of the state to the voltage leading to the optimal block shape
Smart Block Example

optimality relationship of flight condition and shape

- Cost function evaluated at steady-state condition $J = J_y + J_z$
  
  $$J = (y - S_y(F))^2 + (z - S_z(F))^2$$

where $S_y$ and $S_z$ are the optimal $y$ and $z$ dimensions at flight condition $F$:

$$S_y = 3 + \cos\left(\frac{\pi}{2} F\right)$$

$$S_z = 2 + 2e^{-0.5F}$$
Learning Process

Error in the Action Preference Function after:

- 20 Episodes
- 60 Episodes
- 100 Episodes
- 200 Episodes
Comparison of True Optimal Shape and Learned Shape

![Graphs showing comparison of true optimal shape and learned shape](image-url)
Time Histories of Angular States
Time Histories of Adaptive Parameters

<table>
<thead>
<tr>
<th>Inertia Vector</th>
<th>Estimated</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mass</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Valasek, Tandale, & Rong

AIAA-2004-6220-30
Trajectory Tracking Controls

Graphs showing force components and moment components over time.
Improvements

- **Function Approximation**
  - Vehicle shape is on continuous domains
  - Use **K-Nearest Neighbors Policy Iteration** method (KNNPI) to approximate the action-value function $Q(s,a)$

- **Optimal action-values of samples**
  - **Q-learning**: off-policy temporal difference method
  - Collect a set of state-action pair samples
  - The action-value of a new state-action pair is the interpolation of those of its K nearest neighbors

- **Exploration-exploitation dilemma**:
  - **ε-policy with decreasing ε**
  - Explorative early, exploitative later

```latex
\varepsilon \text{- greedy policy}
\begin{align*}
\text{if} \ (\text{probability} > 1 - \varepsilon) \quad & a = \arg \max_{a} Q(s,a) \\
\text{else} \quad & a = \text{rand}(a_i)
\end{align*}
```
Morphing Air Vehicle

TiiMY
Q-Learning

- Action value function $Q(s,a)$
  - How well agent performs action $a$ in state $s$ under policy $p_i$
- Q learning is an off-policy temporal difference method

Q-Learning()

Initialize $Q(s,a)$ arbitrarily

Repeat (for each episode):
  
  Initialize $s$

  Repeat (for each step of episode):
    
    Choose $a$ from $s$ using policy derived from $Q(s,a)$
    
    // (e.g., $\varepsilon$-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 shape of the vehicle is on continuous domains

Use K-nearest neighbors method to approximate the action-value function $Q(s,a)$

- Collect a set of state-action pair samples
- Compute optimal action-values of these samples using Q-learning
- The action-value of a new state-action pair is the interpolation of those of its K nearest neighbors.

The whole policy iteration process is called KNN Policy Iteration (KNNPI)

\[
Q(s_0, a_0) = \sum_{n=1}^{K} \frac{Q_{\text{sample}}(s_i, a_i)}{\text{Distance}(s_i, a_i, s_0, a_0)}
\]
Demo 2
Conclusions

- Shape Changes for Mission Morphing can be treated as piecewise constant parameter changes
  - SAMI is a favorable method for trajectory tracking control

- Reinforcement Learning successfully learns the optimal control policy using the $\epsilon$-greedy policy
  - Takes care of the exploration exploitation dilemma

- Morphing dynamics (stiffness, frequency, damping ratio) and cost function have a major effect on learning times and learning performance.
  - Slowest element in system is critical

- Flight condition transitions, morphing dynamics, learning performance, and adaptive control must be balanced to achieve synergy and therefore good performance.
  - Coordination and timing is everything
Future Research 1

- **Directly learn** voltage inputs required to achieve certain position states with time dependency.
  - Skips mathematical modeling and computer simulation steps
  - Avoids modeling and simulation errors
Future Research 2

- Modify the morphing dynamics to represent SMA actuators.
  - Hysteretic behavior

- Investigate control methodologies to handle faster shape changes
  - Linear Parameter Varying (LPV) control

- Modify the Reinforcement Learning module to incorporate continuous states and actions.
  - Function approximators such as Radial Basis Functions

- Modify the simulation to include a more advanced aircraft model
  - Wing-Body, Wing-Body-Empennage, etc.
Questions?
Reinforcement Learning

- **Actor-critic method**
  - On policy, TD method
  - The actor selects actions
    - **Preference of an action:** \( p(s, a) \)
    - **Greedy Policy:**
      \[
      \pi_t(s, a) = \arg\max_a p(s, a)
      \]
    - **Gibbs softmax policy:**
      \[
      \pi_t(s, a) = \Pr\{a_t = a \mid s_t = s\} = \frac{e^{p(s, a)}}{\sum_a e^{p(s, a)}}
      \]
  - The critic criticize the actions
    - **State value functions:** \( V(s_t) \)
    - **TD error:**
      \[
      \delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)
      \]
    - Strengthen or weaken the tendency to select one action
      \[
      p(s_t, a_t) \leftarrow p(s_t, a_t) + \beta \delta_t
      \]
Reinforcement Learning

function approximation

Parameterized functional form

\[
V_{t}^{\pi}(s) = \sum_{j=1}^{N} \theta_{v_{tj}} \Phi_j(s) \quad p_{at}(s) = \sum_{j=1}^{N} \theta_{p_{atj}} \Phi_j(s)
\]

Gradient descent learning

\[
\theta_{v_{t}} = \theta_{v_{t-1}} + \alpha(\bar{y} - H_{t} \theta_{v_{t-1}})H_{t}^{T} = \theta_{v_{t-1}} + \alpha \delta_{t} H_{t}^{T}
\]

The **Temporal Displacement** error \( \delta_{t} \) drives all learning in both actor and critic

\[
\theta_{p_{at}} = \theta_{p_{at-1}} + \alpha(p_{at}(s_{t}) + \beta \delta_{t} - H_{t} \theta_{p_{at-1}})H_{t}^{T} = \theta_{p_{at-1}} + \alpha \beta \delta_{t} H_{t}^{T}
\]