REINFORCEMENT LEARNING IN THE CONTROL OF A
SIMULATED LIFE SUPPORT SYSTEM

by

TODD M. QUASNY, B.S.

A THESIS

IN

COMPUTER SCIENCE

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

Approved

Chairperson of the Committee

Accepted

Dean of the Graduate School

December, 2003

ACKNOWLEDGMENTS

I would like to first and foremost thank my parents for always being there for me, in the good times and especially the bad ones. I would never have been able to do this without you.

I would also like to thank my committee, Dr. Dan Cooke and Dr. Nelson Rushton for all of their guidance and support they have provided me along the way. A special thanks goes to my adviser, Dr. Larry Pyeatt. You saw in me what others could or would not. It is because of your trust and faith that I am able to do the work that I do.

Additional thanks goes to the many friends and family that contributed to this work in many different ways. They include Brett Moore, Marcello Balduccini, Jackie Moore, Ryan Manning, Kathy Manning, Julian Hooker, Keri Ostermann, Amanda Duvall, George (Whip) Koch, Tim Banks, Heather Metz, Dr. Michael Gelfond, Dr. Don Bagert, and countless others. A special thanks goes to David Kortenkamp for his help and guidance on this project, as well as providing some of the figures for this document.

Lastly, I would like to dedicate this thesis to Jarrett; my biggest fan, my best friend, and my loving brother.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
ABSTRACT	v
LIST OF FIGURES	vi
I INTRODUCTION	1
II CONTROL OF AN ADVANCED LIFE SUPPORT SYSTEM	4
2.1 Advanced Life Support System	5
2.1.1 Crew	5
2.1.2 Biomass	5
2.1.3 Food Processing	6
2.1.4 Air Revitalization	6
2.1.5 Water Recovery	7
2.1.6 Power	8
2.2 The BioSim Simulator	9
III REINFORCEMENT LEARNING	11
3.1 SARSA	15
3.2 Q-Learning	16
3.3 Extensions to Traditional Reinforcement Learning	17
IV REINFORCEMENT LEARNING IN ALS CONTROL	18
4.1 Agent Setup	18
4.2 Experimental Setup	20

4.3	Experimental Results	21
V	CONCLUSIONS	24
5.1	Limitations	25
5.2	Future Work	25
	REFERENCES	27

ABSTRACT

Since the 1970s, the National Aeronautics and Space Administration (NASA) has been conducting experiments to improve the duration and safety of manned space missions. For this purpose, an Advanced Life Support (ALS) system is being developed at NASA's Johnson Space Center (JSC). For research and testing purposes, an ALS system simulator, named BioSim, has been developed to simulate the interactions of the various subsystems of ALS. BioSim provides a testbed for researchers to develop and compare alternative control strategies for an ALS system.

Reinforcement learning (RL) is a machine learning technique that finds effective control strategies. RL does this by interacting with the environment, and has been used successfully to control systems with noisy inputs and stochastic actions. RL methods are able to perform the real-time, reactive control that is vital in embedded control systems.

This work demonstrates that reinforcement learning provides an excellent approach for finding an effective control policy for the water recovery subsystem of an ALS system. The control policy found by RL overcomes the inherent noisy inputs and stochastic actuation methods that exist in ALS systems. Using the policy found by RL, the mission duration is extended to nearly 450 days, at which point the mission ends for reasons other than the lack of consumable water. Since the mission does not end due to water concerns, it is concluded that an effective control policy for the water recovery system has been generated.

LIST OF FIGURES

2.1	The Air Revitalization System	7
2.2	The Water Recovery System	8
2.3	The BioSim ALS Simulator	10
3.1	Reinforcement Learning Agent Model	12
3.2	The SARSA On-Policy TD Control Algorithm	16
3.3	The Q-Learning Off-Policy TD Control Algorithm	16
4.1	The Reinforcement Learning agent for the control of BioSim	20
4.2	Performance of the RL agent with a fixed α	21
4.3	Performance of the RL agent with a decaying α	22
4.4	Squared error of the value function for each episode during learning .	23

CHAPTER I

INTRODUCTION

To make an extended duration space mission, such as a mission to Mars, a reality, a new life support system must be developed that is able to utilize a relatively small amount of resources as well as regenerate consumables like air, water, and food (Escobedo 2003). For this purpose, the National Aeronautics and Space Administration (NASA) has created an Advanced Life Support (ALS) group based out of the Johnson Space Center (JSC) in Houston, TX. The ALS group is charged with the task of developing a life support system that will allow future long-duration missions, as well as missions involving planetary habitats.

To produce such a system, many milestones must be reached. Such milestones include numerous tests of the various subsystems of an ALS system (for a description of the subsystems, see Section 2.1). A number of tests have been conducted at JSC, many of them involving human test subjects. These tests are to evaluate both the effectiveness and reliability of regenerative systems for extended length missions (Escobedo 2003).

To facilitate such evaluations, the ALS group at JSC has commissioned a piece of software that will simulate an ALS system and allow evaluations of many different control systems on a common platform. The resulting software application, named BioSim, is currently being developed by Metrica, Inc. in Houston, TX. BioSim is

described in more detail in Section 2.2.

BioSim provides a specific environment in place in which to develop and experiment with various types of control for an ALS system. Researchers from many different fields of control theory can work on this common platform in which uniform metrics can be used. In terms of artificial intelligence (AI), a number of specific fields of AI and how they could possibly contribute to the control of an ALS system are described in (Bonasso, Kortenkamp, and Thornesbery 2003). One area described is Machine Learning (ML); which, among other uses, may help the problems of sensor noise.

Reinforcement Learning (RL) is one ML technique that has received a considerable amount of research in the area of control. RL methods are outlined in Chapter III. RL has the ability to cope with sensor noise while still providing real-time reactive control.

The goal of this work is to demonstrate that reinforcement learning provides a suitable approach for finding an optimal control policy for the water recovery subsystem of an ALS system. The control policy must overcome the inherent noisy inputs and stochastic actuation methods that exist in ALS systems. This work is focused exclusively on controlling the water recovery system because that system appears to be the most critical system for keeping the crew alive. If the water recovery system is left uncontrolled, the crew *always* dies of thirst within 25 days. Successful control of the water recovery system is indicated by longer mission duration and the termination

of the mission due to reasons other than the crew dying of thirst. In the experiments run for this work, validation of the control policy was performed using BioSim.

Chapter II will give a more detailed description of the problem, including enumerating and describing the subsystems of an ALS system as well as their simulator. Chapter III gives a detailed description of RL and the pertinent algorithms. Chapter IV will then present how RL will be applied to the control of BioSim as well as provide results of the work.

CHAPTER II

CONTROL OF AN ADVANCED LIFE SUPPORT SYSTEM

Since the 1970s, the National Aeronautics and Space Administration (NASA) has been conducting experiments to see if the duration and safety of manned space missions can be improved. In order to do this, NASA is focusing on an Advanced Life Support (ALS) system. Such a system would provide all of the resources needed by a manned crew, while at the same time either regenerating them or utilizing them to their fullest abilities.

The ALS system is composed of many subsystems (see Section 2.1) that must all work together. The interactions between the various subsystems can be extremely complex in terms of resource requirements and timings. Any slight change in the control of one subsystem can have many ripple effects on all of the other subsystems (Kortenkamp and Bell 2003). To effectively control an ALS system, a controller must be aware of all the possible effects of even the slightest of actions.

An additional difficulty of controlling an ALS system is that of hardware consistency. Sensors and actuators that make up the ALS system are not perfect and sometimes fail. Over time, sensor readings and actuation mechanisms will change or drift due to degradation of the physical underlying equipment. As for failures, sometimes they can be repaired, but at other times they cannot. Therefore, alternative control strategies must be in place to adaptively handle such failures.

The goal of an intelligent controller for an ALS system is to extend the duration of the mission for as long as possible. The mission is considered over when at least one of the crew members can no longer survive given the current amount of consumable food, potable water, and breathable air.

2.1 Advanced Life Support System

The following sections give overviews of the subsystem components of an ALS system as described in (Kortenkamp and Bell 2003).

2.1.1 Crew

The crew refers to the mission's astronauts. The size of the crew as well as their genders, ages, and weight are parameters for the model described in (Goudarzi and Ting 1999). As the crew performs its daily activities (exercise, scientific experiments, maintenance, etc), they are consuming O_2 , food, and water. At the same time, they are producing CO_2 , dirty water, and solid waste. The activities that the crew is performing is directly related to the resource requirements of the overall system.

2.1.2 Biomass

Biomass refers to the various crops that can be grown in a fixed area inside of a sealed environment. The environment for the crops is kept separate from the

crew because the ideal gas mixture for growing the crops is not safe for people (Edeen 1993). CO_2 , potable water, and grey water are consumed as the crops grow. In turn, the crops produce O_2 and transpire H_2O .

Once the crops have grown to maturity, they are harvested and biomass is produced. The biomass can then be processed into food that the crew can consume (see Section 2.1.3).

2.1.3 Food Processing

Food processing refers to the process of producing food from harvested biomass (see section 2.1.2) and energy (power). After the processing step, the crew is then able to consume the food. Food processing is viewed as a separate process from the growing and harvesting of crops because it is not a continuous process and has different power requirements.

2.1.4 Air Revitalization

The purpose of the air revitalization subsystem is to remove toxic gases from the crew environment. The majority of these gases are CO_2 . The air revitalization subsystem produces O_2 , CO_2 , and H_2O and places the gases in stores to be used by other subsystems. Methane (CH_4) is also produced and vented, or possibly used for fuel. The air revitalization system is composed of a number of other subsystems including the Variable Configuration CO_2 Removal System (VCCR), the CO_2 Reduc-

Figure 2.1: The Air Revitalization System. Picture courtesy of David Kortenkamp, NASA JSC/Metrica Inc.

tion System (CRS), and the O₂ Generation System (OGR). The interaction between these subsystems as well as the CO₂ and O₂ stores is shown in Figure 2.1. The air revitalization process discussed is based on a test at the NASA Johnson Space Center (Schreckenghost, Ryan, Thronesbery, and Bonasso 1998).

The air revitalization process is an especially difficult one in terms of control due to the fact that it is responsible for both the crew environment and the crop environment. A careful balance between the two must be kept while at the same time conserving as much power and other resources as possible (Kortenkamp and Bell 2003).

Figure 2.2: The Water Recovery System. Picture courtesy of David Kortenkamp, NASA JSC/Metrica Inc.

2.1.5 Water Recovery

The basic function of the water recovery system (WRS) is to consume dirty water (water that contains waste and other elements) and produce potable (human-consumable) water as well as grey water. Grey water is water that is suitable for washing but not for human consumption. The systems that comprise the WRS are the biological water processing (BWP) subsystem, the reverse osmosis (RO) subsystem, the air evaporation system (AES), and the post-processing subsystem (PPS). These subsystems and their interactions are shown in Figure 2.2.

2.1.6 Power

The power subsystem, as the name suggests, supplies power to all of the other subsystems. There are two types of power subsystems to consider, a nuclear-style power system and a solar-style power system. The nuclear-style system provides power that is continuous at a fixed amount. The solar-style system provides varying amounts of power depending on the amount of solar energy that is available (Kortenkamp and Bell 2003).

2.2 The BioSim Simulator

As stated previously, the task of controlling an ALS system is a challenging one due to the tight interaction of the subsystems (Kortenkamp and Bell 2003). In order to facilitate research in the area of controlling such a system, BioSim has been developed by NASA JSC/Metrica Inc. The goal of BioSim is to simulate each individual subsystem in an ALS system and their interactions. A diagram of the components that form BioSim is given in Figure 2.3.

BioSim consists of various modules. Each module is implemented as a CORBA object. Therefore, any programming language that has CORBA extensions is able to interface with BioSim. This allows a wide range of techniques to be used for ALS system control. With a common platform to test multiple control techniques, concrete metrics can be developed with which to evaluate different control strategies.

Figure 2.3: The BioSim ALS Simulator. Picture courtesy of David Kortenkamp, NASA JSC/Metrica Inc.

CHAPTER III

REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a machine learning technique that learns through interaction with the environment (Sutton and Barto 1998). The fundamental idea in RL, which comes from biology, is that when an agent performs actions that are desirable, it receives a positive reward signal. This is similar to a biological system's concept of pleasure. Conversely, when an agent performs actions that are not desirable, it receives a negative reward signal. The ability to link cause and effect, even incorrectly, is a basic ability of even the simplest of animals (Pyeatt 1999).

An RL agent chooses what action to take based on the agent's sensory input. Once the agent performs the action, the agent receives a reinforcement signal from the environment as well as information about the new state of the environment. As illustrated in Figure 3.1, at each time step the agent receives an indication of the current state and the reinforcement signal. The reinforcement signal can be zero, positive, or negative. The goal of the agent is to select actions that maximize this reward signal in the long run. To do this, the agent creates an action selection *policy*, a function mapping states to actions.

The RL agent creates this policy by finding an estimate of the value of taking a given action in a given state. This value, over all states and actions, is called the *value function*. A value function (V) is a mapping from inputs that represent the current

Figure 3.1: An RL agent interacts with the environment by choosing actions. It then receives a reward signal as well as the next state from the environment.

state (s) to a number (v) that is the expected long term reward when starting in state s . RL methods provide ways to learn the value function, which leads to a policy.

A policy π is better than or is equivalent to another policy π' if and only if $V^\pi(s) \geq V^{\pi'}(s) \forall s \in S$, where S is the set of all states that the agent can be in (Sutton and Barto 1998). This establishes a partial ordering of policies. If the value of a policy π is greater at every state than the policy π' , then π is said to be better than π' . The *optimal policy* is the policy that is better than, or equal to, all other policies. There can be more than one optimal policy, but they are all denoted π^* . Optimal policies share the same *optimal state-value function*, denoted V^* . The optimal state-value function is defined as,

$$V^*(s) = \max_{\pi} V^\pi(s), \forall s \in S. \quad (3.1)$$

The optimal policy, simply stated, is the policy that maximizes the value function at every state. Most often the RL agent must choose the optimal action to take given,

the current state. For this, the *optimal action-value function* is used. It is defined as,

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) \forall s \in S, a \in A \quad (3.2)$$

where A is the set of all actions that an agent can take. An RL agent can select the best action to take in a given state s by evaluating Equation 3.2 for all actions $a \in A$ and choosing the action with the largest value.

Q^* can be expressed in terms of V^* as,

$$Q^*(s, a) = E \{ r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a \} \quad (3.3)$$

where $E \{x\}$ is the expected value of x , γ is a discount rate that weighs future rewards, and r_{t+1} is the reward at the next time step. One can see from Equation 3.3 that Q^* is simply the expected value of the immediate reward for choosing action a in state s and then following the optimal policy from the next state (s_{t+1}).

The Bellman optimality equation states that the value of a state under an optimal policy is equivalent to the expected return for taking the best action from that state (Sutton and Barto 1998). Using Equation 3.3, the Bellman optimality equation is,

$$V^*(s) = \max_a E \{ r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a \} . \quad (3.4)$$

For Q^* , the Bellman optimality equation is,

$$Q^*(s, a) = E \left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a \right\}. \quad (3.5)$$

Given a state s , the *greedy* action for that state is the action with the highest expected reward (see Equation 3.2). The simplest, and therefore the most commonly used, way to ensure that the RL agent is exploring the state space is for the agent to choose a random action occasionally. One method for exploring the state space is known as ϵ -greedy action selection. Using this technique, the probability of the RL agent choosing a greedy action in state s is,

$$1 - \epsilon + \frac{\epsilon}{|A(s)|}, \quad (3.6)$$

where ϵ is the probability of choosing a random action. Therefore, the probability of the agent choosing a non-greedy action is,

$$\frac{\epsilon}{|A(s)|}. \quad (3.7)$$

The policy that the RL agent is then following is called an ϵ -greedy policy.

Equations 3.4 and 3.5, combined with techniques for exploration of the state space, such as ϵ -greedy action selection, serve as the basis for Temporal Differencing (TD) methods that are the core of RL (Sutton and Barto 1998). Sections 3.1 and 3.2

cover the two main TD methods that utilize ϵ -greedy action selection.

There are two major classes of TD methods used in RL: on-policy and off-policy. On-policy methods update the value of the actions given by the current policy, that is, the policy currently being used to make decisions. Off-policy methods update the value of the actions given by an alternative policy (Sutton and Barto 1998). In off-policy learning, the policy used to select actions is called the *behavior* policy, while the policy that is being updated is called the *estimation* policy (Pyeatt 1999). Both on-policy and off-policy methods utilize stochastic methods, such as ϵ -greedy, for action selection.

3.1 SARSA

The most basic on-policy TD control update rule, derived from Equation 3.3, is known as SARSA. The name SARSA comes from the tuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ which is present in the update equation in the SARSA algorithm (Sutton and Barto 1998). The SARSA update equation is,

$$\Delta Q_t(s_t, a_t) = \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \quad (3.8)$$

where $Q(s_t, a_t)$ is the estimated value of state s at time step t given action a_t , r_{t+1} is the reward at time step $t + 1$, α is a learning rate parameter, and γ governs how important future rewards are relative to the immediate reward. The entire SARSA

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode)
  Initialize  $s$ 
  Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
  Repeat (for each step in the episode)
    Take action  $a$ , observe reward  $r$ , and next state  $s'$ 
    Choose  $a'$  from  $s'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
     $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s', a \leftarrow a'$ 
  Until  $s$  is terminal

```

Figure 3.2: The SARSA On-Policy TD Control Algorithm (Sutton and Barto 1998)

on-policy control algorithm is given in Figure 3.2.

3.2 Q-Learning

The most commonly used off-policy TD control method is Q-learning (Watkins 1989), which follows one policy while updating another. In the most basic form, known as one-step Q-learning (Sutton and Barto 1998), the following update equation is used:

$$\Delta Q_t(s_t, a_t) = \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \quad (3.9)$$

The objective of an RL agent is to learn the value of each state-action pair in the state space (Sutton and Barto 1998). In Q-learning, the function Q is a direct approximation of Q^* . This approximation is improved iteratively, independent of the policy that the RL agent is currently following. The entire Q-learning off-policy control algorithm is given in Figure 3.3.

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode)
  Initialize  $s$ 
  Repeat (for each step in the episode)
    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $a$ , observe reward  $r$ , and next state  $s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$ 
     $s \leftarrow s'$ 
  Until  $s$  is terminal

```

Figure 3.3: The Q-Learning Off-Policy TD Control Algorithm (Sutton and Barto 1998)

3.3 Extensions to Traditional Reinforcement Learning

Sections 3.1 and 3.2 presented the two major TD control algorithms for reinforcement learning. Each algorithm finds the function $Q(s, a)$. The representation of this function becomes difficult as the number of states and actions grow large (Sutton and Barto 1998). In order to cope with this, in practice RL algorithms employ various function approximation techniques, including neural networks and decision trees (Sutton and Barto 1998; Pyeatt and Howe 1995).

Another extension to RL that is commonly used is *eligibility traces*, or *e-traces*. E-traces use n -step prediction (using future rewards to update the current value function), assigning an *eligibility* to each future reward (Sutton and Barto 1998). The eligibility is discounted by λ^k for the k^{th} time step in the future.

CHAPTER IV

REINFORCEMENT LEARNING IN ALS CONTROL

As shown in Chapter III, reinforcement learning provides a machine learning technique that finds optimal control strategies through interacting with the environment. Since the RL agent learns from interacting directly with the environment, RL is able to cope with noisy inputs from sensors. For this reason, RL provides an excellent technique for controlling an ALS system (Bonasso, Kortenkamp, and Thornesbery 2003).

The goal of a controller for an ALS system is to extend the mission length for as long as possible. The most significant aspect in maintaining the crew is the production of consumable, or potable, water. Therefore, for this work, the goal is to develop an RL agent to perform control on the water recovery subsystem.

4.1 Agent Setup

Four inputs were chosen to describe the water recovery subsystem's state to the RL agent. These inputs are:

1. **Dirty Water Store Level:** the amount of dirty water (in liters) that is in the dirty water store. The dirty water store contains the dirty water for use by all subsystems in the ALS system.

2. **Grey Water Store Level:** the amount of grey water (in liters) that is in the grey water store. The grey water store, like the dirty water store, contains the grey water for use by all subsystems in the ALS system.
3. **Potable Water Store Level:** the amount of potable water (in liters) that is in the potable water store. Similar to the dirty and grey water stores, this contains the potable water available to all of the subsystems in the ALS system.
4. **Thirsty?:** a boolean value that is *true* when at least one of the crew members is thirsty. Otherwise, this value is *false*.

Each input, excluding the boolean input **Thirsty?**, are continuous, real values. The three continuous inputs are discretized into thirty discrete values. Thus, the state space forms a table with 2×30^3 entries.

The agent's actions are able to control the flow rates from the dirty and grey water stores into the water recovery system, as well as the flow rate from the potable water store to the crew. Each flow rate can either be turned *off* or *on*. Thus, the action value table $Q(s, a)$ contains 2^6 entries. Therefore, the entire state space table for the RL agent consists of $2^7 \times 30^3 = 3.456 \times 10^6$ entries.

An illustration of the controller, with its inputs and outputs, is given in Figure 4.1.

Figure 4.1: The Reinforcement Learning agent for the control of BioSim

4.2 Experimental Setup

Due to the fact that the goal of the RL agent is to extend the mission duration (i.e., the lives of the crew) for as long as possible, and that there are no other constraints put on how the agent is supposed to achieve this goal, the natural reward selection for the problem is to give a negative reward (a punishment) when the mission is over, and zero at all other times. This way the agent is free to “experiment” without being punished, unless the actions taken lead to the end of the mission.

Given the large number of time steps in each episode (often $> 10,000$), a γ of 0.9 and a λ of 0.85 is used. α is initially set to 0.1 and is decayed by 0.1 percent after each episode. Random actions are chosen fifteen percent of the time to start with and a decay factor of 0.56 percent is used.

The RL agent was trained by running numerous episodes controlling BioSim. Through the agent’s interaction with BioSim, optimal control strategies were developed. A more detailed description of the agent’s performance is given in Section 4.3.

Figure 4.2: Performance of the RL agent with a fixed α

4.3 Experimental Results

The RL agent, after approximately 1200 episodes, converged to an optimal policy. This policy extends the mission to 447 days, at which time the crew begins to starve. Since the agent is only controlling the water recovery system, the lack of food to the crew is not controlled by this agent. By achieving a mission length of 447 days, the agent has effectively done as good as it possibly can given its ability to control the system.

Two variations of the experiment were run, one with a constant fixed α value (0.1), and one with the previously described decaying alpha. A windowed average of mission duration over 100 episodes is shown for the fixed- α and decaying- α in Figures 4.2 and 4.3, respectively. Figure 4.2 indicates that despite being able to

Figure 4.3: Performance of the RL agent with a decaying α

significantly modify the value function late in the agent's training. Figures 4.2 and 4.3 both show the agent learning the optimal policy.

During training, the squared error of the value function as the agent interacted with the environment was calculated. This metric provides an insight into how long the agent needed to train, as well as when an optimal, or near optimal, policy was reached. Figure 4.4 shows the calculated squared error for the fixed- α agent during training. This graph shows that the change in the value function slowly decreased as the agent gained more experience.

Figure 4.4: Squared error of the value function for each episode during learning

CHAPTER V

CONCLUSIONS

The goal of this work was to produce an intelligent controller for an advanced life support system. This controller was tested using the ALS simulator, BioSim. The objective of the controller was to extend the duration of a mission by any means necessary. Furthermore, the controller was required to handle the inherently noisy inputs from the sensors as well as the inconsistencies of actuation methods in BioSim while providing reactive, real-time control.

This work used reinforcement learning to create an intelligent agent that learned through interacting with its environment (BioSim) and found optimal control policies for controlling the water recovery subsystem of ALS. The RL agent was able to do this even given the stochastic nature of the sensory input and the actuation methods. This shows that intelligent, reactive control can be learned for an ALS system and provides an excellent base for controllers.

The resulting control strategy learned by the RL agent extends the mission length to a point in which there is no longer enough food to support the human crew. Therefore, since the RL agent has no control over the amount of food produced, an optimal control policy for the water recovery system has been reached.

5.1 Limitations

As stated in (Littman 1994), no agent lives in a vacuum. Therefore, any control agent must be aware of other rational agents that are acting upon the system, whether they are autonomous agents or humans. This work is not proven to converge to an optimal control strategy if there are other agents controlling any aspect of the ALS system that would have an effect on the agent’s inputs (Ivančić 2001). This can be a difficult assumption to make in many cases since in a production system, a human will ultimately have control of any part of the system if necessary, while the agent will be completely unaware of an interloper’s actions.

5.2 Future Work

This work provides a sound base for further study into this problem. A control system should be developed that controls all of the subsystems in the ALS system in order to extend the mission length to multiple years. Several variations of reinforcement learning, including function approximation in RL and multiagent RL, could possibly manage this large and complex control scenario.

Additionally, the control strategies learned from the agent should be extracted into finite state machines and thus simple programming constructs such **if-then-else** or **switch** statements. These can then be tested on actual hardware as a form of validation. This would provide insights into how well the models of BioSim simulate the real world situations. Without actual real-world experience, there can be no conclu-

ve evidence that the control strategies developed by this work is actually viable to
.LS.

REFERENCES

- Bonasso, P., D. Kortenkamp, and C. Thornesbery (2003, Spring). Intelligent control of a water-recovery system: Three years in the trenches. *AI Magazine*, 19–44.
- Edeen, M. (1993). Modeling and validation of the ambient and variable pressure growth chamber models. SAE Technical Paper Series 932171.
- Escobedo, V. E. (2003, August). JSC Advanced Life Support Web Page. Available via WWW: <http://advlifesupport.jsc.nasa.gov/>.
- Goudarzi, S. and K. Ting (1999). Top level modeling of crew component of ALSS. In *Proceedings International Conference on Environmental Systems*.
- Ivančić, F. (2001, March). Reinforcement learning in multiagent systems using game theory concepts. Technical report, University of Pennsylvania.
- Kortenkamp, D. and S. Bell (2003, July). Simulating advanced life support systems for integrated controls research. In *Proceedings 33rd International Conference on Environmental Systems (ICES)*.
- Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, San Francisco, CA, pp. 157–163. Morgan Kaufmann.
- Pyeatt, L. D. (1999). *Integration of Partially Observable Markov Decision Processes and Reinforcement Learning for Simulated Robot Navigation*. Ph. D. thesis, Colorado State University.
- Pyeatt, L. D. and A. E. Howe (1995). Decision tree function approximation in reinforcement learning. Technical Report TR CS-98-112, Colorado State University.
- Schreckenghost, D., D. Ryan, C. Thornesbery, and R. P. Bonasso (1998). Intelligent control of life support systems for space habitats. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*.
- Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Watkins, C. (1989). *Learning from Delayed Rewards*. Ph. D. thesis, Cambridge University.

PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Texas Tech University or Texas Tech University Health Sciences Center, I agree that the Library and my major department shall make it freely available for research purposes. Permission to copy this thesis for scholarly purposes may be granted by the Director of the Library or my major professor. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my further written permission and that any user may be liable for copyright infringement.

Agree (Permission is granted.)

Student Signature

Date

Disagree (Permission is not granted.)

Student Signature

Date