

# Journal of Intelligent Material Systems and Structures

<http://jim.sagepub.com/>

---

## Active Length Control of Shape Memory Alloy Wires Using Reinforcement Learning

Kenton Kirkpatrick and John Valasek

*Journal of Intelligent Material Systems and Structures* published online 22 August 2011

DOI: 10.1177/1045389X11411117

The online version of this article can be found at:

<http://jim.sagepub.com/content/early/2011/08/11/1045389X11411117>

---

Published by:



<http://www.sagepublications.com>

Additional services and information for *Journal of Intelligent Material Systems and Structures* can be found at:

**Email Alerts:** <http://jim.sagepub.com/cgi/alerts>

**Subscriptions:** <http://jim.sagepub.com/subscriptions>

**Reprints:** <http://www.sagepub.com/journalsReprints.nav>

**Permissions:** <http://www.sagepub.com/journalsPermissions.nav>

# Active Length Control of Shape Memory Alloy Wires Using Reinforcement Learning

KENTON KIRKPATRICK AND JOHN VALASEK\*

*Aerospace Engineering Department, Texas A&M University, College station, TX, USA*

**ABSTRACT:** Actively controlled shape memory alloy actuators are useful for a variety of applications that require accurate shape control. For shape memory alloy wires, strain is modulated with temperature, usually by an applied voltage difference across the length. Numerical simulation using reinforcement learning has previously been used for determining the temperature–strain relationship of a shape memory alloy wire and for synthesizing a limited control policy that relates applied temperature to desired strain. However, learning the voltage–strain relationship is of more practical interest in synthesizing feedback control laws for shape memory alloy wires since the control input in practical applications will be an applied voltage that modulates temperature. This article implements a Sarsa-based algorithm for determining a feedback control law in voltage–strain space and validates it experimentally. Experimental results presented in this article demonstrate the ability to control a shape memory alloy specimen from arbitrary initial strains ranging from zero to maximum, including intermediate strains, to an arbitrary intermediate strain. The results also demonstrate the ability to control the specimen from similar arbitrary initial values of strain to zero strain. The voltage–strain learning algorithm developed in this article is a promising candidate for synthesizing practical shape memory alloy actuator feedback control laws.

*Key Words:* shape memory alloys, reinforcement learning, feedback control, Sarsa.

## INTRODUCTION

SHAPE memory alloys (SMAs) have been under considerable investigation as a candidate smart material for aerospace applications. Two of these applications are structural shape-changing for a morphing aircraft (Kudva, 2004; Barbarino et al., 2009) and control of fluid-structure interactions or aeroelastic effects (Strelec et al., 2003; Bae et al., 2005). The shape memory effect (Waram, 1993) makes SMAs ideal for use in structures that undergo large amounts of strain (Mavroidis et al., 1999). At lower temperatures, SMAs begin in a crystalline structure of martensite and undergo a phase change to austenite as the alloy is heated. Due to asymmetry in these phase transformations, an SMA wire exhibits a hysteresis behavior in its temperature and strain relationship (Mavroidis et al., 1999). This hysteresis occurs because the phase transformation from martensite to austenite begins and ends at different temperatures than the reverse process.

Control of this transformation is critical for successful actuation. Heating *via* electrical resistance is commonly used to affect temperature changes in SMAs, and

modulating this heating is necessary for purposes of control. For an SMA wire, the rate at which temperature changes is a function of the physical properties of the wire, the rate at which heat is lost to the environment, and the rate at which the wire heats due to electric current. The relationship between temperature and applied voltage during the crystal phase transformation can be determined through the following differential equation (Marchado, 2007):

$$\rho C \frac{\partial T}{\partial t} = \frac{V^2}{R} - hA(T - T_\infty) + k \frac{\partial^2 T}{\partial x^2} - \dot{\sigma} \alpha T + (\pi - \sigma \Delta \alpha T - \rho \Delta s_0 T) \dot{\xi} \quad (1)$$

In Equation (1),  $\rho$  is the wire density,  $C$  the specific heat of the wire at constant volume,  $T$  the wire temperature,  $V$  the voltage difference in the wire,  $t$  the time,  $R$  the wire electrical resistance,  $h$  the convective heat transfer coefficient,  $A$  the wire surface area,  $T_\infty$  the ambient temperature of the coolant surrounding the wire,  $k$  the thermal conductivity, and  $x$  the direction along the wire length. The phase transformation causes extra terms to appear as a function of the properties during transformation. In this additional part of the heat transfer equation,  $\sigma$  is the stress,  $\alpha$  the thermal diffusivity,  $\pi$  the thermodynamic driving force for transformation,  $s_0$

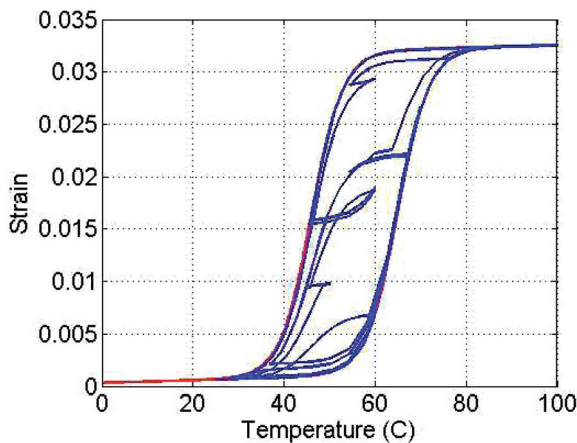
\*Author to whom correspondence should be addressed.  
Email: valasek@tamu.edu  
Figures 1–4 and 7–12 appear in color online: <http://jim.sagepub.com>

the specific entropy at the reference state, and  $\xi$  the martensitic volume fraction. The parameters  $\alpha$ ,  $\rho$ , and  $C$  are typically assumed to be the same in each phase, but  $s_0$  and  $\pi$  are variable between the phases. The thermodynamic driving force is a function of both the martensitic volume fraction and the Gibbs free energy.

When temperature and its time derivative are known, Equation (1) can be used to determine the required voltage. However, it is not readily suitable for use in SMA control policy learning because it requires tracking several terms that are not easily measured or derived. *Thus, it is a more straightforward process to learn the policy for voltage–strain directly, rather than try to convert between temperature and voltage.*

Considering a state space consisting of temperature–strain variables, the hysteresis behavior of SMAs is most often characterized with constitutive models based on material parameters (Lagoudas et al., 2001). This can be a time and labor intensive process, and it does not characterize the hysteresis in real-time. Other methods that characterize hysteresis behavior are phenomenological models (Lagoudas et al., 1996; Bo and Lagoudas, 1999), micromechanical models (Patoe et al., 1987; Falk, 1989), and empirical models based on system identification (Banks et al., 1997; Webbet et al., 1998). These models are quite accurate, but some only work for particular types of SMAs, and most require complex computations. Many of them are also unable to be used in dynamic loading conditions, making them unsuitable for morphing control applications. Additionally, none of these methods directly characterize the minor hysteresis loops that correspond to an SMA that is not fully actuated (Figure 1).

Learning a control policy capable of achieving a strain that rests within the interior of the transformation curve is important because it can greatly increase the range, and therefore functionality, of SMA actuators. If the only values that can be controlled correspond to maximum and minimum strains, an SMA actuator would be limited



**Figure 1.** Typical SMA major and minor hysteresis loops (Kirkpatrick and Valasek, 2009).

to only two possible positions. Learning these interior goals is more complicated than learning the extreme values because all that would be required for the control policy would be simply to apply the maximum and minimum voltages every time. A Machine Learning algorithm introduced in the section called ‘Reinforcement Learning’ (RL) is used to determine this hysteresis mapping in temperature–strain space, and in real-time (Kirkpatrick and Valasek, 2009). RL uses a process of earning rewards and incurring penalties in order to build a memory of which actions (control inputs) are best at achieving a goal (desired strain), and which are poor. Since it learns how to control an SMA to intermediate strains and is applicable to both simulation and experimentation-based research, the RL-based approach is useful for the morphing actuation and control problem (Haag et al., 2005; Valasek et al., 2005, 2008). Additionally, when used in an experimental setting with SMA specimens, it determines both the major and minor hysteresis loops in addition to learning the control policy. However, this method learns the temperature–strain relationship and control policy, not the voltage–strain relationship and control policy which are much more useful for applications.

This article develops an RL-based method for determining the voltage–strain relationship and control policy for SMA length control and validates it with an SMA specimen using an experimental apparatus. This article is organized as follows. The second section introduces the basics of RL and extends it to this specific research problem. Details of the Sarsa method, the  $\epsilon$ -Greedy approach, and function approximations are also presented. The third section explains how the experimental apparatus is built for verifying the approach and includes a discussion of cooling fluid selection. The fourth section shows how the RL learning agent is interfaced and connected to the experimental apparatus for real-time learning. The fifth section provides a detailed explanation of the characterization of SMA hysteresis behavior in temperature–strain space. Both simulation and experimentation results are discussed. The sixth section presents the voltage–strain learning results and then demonstrates how to control an SMA wire’s thermally induced crystal phase transformation. Conclusions are presented in the final section.

## REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a process of learning through interaction with an environment, in which a software agent (algorithm program) uses previous knowledge of the results of its actions in a particular situation to make an informed decision when it later returns to the same situation. It is a method that has been used for many diverse situations ranging from board games to behavior-based robotics (Sutton and Barto, 1998; Konidaris and Hayes, 2005;

Varshavskaya et al., 2008). RL methods learn an action-value function that provides a mapping from states to actions. The action-value function can be used as a control policy to determine the proper control inputs for a given state. For use in applications, this control policy is usually approximated with a large matrix that is composed of every possible state for the rows, and every possible action for the columns. In this study, a third dimension is included in the control policy that is composed of every possible goal state.

The three most commonly used classes of algorithms for RL are Dynamic Programming, Monte Carlo, and Temporal Difference (Sutton and Barto, 1998). The majority of Dynamic Programming methods require an accurate environmental model, often making them impractical in problems with complex models. Monte Carlo only allows learning to occur at the end of each episode, causing problems that have long episodes to have a slow learning rate. Temporal Difference methods have the advantage of being able to learn at every time step without requiring the input of an environmental model. Here, a method of Temporal Difference called *Sarsa* is used. *Sarsa* is an on-policy form of Temporal Difference, meaning that at every time interval the control policy is used to select actions to be taken, and the control policy is also evaluated and improved as learning occurs. *Sarsa* updates the control policy using the current state, current action, future reward, future state, and future action to dictate the transition from one state/action pair to the next (Sutton and Barto, 1998). The action value function used to update this control policy is:

$$Q_k(s, a) = Q_k(s, a) + \alpha \delta_k \quad (2)$$

where  $s$  is the current state,  $a$  the current action,  $Q$  the control policy, and the  $k$  subscript signifies the current policy. The  $\alpha$  term is a parameter that is used to ‘penalize’ the RL algorithm when it repeats itself within each episode. The term  $\delta_k$  is defined as

$$\delta_k = r_{k+1}(s', a') + \gamma Q_{k+1}(s', a') - Q_k(s, a) \quad (3)$$

where the term  $s'$  refers to the future state,  $a'$  the future action,  $k+1$  the future policy, and  $\gamma$  a constant that is used to optimize the rate of convergence by weighting the future policy. Equations (2) and (3) can be combined to form the detailed action-value function:

$$Q_k(s, a) = Q_k(s, a) + \alpha [r_{k+1}(s', a') + \gamma Q_{k+1}(s', a') - Q_k(s, a)] \quad (4)$$

The reward given for each state/action pair is defined by  $r$ , which is defined by the user for each situation. Here, a reward of 1 is assigned when a goal state is achieved, while a reward of 0 is assigned for achieving

any other state within range. If the boundaries of the problem are exceeded, a reward of -1 is assigned to discourage the agent from taking that action again. Likewise, the learning rate,  $\alpha$ , and the discount factor,  $\gamma$ , are user defined. A constant discount factor of  $\gamma = 0.8$  is used to generate the results presented here, while the learning rate is defined to be a function of the number of visits to a particular state. With the variable  $n_s$  representing the number of times the current state is visited during an episode, the learning rate for that state is determined by:

$$\alpha = \frac{1}{n_s} \quad (5)$$

For properly learning the voltage–strain relationship, the control policy is modified to include the goal (strain) as a third dimension. This permits the control policy to be represented as a set of tables that can be used to look up the correct voltage values to use when the current state and goal state are known. With  $g$  representing the goal state, the modified action-value function now becomes:

$$Q_k(s, a, g) = Q_k(s, a, g) + \alpha [r_{k+1}(s', a', g) + \gamma Q_{k+1}(s', a', g) - Q_k(s, a, g)] \quad (6)$$

This action-value function generates the policy that is used to learn the parameters of the system being explored. This notation is slightly different than the classic action-value function because of the inclusion of the goal state as an extra dimension to the function. However, this is a notational difference only. Separate value functions must be learned for each desired goal state, so including  $g$  in the function simply indicates that this is the case, and thus does not affect the convergence of the learner.

Next, to update the policy using the action-value function provided in Equation (6), the *Sarsa* algorithm is used (Sutton and Barto, 1998):

#### *Sarsa Algorithm:*

- Initialize  $Q(s, a, g)$  arbitrarily
- Repeat for each episode:
  - Initialize  $s$
  - Choose  $a$  from  $s$  using policy derived from  $Q(s, a, g)$  (e.g.,  $\epsilon$ -Greedy)
  - Repeat for each time step:
    - \* Take action  $a$ , observe  $r, s'$ ,
    - \* Choose  $a'$  from  $s'$  using policy derived from  $Q(s, a, g)$  (e.g.,  $\epsilon$ -Greedy)
    - \*  $Q(s, a, g) \leftarrow Q(s, a, g) + \alpha [r + \gamma Q(s', a', g) - Q(s, a, g)]$
    - \*  $s \leftarrow s', a \leftarrow a'$
  - Until  $s$  is terminal

The dilemma of selecting the best action from  $Q$  lies in the fact that the policy does not have any information about the system in the beginning, and must therefore explore so as to learn from interactions with the environment. Sarsa is able to learn the system behavior when the algorithm has no prior knowledge. However, it must explore in the early episodes since it cannot exploit previous knowledge. In future episodes, exploitation of knowledge becomes more favorable so that actions leading to the goal are reinforced. Balancing exploration and exploitation is key to convergence of the RL agent and therefore to obtaining the best control policy. The  $\epsilon$ -Greedy method of choosing an action is used here, which means that for some percentage of the time the RL agent will choose to randomly select actions, i.e. explore, rather than choose an action that the action-value function believes will yield the highest reward (Whiteson et al., 2007). This is because the RL agent might not have already explored every possible action, and a better action may exist than the one that is presently thought to yield the greatest reward. The variable  $\epsilon$  is a number between 0 and 1 that determines the percent chance that exploration will be used instead of exploitation. By definition, a fully greedy method chooses only the optimal path without ever choosing to explore new paths. This corresponds to an  $\epsilon$ -Greedy method where  $\epsilon = 0$ . The  $\epsilon$ -Greedy action-value method is implemented with the following algorithm:

*$\epsilon$ -Greedy Action-Value Method:*

- Choose  $\epsilon$  between 0 and 1
- Repeat for each action selection:
  - Generate random value  $\beta$  between 0 and 1
  - If  $\beta \geq 1 - \epsilon$ 
    - \*  $a \leftarrow \text{random}$
  - If  $\beta < 1 - \epsilon$ 
    - \*  $a \leftarrow Q(s,a,g)$  exploitation

To converge to the best control policy in the shortest time, an episodically annealing  $\epsilon$ -Greedy method is used to alter the exploration constant,  $\epsilon$ , depending upon the current episode. In the first episodes, little to no information has been learned by the policy, so a greater degree of exploration is required. Conversely, in future episodes less exploration is desired so that the RL agent can exploit the knowledge that it has learned. To achieve an annealing  $\epsilon$ -Greedy method, a simple algorithm is constructed that determines what value should be used for  $\epsilon$  at each individual episode. The exploration probability ranges from 70% ( $\epsilon = 0.7$ ) in the first several episodes to 5% ( $\epsilon = 0.05$ ) in the final episodes. Even during later episodes, the algorithm still never exhibits a fully greedy method of choosing actions. A small chance of performing exploratory actions is still allowed because it forces the agent to

check for better paths in case the path it converged upon is not actually the best choice.

The objective of the RL agent is to converge upon the optimal voltage needed to produce the desired strain based upon the current strain in the SMA wire. The states are defined by the current axial strain, and the actions are defined by the desired voltage that is applied to the SMA wire. Conversion between strain and position is straightforward, so strain is used as the state because it can be global to specimens of different lengths. Thus, the specified goal is a desired strain of the SMA wire.

When RL is applied to a system with a continuous state space, the state space is not entirely represented. Since SMA wire length is continuous, the policy must be approximated for a continuous system (Santa Maria et al., 1997). The use of function approximation allows the agent to explore a state space that has fewer discrete values, resulting in shorter learning times. The  $k$ -Nearest Neighbor method, used here, is an instance-based machine learning algorithm that learns an approximation of a target function by means of assigning values to attributes associated with the  $k$ -nearest points in Euclidean distance to the target instance (Mitchell, 1997). The Euclidean distance is the geometric distance between instances in  $n$ -dimensional space, where  $n$  is the total number of attributes. It is assumed that the properties of a point in the state-space are likely to be very similar to the properties of the points that have similar attribute values (Russel and Norvig, 2003). In this study, the goal state is 1-dimensional in both formulations of the problem that are explored, so a 1-Nearest Neighbor is implemented. Once the RL agent learns the most appropriate voltage required to achieve each goal strain from each initial strain, it can be used thereafter to control the length of an SMA wire in real-time.

## EXPERIMENTAL APPARATUS

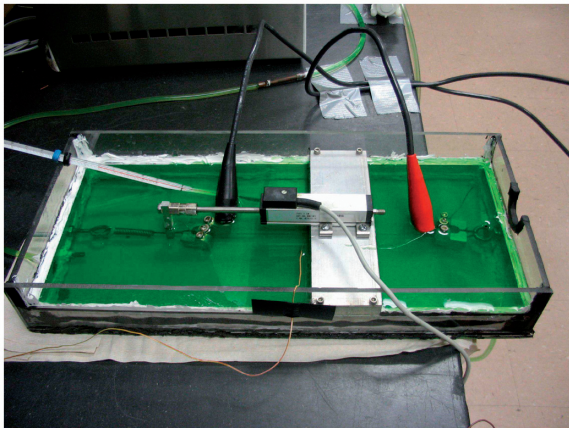
The SMA wire is mounted in an apparatus that is constructed of Plexiglas and aluminum supports. The apparatus is sealed so that no fluid can leak out during the experiment. The SMA wire is attached to the walls by Kevlar cords for strength and insulation and is set in series with a free-weight that is attached by Kevlar over a dual pulley system. The mass of the free-weight changes depending on the diameter of the wire being tested and is selected so that the wire experiences a stress of approximately 120 MPa in its initial martensitic state at zero voltage and room temperature. A Linear Position Transducer (LPT) is supported above the fluid by an aluminum beam, and the probe end is connected to the Kevlar cords for position measurement without receiving current from the SMA wire. The LPT sends a voltage to the Data Acquisition (DAQ) board, which changes depending on the position of the probe.



The DAQ board has a sampling rate of 250 kS/s, and therefore does not limit the RL agent convergence time since the agent chooses new actions based on one sample every 15 s. A variable voltage supply is used to provide a voltage difference across the wire for resistive heating and is connected to the SMA wire *via* alligator clips that are positioned carefully along the wire so that every specimen tested maintains the same effective initial length. The voltage supply receives its commands from the DAQ board with an input/output voltage ratio of 3.6 and outputs voltages in the range of 0.00–2.80 V. For the state inputs to the RL agent, only strain and voltage are needed, but since the actuation of the SMA wire is temperature-based, the experimental setup includes temperature measurements for the sake of reference to make sure that the wire follows the hysteresis path correctly. A thermocouple is attached to the SMA wire for temperature measurements and sends small voltages to the DAQ board that are converted to temperature measurements. Figure 2 shows the complete experimental apparatus.

The apparatus contains a pool of antifreeze that completely submerges the SMA wire and the alligator clips to allow sufficient cooling of the wire for prevention of overheating and to decrease the time required for the reverse phase transformation from austenite to martensite. The antifreeze is drawn out of the apparatus by a pump that sends it into a pool for temperature regulation. The external pool contains both heating and cooling coils that allow it to keep the antifreeze at a specified ambient temperature. In this experiment the ambient temperature is kept at  $21 \pm 2$  °C. The cooled antifreeze is then drawn back out of the temperature regulation pool by another pump and is sent into the apparatus to continue fluid circulation and keep the coolant at a constant room temperature. Figure 3 shows the full setup of the experiment.

Antifreeze is used as the coolant because it cools the SMA wire faster than air and is less conductive than



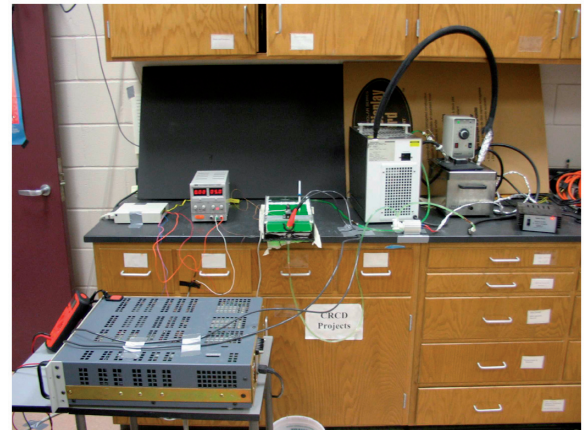
**Figure 2.** Experimental apparatus containing a pool of antifreeze (Kirkpatrick and Valasek, 2009).

water. By using antifreeze full actuation of the SMA specimen can occur with 2.8 V instead of the 12 V required with water. Figure 4 shows an example of using this apparatus to characterize the major hysteresis behavior. The input values were not determined by the RL algorithm, but manually selected over a range of temperature inputs. It should be noted that a bias exists in the temperature measurements as a result of the thermocouple attachment method. Because attaching the thermocouple rigidly to the SMA wire could change the mechanical response, it is instead tied to the specimen by Kevlar string. As a result, the temperature measurements do not precisely reflect the temperature of the specimen, but rather the temperature of the fluid–surface interface.

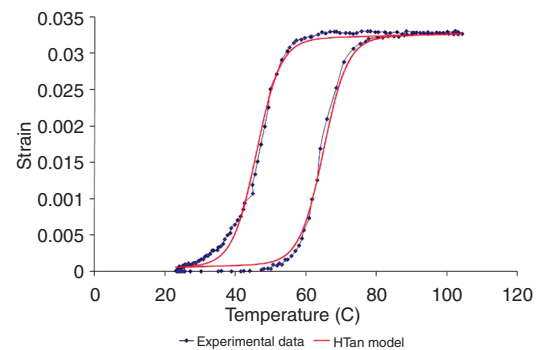
The solid line in Figure 4 is determined with a mathematical model based on a hyperbolic tangent curve represented as double-exponentials:

$$M_l = H/2 \tanh((T - ct_l)a) + s(T - (ct_l + ct_r)/2) + H/2 + c_s \quad (7)$$

$$M_r = H/2 \tanh((T - ct_r)a) + s(T - (ct_l + ct_r)/2) + H/2 + c_s \quad (8)$$



**Figure 3.** Experimental setup (Kirkpatrick and Valasek, 2009).



**Figure 4.** NiTi major hysteresis in antifreeze-filled apparatus (Kirkpatrick and Valasek, 2009).

In Equations (7) and (8),  $H$ ,  $ct_r$ ,  $a$ ,  $s$ ,  $ct_l$ , and  $c_s$  are constants that determine the shape of the hyperbolic tangent model.  $M_r$  and  $M_l$  are the strain values that correspond to the temperature input into the equations. The constants were selected by creating a curve that best fits the known hysteresis behavior for the particular SMA wire specimen used in this experiment. The constants in this model are purely numerical and are not derived from constitutive models. However, the model is conceptually based on similar exponential-based constitutive models such as the Tanaka Model (Lagoudas, 2008).

### HARDWARE/SOFTWARE INTERFACE

The RL agent is implemented as a MATLAB script which communicates with the experimental apparatus *via* a LabVIEW program. This LabVIEW program uses graphical functions to create an application capable of communicating with external hardware. First, the DAQ board relays the input voltages from the thermocouple and the LPT to the computer via a DAQ card installed in the computer. Then, the LabVIEW program takes these voltages and converts them into the present temperature and strain readings. These inputs are sent to MATLAB for use by the RL agent, and MATLAB passes to LabVIEW the magnitude of the voltage that needs to be applied to the wire in order for the desired strain to be reached. LabVIEW transfers this voltage to the DAQ board, which sends the signal to the variable voltage supply, telling it to output the required voltage to the SMA wire. In this manner, the RL agent is able to learn the required control policy of an SMA wire specimen in an experimental fashion (Figure 5).

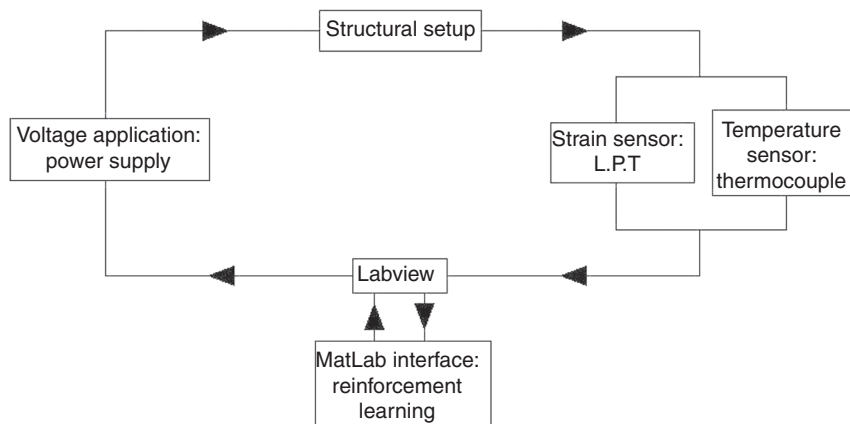
### TEMPERATURE–STRAIN LEARNING RESULTS

The main benefit of using an RL method like Sarsa is that learning a control policy does not require a model of the system. In a numerical simulation, a model is

required for the purposes of simulating the environment. To demonstrate that this method works for SMA control learning without requiring a model, the same temperature–strain learning experiment that was previously done in simulation is repeated in an experimental setting. The experimental apparatus and software previously discussed were used to this end.

This experiment has been tested in temperature–strain space over many episodes at several different goal states corresponding to individual strain states, where an episode is defined as the achievement of a goal. The state space for this experiment consists of the strain and temperature at the current time step, and the action-space consists of desired temperature. The LPT allows for measuring the full range of motion of the wire changes, with a strain resolution of better than 0.0001 mm/mm. The discretization used for strain was chosen to be from 0 to 0.035 in steps of 0.001. The thermocouple used to measure the temperature allowed for temperature measurements accurate to 1°C. The temperature ranges were chosen to be broken into bins of 10°C increments from 25°C to 125°C, plus one bin of <25°C, making 11 discrete temperature bins. One extra slot is added to the state space representing going out of bounds. Since the dimension of  $Q$  is  $(s \times a \times g)$ , the dimension of  $Q$  comes out to be  $(386 \times 11 \times 35)$ .

With the current configuration, 3.3% strain is the maximum strain possible that corresponds to complete actuation. To demonstrate the convergence of the RL agent, a goal state of 2.7% is investigated in detail. This goal was chosen because it requires nearly complete actuation of the SMA wire, but does not reach a fully actuated state. This forces the RL program to find the correct temperature state exactly. When the maximum goal state of 3.3% is chosen, the state is achieved more easily since any temperature exceeding the austenite finish temperature will yield a fully actuated strain state. This makes observing an intermediate strain state much more useful for analyzing the success of the learning agent.

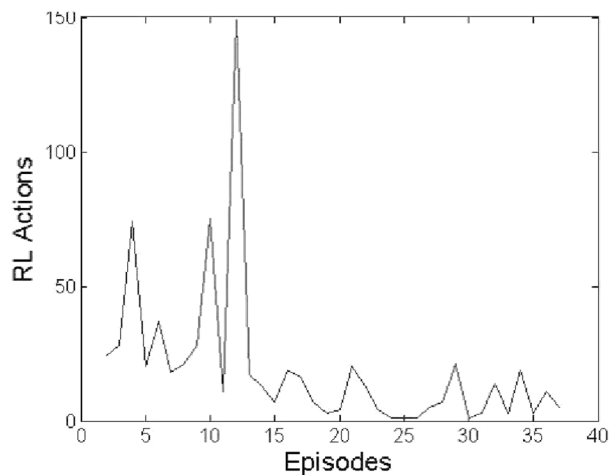


**Figure 5.** Hardware and software interfaces of the experimental apparatus Kirkpatrick and Valasek, 2009.

Figure 6 shows the relationship between the episodes completed and the total Reinforcement Learning actions attempted for reaching a goal of 2.7% strain. Every episode presented in this data begins at a fully unactuated strain of 0%. As this graph shows, the RL algorithm takes fewer actions to achieve the desired goal state as it experiences more episodes. This suggests that the RL agent becomes more successful in completing its objective of finding the optimal temperature required to achieve this goal state as it continues to learn.

The y-axis in Figure 6 refers to the number of times a new action is chosen before the goal state is reached. A new action is set to be chosen every 15 s so that the SMA wire has time to settle out when it is an action that requires cooling. During that 15 s delay, the previously selected action is considered a step command. Figure 6 reveals that the control policy begins learning enough about the system to obtain the desired strain with only a few actions by the time it has reached 20–25 episodes. However, it can also be seen that even after this point, there are a few episodes that required a larger number of actions to find the goal. This happens for two main reasons. Since the RL agent being used incorporates the logic of the  $\epsilon$ -Greedy method, even after the algorithm begins converging on the optimal policy, exploration is still encouraged to allow the system to find a better path to goal state achievement. The other reason that the agent still does not exhibit perfect control is because the measurements of the thermocouple are inaccurate during the intermediate phase changes and can sometimes be off by as much as  $10^{\circ}\text{C}$ . This can cause problems with the learning process that require many more episodes to achieve an optimal policy.

Over the course of 37 episodes to a goal state of 2.7% strain and back to a goal state of 0% strain, the major hysteresis behavior becomes visible. The progression of the control policy's ability to obtain the hysteresis behavior is also of interest from this experiment. This



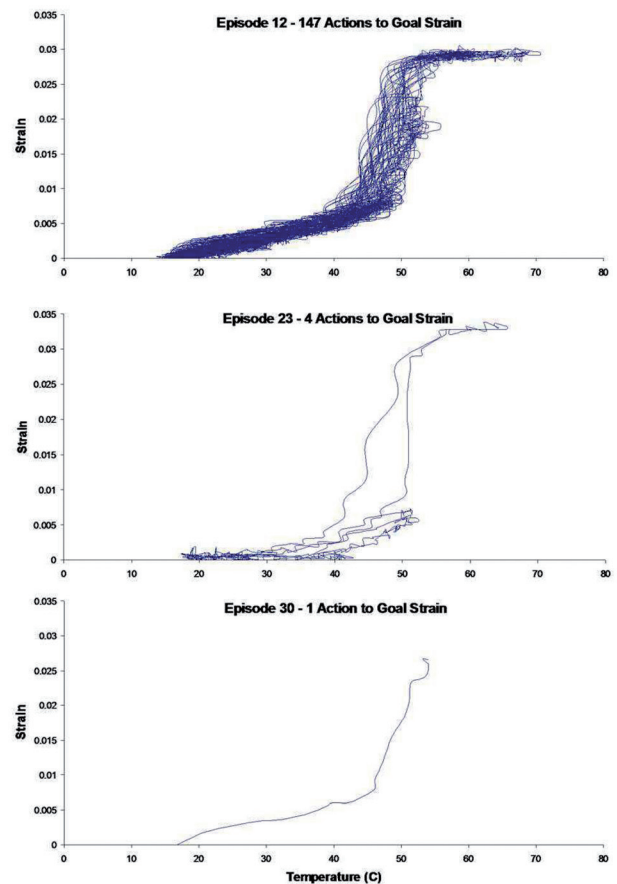
**Figure 6.** Episode number vs actions required to find goal in temperature–strain space (Kirkpatrick and Valasek, 2009).

information shows how well the experiment was able to utilize the learning capabilities of an RL algorithm. Figure 7 shows the paths that are taken to obtain the final goal state for three different episodes.

During episode 12 (Figure 7(a)), the experimental system required 147 actions to achieve the goal strain of 2.7%. As a result, the system wandered between many different temperatures before it was finally able to find the temperature that would yield the correct goal state. After running more similar episodes, the control policy learned how to achieve the goal state while taking fewer actions. By episode 23 (Figure 7(b)), only four actions were required to achieve the goal of 2.7% strain. Episode 30 (Figure 7(c)) demonstrates the control policy's ability to find the correct goal state in only one action.

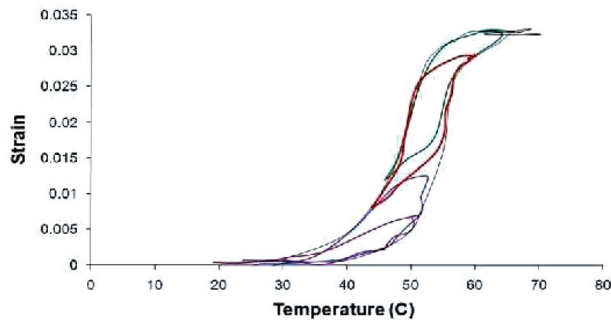
The RL agent's ability to find a control policy that learns the minor hysteresis behavior of a shape memory alloy was of special interest because minor hysteresis loops are difficult to obtain by other methods. By using RL to characterize the hysteresis behavior, the minor loops are obtained just as easily as the major loops. The minor hysteresis behavior can be extracted from individual episodes, as demonstrated in Figure 8.

Figure 8 represents the extraction of the major hysteresis loop and three minor hysteresis loops from



**Figure 7.** Path Progression (Kirkpatrick and Valasek, 2009).





**Figure 8.** Minor hysteresis loops from learning episodes.

the data obtained during episode 12 of the 2.7% goal experimentation. Normally, these minor loops must be obtained using mathematical models based on the major hysteresis behavior, but this result shows that the minor hysteresis loops can be experimentally obtained through the RL method. The real-time data collection as the RL algorithm experimentally determines how to achieve each goal state allows both major and minor hysteresis loops to be mapped precisely. This is of particular importance for extension to voltage–strain space control because it shows that the control policy learned by RL can achieve a goal state starting from any initial state, not just from the fully un-actuated or actuated states.

## VOLTAGE–STRAIN LEARNING RESULTS

The control policy developed for this SMA specimen provides the ability to control the length of a NiTi SMA wire for two specific goal strains within an error range of  $\pm 0.005$  strain. For this experiment, the size of the state space is decreased to only include the current strain, and the action-space is altered to choose a commanded voltage directly. This is more useful from a feedback control perspective because applications usually will use an applied voltage as the control input to effect a temperature change. The LPT and thermocouple used were the same as in the previous experiment, but the state space only consists of current strain, so the size of the state space discretization is 36 with one added for out-of-bounds. The variable voltage supply is capable of achieving accurate voltage outputs of less than 0.01V, so the voltage action discretization is chosen to be 21 equal bins from 0.00–2.80 V. This makes the dimension of the  $Q$ -matrix for this experiment ( $36 \times 21 \times 35$ ).

The wire specimen used for this experiment had an initial effective length of 13 cm, so with a maximum possible strain of 3.3%, the total operating range of motion is 4.29 mm. Since the learned control policy is able to reach its goal within a range of  $\pm 0.5\%$ , the error range allowed is  $\pm 0.65$  mm. Under these specified conditions, the RL agent is executed for 100 episodes using specified alternating goal strains of 2.7% and 0.1%, providing 50 episodes per goal. Each episode in this experiment

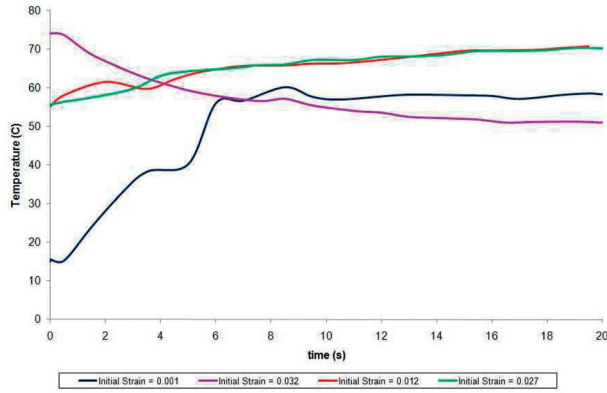
consists of 450 s worth of seeking a single goal, where the RL agent is called every 15 s. This provides 30 new actions per episode for the learning module.

Two test cases are presented. Case 1 uses a goal of 2.7% strain. This goal is chosen for experimentation because it represents a partially actuated state for which the maximum strain of 3.3% falls outside of the allowed tolerance range of  $\pm 0.5\%$ . This ensures that it cannot achieve the goal by simply applying the maximum voltage available. This goal is also of particular interest since it was previously used for temperature–strain space validation. Case 2 uses a goal strain of 0.1%. This goal is chosen because it represents a state that is not quite on the boundary of the system, but effectively is on the boundary because the lower bound is encompassed by the tolerance range. While it could achieve its goal by applying 0 V, it is not limited to this action.

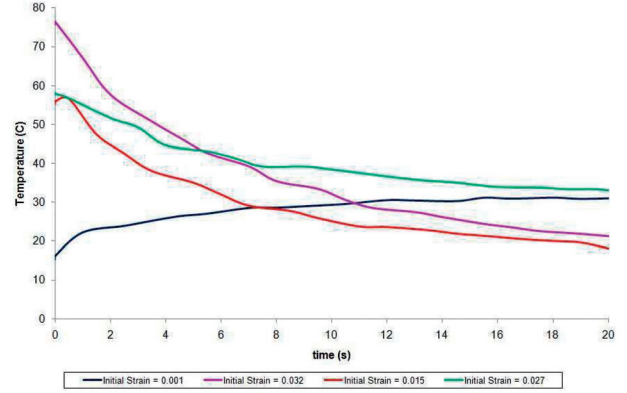
The temperature and strain time histories for Case 1 are shown in Figures 9 and 10. Figure 9 shows that the required final temperatures for reaching the goal differ depending upon the initial temperature and strain conditions. This is due to the hysteretic nature of the temperature–strain relationship. Figure 10 demonstrates that the learned control policy is capable of bringing an SMA wire specimen to the desired goal from various initial strains. Initial voltages were applied before the time history recording began so that the control policy could be tested at several different initial strains. The initial strains are 0.1%, 3.2%, 1.2%, and 2.7%. The actual exploitation of the control policy began at time = 0 s in each case, and the two horizontal lines represent the goal range of  $2.7 \pm 0.5\%$  strain. The initial strains of 0.1% and 3.2% are chosen so that the control policy could be tested from initial strains corresponding to fully un-actuated and fully actuated states, respectively. The initial strain of 1.2% is selected to test from an initially intermediate strain, and the goal strain of 2.7% is also chosen as an initial strain to show that the agent can learn how to stay within the specified range when the specimen is there initially. As Figure 10 shows, the control policy is successful in achieving its goal of  $2.7 \pm 0.5\%$  in all four test cases. The voltages that the learned action-value function commanded for each of these initial conditions are displayed in Table 1. These values are the commanded voltages for changing the SMA wire from each of these initial strains to the goal strain of 2.7% in 15 s.

The results for Case 1 demonstrate that this RL agent can learn a voltage–strain SMA control policy for achieving multiple commanded intermediate strain changes.

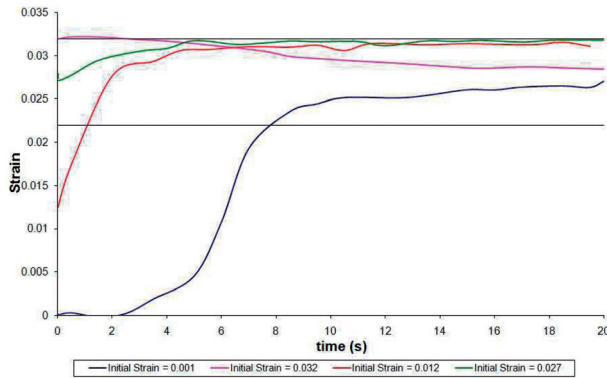
The results for Case 2 are presented in Figures 11 and 12. Like Case 1, initial voltages were applied to reach initial strain conditions, and the control policy exploitation begins at time = 0 s. Figure 11 again demonstrates that the final temperature values for reaching the strain goal vary based on initial conditions. In Figure 12, the horizontal line represents the upper bound of the



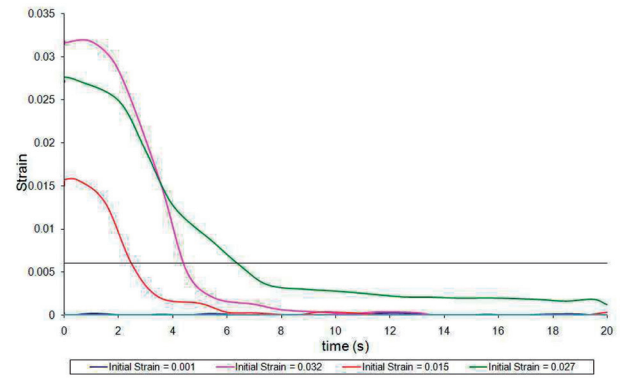
**Figure 9.** Case 1 Temperature–time histories of learned policy, goal strain = 2.7%.



**Figure 11.** Case 2 Temperature–time histories of learned policy, goal strain = 0.1%.



**Figure 10.** Case 1 strain–time histories of learned policy, goal strain = 2.7%.



**Figure 12.** Case 2 strain–time histories of learned policy, goal strain = 0.1%.

**Table 1. Case 1: Commanded voltages, goal = 2.7%.**

$\varepsilon_0$	0.001	0.012	0.027	0.032
$V_{cmd}$	2.38	1.96	2.66	2.66

tolerance range, while the lower bound corresponds to a strain of 0. The initial strains chosen for Case 2 were 0.1%, 3.2%, 1.5%, and 2.7%, which are nearly identical to the initial strains chosen in Case 1. The 0.1% strain is chosen because it demonstrates the ability of the system to remain at the goal strain when already there, and 3.2% is selected because it is the other system boundary. The other strains were chosen because they nearly match the initial strains used in Case 1. Figure 12 shows that for each of these initial strains, the control policy is able to achieve its specified goal, but here it is accomplished for the goal of  $0.1 \pm 0.5\%$  strain. The voltages that the learned action-value function commanded for each of these initial conditions are displayed in Table 2. These values are the commanded voltages for changing the SMA wire from each of these initial strains to the goal strain of 2.7% in 15 s.

**Table 2. Case 2: Commanded voltages, goal = 0.1%.**

$\varepsilon_0$	0.001	0.012	0.027	0.032
$V_{cmd}$	1.12	1.26	0	0

The results for Case 2 demonstrate that the RL agent can learn a voltage–strain SMA control policy for achieving a near zero strain (0.1%) from multiple initial intermediate strains.

## CONCLUSIONS

This article developed a Sarsa-based reinforcement Learning algorithm for learning a voltage–strain control policy and experimentally validated it using a NiTi shape memory alloy wire. This approach for learning a control policy in voltage–strain space was simpler than previous Reinforcement Learning approaches for learning a control policy in temperature–strain space because a separately determined mapping from voltage to temperature is not needed. It is more accurate because the voltage–strain space approach uses accurate

voltage measurements, whereas the temperature–strain approach uses inaccurate thermocouple measurements. Additionally, the control policy in voltage–strain space has the added benefit of being directly useful as a feedback control law for applications.

The learned voltage–strain space control policy permits control in the interior points of the phase transformation process, not just the boundaries. Experimental results presented in this article demonstrate the ability to control a shape memory alloy specimen from arbitrary initial strains ranging from zero to maximum, including intermediate strains, to an arbitrary non-maximum intermediate strain. Additionally, the same control policy is capable of controlling the specimen from similar arbitrary initial values of strain to zero strain.

It is concluded that the Sarsa-based Reinforcement Learning algorithm developed in this article for learning a voltage–strain control policy is a promising candidate for synthesizing useful shape memory alloy actuator feedback control laws.

## ACKNOWLEDGEMENTS

This work was sponsored (in part) by the National Science Foundation Graduate Research Fellowship Program, and the Air Force Office of Scientific Research, USAF, under grant/contract number FA9550-08-1-0038. The technical monitor is Dr. Fariba Fahroo. 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 Science Foundation, Air Force Office of Scientific Research, or the U.S. Government. The authors would also like to acknowledge Dimitris C. Lagoudas and Darren Hartl for their insightful suggestions and comments.

## REFERENCES

- Bae, J.-S., Kyong, N.-H., Seigler, T.M. and Inman, D.J. 2005. "Aeroelastic considerations on shape control of an Adaptive Wing," *Journal of Intelligent Material Systems and Structures*, 16:1051–1056.
- Banks, H., Kurdila, A. and Webb, G. 1997. "Modeling and Identification of Hysteresis in Active Material Actuators, Part (II): Convergent Approximations," *Journal of Intelligent Material Systems and Structures*, 8.
- Barbarino, S., Ameduri, S., Lecce, L. and Concilio, A. 2009. "Wing Shape Control through an Sma-based Device," *Journal of Intelligent Material Systems and Structures*, 20:283–296.
- Bo, Z. and Lagoudas, D.C. 1999. "Thermomechanical Modeling of polycrystalline SMAs Under Cyclic Loading, Part I-IV," *International Journal of Engineering Science*, 37.
- Falk, F. 1989. "Pseudoelastic Stress Strain Curves of Polycrystalline Shape Memory Alloys Calculated from Single Crystal Data," *International Journal of Engineering Science*, 27:277.
- Haag, C., Tandale, M. and Valasek, J. 2005. "Characterization of Shape Memory Alloy Behavior and Position Control Using Reinforcement Learning," AIAA Infotech@Aerospace Conference, Arlington, VA, 26-29 September.
- Kirkpatrick, K. and Valasek, J. 2009. "Reinforcement Learning for Characterizing Hysteresis Behavior of Shape Memory Alloys," *Journal of Aerospace Computing, Information, and Communication*, 6:227–238. March.
- Konidaris, G.D. and Hayes, G.M. 2005. "An Architecture for Behavior-based Reinforcement Learning," *Adaptive Behavior*, 13:5–32.
- Kudva, J.N. 2004. "Overview of the Darpa Smart Wing Project," *Journal of Intelligent Material Systems and Structures*, 15:261–267.
- Lagoudas, D.C., Bo, Z. and Qidwai, M.A. 1996. "A Unified Thermodynamic Constitutive Model for SMA and finite element analysis of Active Metal Matrix Composites," *Mechanics of Composite Materials and Structures*, 3.
- Lagoudas, D., Mayes, J. and Khan, M. 2001. "Simplified Shape Memory Alloy (sma) Material Model for Vibration Isolation," Smart Structures and Materials Conference, Newport Beach, CA, 5–8 March.
- Lagoudas, D.C. 2008. *Shape Memory Alloys: Modeling and Engineering Applications*, Springer Science+Business Media, LLC.
- Machado, L. 2007. "Shape Memory Alloys for Vibration Isolation and Damping," *PhD thesis*, Texas A&M University, December.
- Mitchell, T.M. 1997. *Machine Learning*, The McGraw-Hill Companies, Inc, Singapore.
- Mavroidis, C., Pfeiffer, C. and Mosley, M. 1999. "Conventional Actuators, Shape Memory Alloys, and Electrorheological Fluids," *Automation, Miniature Robotics and Sensors for Non-Destructive Testing and Evaluation*, 10–21. April.
- Patoor, E., Eberhardt, A. and Berveiller, M. 1987. "Potential Pseudoelastic et Plasticité de Transformation Martensitique Dans les Mono-et Polycristaux Metalliques," *Acta Metall*, 35:2779.
- Russell, S. and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*, Pearson Education, Inc, Upper Saddle River, New Jersey.
- Santamaria, J.C., Sutton, R.S. and Ram, A. 1997. "Experiments with reinforcement learning in problems with continuous state and action spaces," *Adaptive Behavior*, 6:163–217.
- Strelec, J.K., Lagoudas, D.C., Khan, M.A. and Yen, J. 2003. "Design and Implementation of a Shape Memory Alloy Actuated Reconfigurable airfoil," *Journal of Intelligent Material Systems and Structures*, 14:257–273.
- Sutton, R. and Barto, A. 1998. *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, MA.
- Valasek, J., Tandale, M. and Rong, J. 2005. "A Reinforcement Learning - Adaptive Control Architecture for Morphing," *Journal of Aerospace Computing, Information, and Communication*, 2:174–195.
- Valasek, J., Doebbler, J., Tandale, M.D. and Meade, A.J. 2008. "Improved Adaptive-reinforcement Learning Control for Morphing Unmanned Air Vehicles," *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, 38:1014–1020. August.
- Varshavskaya, P., Kaelbling, L.P. and Rus, D. 2008. "Automated Design of Adaptive Controllers for Modular Robots Using Reinforcement Learning," *The International Journal of Robotics Research*, 27:505–526.
- Waram, T. 1993. *Actuator Design Using Shape Memory Alloys*, Hamilton, Ontario.
- Webb, G., Kurdila, A. and Lagoudas, D. 1998. "Hysteresis modeling of sma actuators for control applications," *Journal of Intelligent Material Systems and Structures*, 9:432–447.
- Whiteson, S., Taylor, M.E. and Stone, P. 2007. "Empirical Studies in Action Selection with Reinforcement Learning," *Adaptive Behavior*, 15:33–50.