Using Learning Algorithms to Develop Dynamic Gaits for Legged Robots

by

Brian Schaaf

B.S., Mechanical Engineering
B.S., Electrical Engineering
Duke University, 2004

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2006

Signature of the Author _____
Department of Aeronautics and Astronautics
May 26, 2006

Certified by _____
Greg Andrews
Charles Stark Draper Laboratory
Thesis Supervisor

Certified by _____
Brent Appleby
Lecturer in Aeronautics and Astronautics
Charles Stark Draper Laboratory
Thesis Advisor

Accepted by _____
Jaime Peraire
Chairman, Departmental Graduate Committee

Using Learning Algorithms to Develop Dynamic Gaits for Legged Robots

by

Brian Schaaf

Submitted to the Department of Aeronautics and Astronautics on May 26th, 2006
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

As more legged robots have begun to be developed for their obvious advantages in overall maneuverability and mobility over rough terrain and difficult obstacles, their shortcomings over flat terrain have become more apparent. These robots plod along at extremely low speeds even when the ground is flat and level due to the fact that virtually all legged robots use a very stable, very slow walking gait to move, regardless of whether the ground is flat or rough.

The simplest way of solving this problem is to use the same method as legged animals: simply change the gait from a walk to a faster more dynamic gait in order to increase the robot's speed. It would be extremely useful if legged robots were capable of moving across flat ground at high velocities while still retaining their ability to cross extremely rough or broken ground. Unfortunately, dynamic gaits are quite difficult to program by hand and only minimal research has been done on them.

This thesis evaluates the use of two different types of learning algorithms (a genetic algorithm and a modified gradient-climbing reinforcement learning algorithm) as applied to the problem of developing dynamic gaits for a simulation of the Sony Aibo robot. The two algorithms are tested using a random starting population and a high-fitness starting population and the results from both tests are compared.

The research focuses on three different types of dynamic gaits: the trot, the canter, and the gallop. The efficiencies of the learned gaits are compared to each other in order to try to determine the best type of high-speed gait for use on the Aibo robot. Problems with the design of the Aibo robot as related to performing dynamic gaits are also identified and solutions are proposed.

Thesis Supervisor: Greg Andrews
Title: Member of the Technical Staff

Thesis Advisor: Brent Appleby
Title: Lecturer in Aeronautics and Astronautics

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1 Introduction

The field of robotics has two main branches, industrial robotics and mobile robotics, but until relatively recently researchers focused primarily on industrial robotics. This was because the large robots used in industrial applications were relatively easy to design and program. They sat in fixed positions and performed the same tasks over and over again. This meant that they required no real intelligence or sensing ability, since their programmers knew everything about the robot's environment beforehand and therefore could program in the precise set of actions they wanted the robot to carry out. Mobile robots, on the other hand, have to operate in constantly changing and unpredictable environments. Thus, these kinds of robots needed sensors to learn about their surroundings and enough intelligence to process the data from these sensors and to determine the best course of action. The difficulty of achieving this, combined with the limited range of mobile robots due to power constraints kept these robots from being anything but university research projects for some time.

However, in the last decade or so processors and power supplies have advanced to the point where mobile robots can now actually be used for practical purposes. Mobile robots are currently used for entertainment, search and rescue, scouting, inspection of hazardous environments, and many other purposes. The military in particular has come to see the massive potential of mobile robots in helping to gather information and limit casualties in future conflicts.

The simplest mobile robots use wheels or tracks to move around because those locomotion systems are fast, efficient, and easy to control. Unfortunately, they cannot effectively traverse the more extreme terrain that is commonly found in the real world. On a battlefield in particular, a mobile robot is likely to have to deal with crossing large gaps, rubble, and rocky terrain. Thus, the military is turning more towards legged robots, which have the ability to handle almost any terrain because the robot can vary the height and placement of its feet to adapt to the topography of the ground. Legged robots can also use their legs to lift their bodies up, allowing them to

climb over obstacles that would block a normal wheeled robot and they can go up slopes that wheeled robots could not simply by leaning their bodies forward and thus shifting their center of mass as the slope changes.

This ability to adapt to changing conditions and overcome most obstacles ensures that the recent interest in legged robots will continue to grow as they become more sophisticated. The military is already hoping to use them for scouting missions where sending in a person would be too dangerous and for carrying supplies places ordinary vehicles could not reach. However, legged robots still possess a number of drawbacks and there is considerable room for improving them through future research.

1.1 Problem Statement

As more legged robots have begun to be developed for their obvious advantages in overall maneuverability and mobility over rough terrain and difficult obstacles, their shortcomings over flat terrain have become more apparent. Wheeled vehicles can cover flat terrain very efficiently and at a high rate of speed while legged platforms have been forced to plod along at nearly the same speeds they use to traverse rough terrain. This is primarily due to the fact that virtually all legged robots use a very stable, very slow walking gait to move, regardless of whether the ground is flat or rough [2, 3, 9, 10, 16, 21, 23, 35, 36, 39, 40, 43, 45, 48, 50, 53, 56, 59].

Some attempts have been made to get around this problem by adding powered wheels or even skates to legged robots in order to allow them to travel across level ground much faster [22]. Unfortunately, the additional drive trains, wheels, and other parts necessitated by this approach add a considerable amount of weight and complexity to the robot making the robot more expensive to build and much more likely to break.

A simpler way of solving this problem is to use the same method as legged animals: simply change the gait from a walk to a faster more dynamic gait in order to increase the robot's speed. Changing gaits allows an animal to move much faster and

more efficiently than they can at a walk by decreasing the amount of legs on the ground at any given time. This allows faster movement because the animal's body can move forward more per step. In a walk, at least three feet are on the ground at all times and only one foot is moved at a time, thus it takes three individual steps for the animal's body to move forward one stride length. In a trot, two feet are on the ground at any given time, but the two feet move in concert, so it only takes one step to move the animal's body forward one stride length. In a gallop, there are times when no feet are on the ground, which allows the animal to greatly extend its effective stride length and move even faster than it can while trotting.

The benefit of these more dynamic gaits is that they can greatly increase the forward movement of an animal without requiring a corresponding increase in the speed of the leg movements. For example, a gallop might be 5 times faster than a walk, but only require the animal to move its legs twice as fast as it would when walking. The drawback of these gaits is, of course, that they are far less stable than a walk. When walking, a four legged robot can be paused at any point in the gait and it will not fall over because it always has a stable tripod of three legs to stand on. Such is not the case with faster gaits such as trots and gallops. The only reason an animal can remain upright while performing these gaits is due to the inertia of its body. The slower the animal goes the less stable the gait becomes. This dynamic instability is why these dynamic gaits are only truly viable over smooth terrain.

Nevertheless, a large proportion of the world does consist of smooth terrain, especially in and around cities and other places humans live, which is where most robots will be used. Therefore, it would be extremely useful if legged robots were capable of moving across flat ground at speed while still retaining their ability to cross extremely rough or broken ground. Currently, legged robots are not used nearly as much as wheeled ones in large part due to how extremely slow they are. The ability to utilize dynamic gaits to achieve speeds nearer to those of their wheeled counterparts would close this gap and make legged robots far more useful and versatile than they currently are.

1.2 Objectives

The objective of this thesis is to demonstrate that machine learning algorithms can be used to develop fast and efficient dynamic gaits for legged robots. Dynamic gaits show great potential for significantly expanding the capabilities of legged robots, but have proven extremely difficult to program properly due to their instability and the large number of parameters that define them. An individual gait might be made up of twenty or so parameters and can be very sensitive to any perturbations in those parameters. This makes finding a decent gait via either trial and error or systematic search methods almost impossible. Machine learning algorithms, however, are designed to solve this exact type of problem, where the search space is extremely large and possesses many local maxima. Thus these algorithms should prove more successful at developing dynamic gaits than the trial and error or mathematically driven attempts that many other researchers have used.

This thesis will also compare the results achieved by two very different forms of learning algorithms to both each other and to manually developed gaits in an attempt to determine which approach is more suited to solving the problem. The two learning algorithms that were chosen were a genetic algorithm (GA) and a gradient-climbing reinforcement learning algorithm (GCRL). Genetic algorithms are good at exploring very large search spaces and zeroing in on the general area of overall maxima. Gradient-climbing reinforcement learning algorithms tweak a single base set of parameters to slowly improve it. They are good at finding the specific parameters of a maxima, but generally the maxima they find is a local one rather than a global one. Thus, the best gaits will most likely be generated by using the GA to hone in on a few promising parameter sets and then using the GCRL to refine them.

1.3 Overview

The outline of this thesis is as follows: Chapter 2 provides some background information on different types of gaits, defining what various gaits are called, when they are used, and what differentiates them from each other. It then details what kind of research has been done with gaits as applied to robotics and gives examples of several experiments that have been done to try and enable legged robots to move more quickly across smooth terrain.

Chapter 3 describes the support programs that were written so that the machine learning algorithms could be applied to the problem of gait generation. The chapter first focuses on the simulator that was used to test out all the gaits in a controlled, though still randomized, environment. Then, it explains the gait program that was written to take in a set of parameters and output the commands necessary for the simulated robot to move. This section will also detail what all the gait parameters are and why each was chosen.

Chapter 4 focuses on the genetic algorithm, detailing how it works, why it was chosen, and then presenting the results from the tests using it. There were two rounds of learning: in the first, gaits were generated using a random starting population, in the second, the best gaits from several runs of the first round of testing were used as the starting population. The chapter concludes with an analysis of the gaits that were found in the second phase of learning.

Chapter 5 is laid out exactly like chapter 4, but focuses on the gradient-climbing reinforcement learning algorithm rather than the GA. Background on the algorithm is given, the results from the two rounds of learning are shown, and then the gaits from the second round of learning are analyzed.

Chapter 6 compares the results from the two learning methods for both rounds of testing and analyses which is better and why. It also compares the different types of gaits that were generated to determine the overall best type of gait for high-speed running with the Aibo. The chapter concludes by detailing various lessons that were learned during the course of the research, identifies problems with the design of the

Aibo with respect to dynamic running, and proposes some solutions to these problems.

Chapter 7 summarizes the information presented in the thesis. This chapter also goes over future areas of research, such as possible improvements to the gait controller, testing the gaits on actual hardware, and incorporating sensor data into the learning algorithms so as to allow online learning by the robot.

Chapter 2 Background

2.1 Overview of Gaits

There are several different ways to classify gaits, but the main categories used are symmetric/asymmetric and static/dynamic [20]. Symmetric gaits are those in which the phase difference in the footfalls of the legs on one side of the animal's body mirrors the phase difference of the legs on the other side of the body [66]. Examples of symmetric gaits include walking, trotting, and pacing, among others. Asymmetric gaits are gaits in which the phase differences between the legs on one side of the body do not mirror the phase differences between the legs on the other side of the body [19]. Canters, gallops, and bounds are examples of asymmetric gaits. Symmetric gaits tend to be more stable than asymmetric gaits due to the mirroring of the legs on each side of the body which helps to counter pitching and rolling during the gait.

Static gaits are gaits in which there is a stable triangle of legs on the ground throughout the entire gait and the center of mass remains within this triangle. An illustration of this is shown in Figure 2.1. These gaits are called static because the animal could be frozen at any point in the gait without falling over. The only truly static gaits are various types of walks. Dynamic gaits do not have a stable triangle of feet at all times and in fact the number of feet on the ground at any given time can vary from zero to four throughout the course of the gait. Examples of dynamic gaits are trots, canters, paces, and gallops.

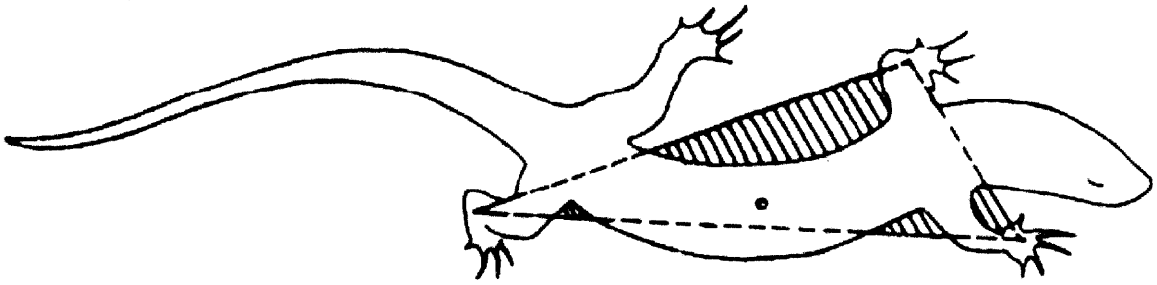

Figure 2.1: Stability Triangle for a Lizard [15]

In general, asymmetrical dynamic gaits are the fastest and least stable, while symmetrical static gaits are the slowest and most stable. For the purposes of this research we will be focusing on dynamic gaits since attaining high rates of speed is the primary goal. The speed of a legged system is primarily determined by the amount of legs in the return phase at any given time [60]. The return phase is the portion of the leg's movement when it is going forward. The maximum speed of the legs during return is limited because the animal lets the legs swing forward like pendulums so that they do not have to input any energy into the legs during this phase. Therefore there is a lower limit on the stride frequency for any given gait. Increasing forward speed therefore requires increasing the stride length of a gait. Stride length is the amount of forward distance covered by the body during the course of one full leg cycle.

Having more legs in the return phase means that fewer of the legs are moving backwards and propelling the animal. Thus the animal is getting more overall forward body movement per leg during each leg's propulsion phase, which means that as the number of legs in return increases, so does the stride length and the overall travel speed. For example, in a walk, the number of legs in return averages out to be around one; therefore three legs are being used to propel the animal forward one stride length. In a trot, diagonal pairs of legs move in sequence, so the number of legs in return varies from slightly less than two, for slow trots that have periods with all four legs on the ground, to slightly more than two for trots that have a brief flight phase. Because of this, trots tend to be significantly faster than walks.

Flight phases during a gait have the effect of greatly increasing the stride length by increasing the average number of legs in return, since during the flight phase all

four legs are in return. The longer the flight phase lasts, the greater the stride length increase and the greater the overall speed of the animal, even though the frequency of each leg cycle doesn't change much. As a matter of fact, stride frequency can change by as little as ten percent over a doubling of a quadruped's forward speed [17]. The faster a runner wants to move, the longer the flight phase has to be. This is why the fastest gallops actually contain two flight phases during the course of a single stride.

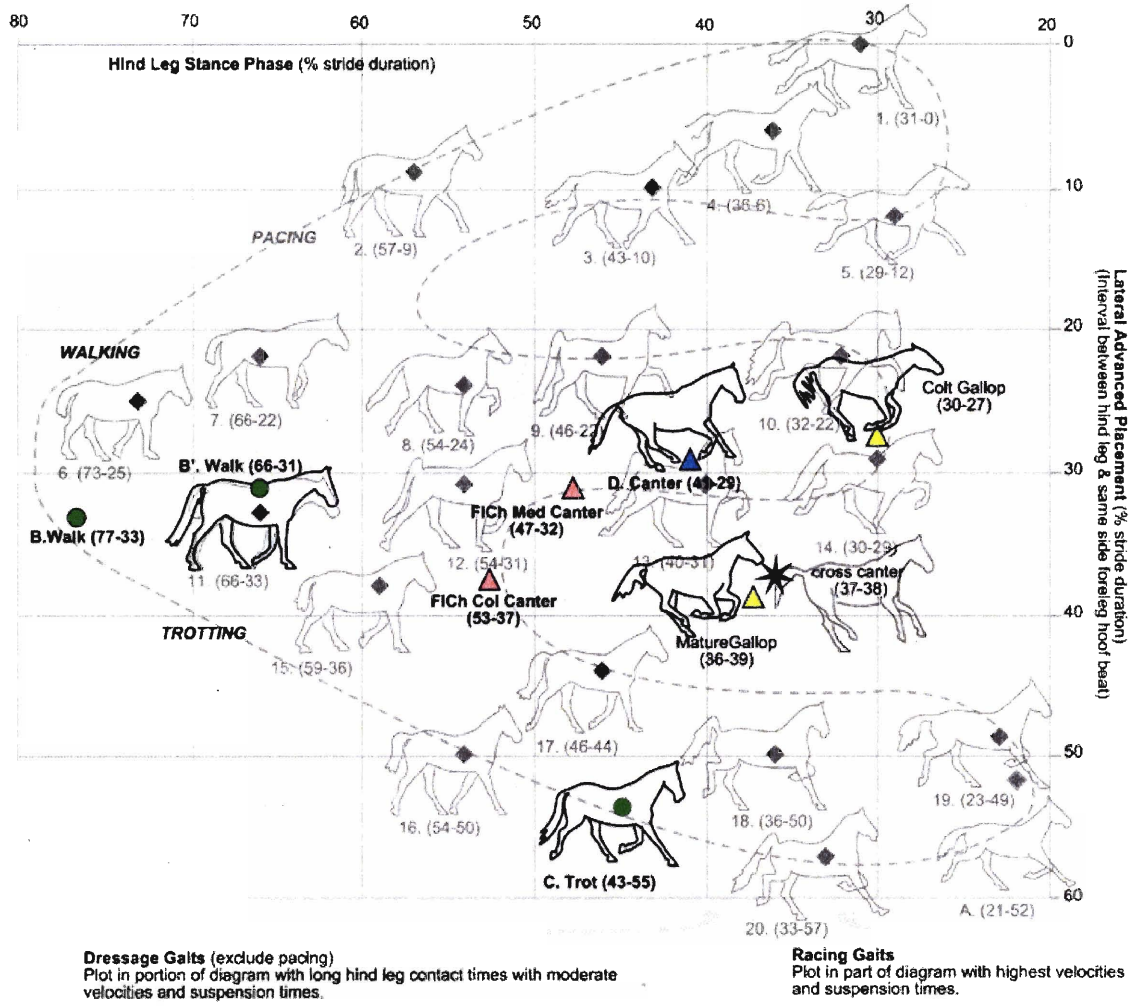

Figure 2.2: Plot of Leg Phasing vs. Ground Time Fraction for Different Gaits [46]

However, while the overall speed of an animal is linked to the average number of legs in return over a full step cycle, gaits are defined by the phasing differences between the animal's footfalls. This is why plodding trots can have fewer legs in return and be slower than quick walks and why racing trots, which have a long flight

phase, can be faster than some gallops. Varying the ground contact time of the legs during the stride can vary the animal's forward speed by a significant amount. This can be seen in Figure 2.2, which plots different horse gaits based on the percentage of the time the hind legs are on the ground and the phase interval between the same side front and back legs. The three lobes of the graph correspond to three different gaits: paces, canters, and trots; but within those lobes the suspension time, and hence the overall speed of the horse, varies widely.

So why do animal's change gaits at all? A trot could cover pretty much the entire velocity range of a quadruped without the necessity of shifting between gaits. The reason lies in energy consumption. As Figure 2.3 shows, different gaits are most efficient at different speeds, so while an animal could use a trot for everything, it would waste a tremendous amount of energy to do so.

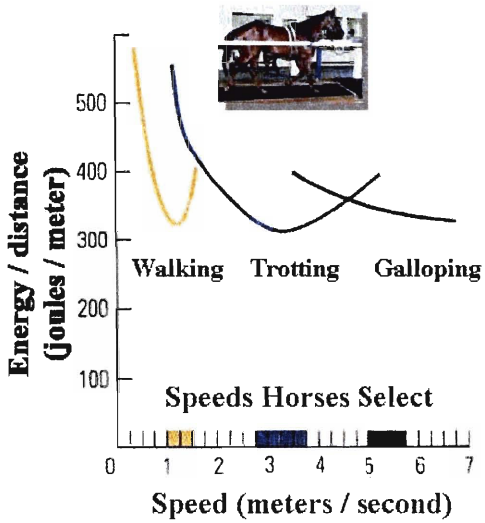

Figure 2.3: Efficiency vs. Speed for Typical Horse Gaits [6]

The transition between gaits occurs where the efficiency lines overlap. These transition speeds can actually be predicted fairly accurately for a wide variety of animals using a dimensionless quantity known as the Froude number [38]. The Froude number F is given by

$$F = \frac{v^2}{gl} \tag{2.1}$$

where v is the velocity of locomotion, g is gravitational acceleration, and l is a characteristic length, usually taken to be the length from the animal's hip to the ground while standing. The Froude number is a dimensionless speed that is derived from the movement of a mass through an arc. Dynamically similar behaviors have similar Froude numbers, which is why a horse transitions between a walk and a trot at nearly the same Froude number as a dog does [12]. At Froude numbers greater than one, the force required to make the animal pivot in an arc over a planted foot exceeds the force of gravity, thus requiring it to leave the ground and run [26]. Generally however, animals switch from a walk to a trot at a Froude number of 0.5 and begin galloping at a Froude number of 2.5 [6].

When moving, animals tend to start with a walk and transition to a trot, then a canter, and finally a gallop as they increase their speed. The focus of this paper is on developing and testing trotting, cantering, and galloping gaits for a quadruped robot, so these gaits are explained in more detail in the following sections.

2.1.1 Trot

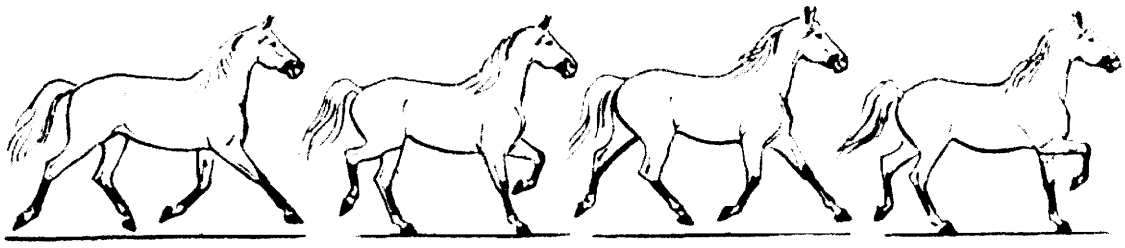

Figure 2.4: Trotting Horse [15]

The trot is a symmetrical dynamic gait that is primarily used for medium speeds. When quadrupeds increase their speed from a walk, the trot is generally the first gait they employ. For the trot, diagonal pairs of legs move in concert, touching the ground

and lifting off together. The phasing for each leg, normalized for stride period, can be seen in Figure 2.5. In this figure the left front leg is used as the zero leg, meaning that all the phases are calculated from the time that the left front foot hits the ground. For instance, if a trot had a stride period of one second then the left front foot and right hind foot would both land at time zero, while the right front and left hind foot would land 0.5 seconds later. If the stride period was 4 seconds then the right front and left hind feet would land 2 seconds after the left front and right hind feet. This method of normalizing the leg phasing for stride period is used for all subsequent leg phasing figures as well.

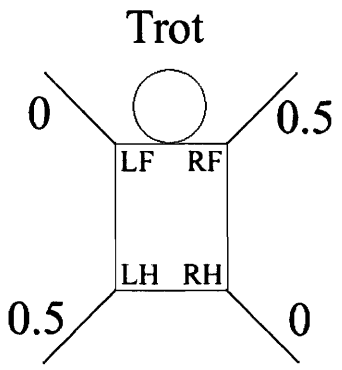

Figure 2.5: Leg Phasing for a Trot

In the traditional trot, each leg stays on the ground half the time and is in the air half the time, however the ground contact percentage can vary from much greater than 50% in plodding trots to significantly less than 50% in racing trots. The trot is more stable than most of the other dynamic gaits since it has two legs grounded at the same time and the line of support between the grounded legs passes diagonally under the animal's body [55]. The trot is actually the fastest gait for some reptiles and amphibians because their short stance height in comparison to their body length and the fact that their spines bend side to side more easily than up and down prevent them from effectively employing faster gaits such as canters and gallops [63].

2.1.2 Canter

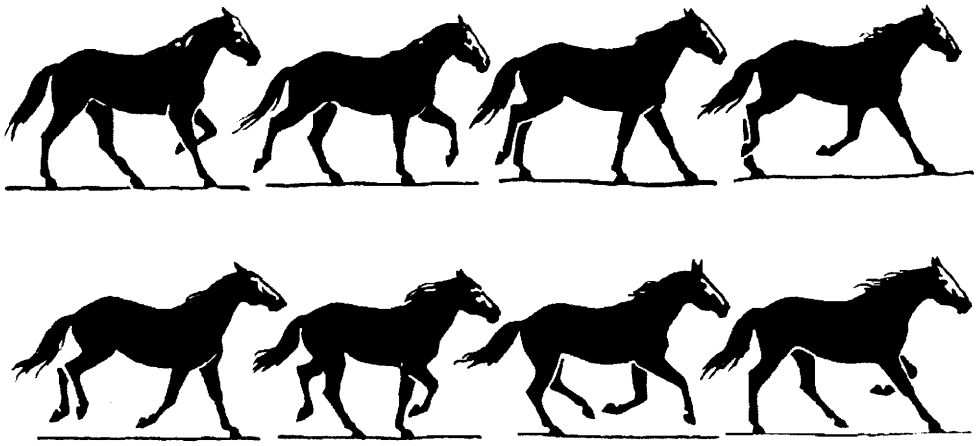

Figure 2.6: Horse in a Slow Left Canter [15]

The canter is an intermediate gait used between a trot and a gallop. It is one of the smoothest of all the gaits and is used to cover moderately long distances efficiently at relatively high speeds. Canters contain elements of both the trot and the gallop, as one would expect from a transition gait. One diagonal pair of legs operates together, like the trot, while the other two legs operate separately, like a gallop. In the canter, the diagonal pair of legs is grounded first. Following the diagonal leg's support phase the individually moving front foot touches down and there may be a period of flight after this foot is lifted up. Then the individually moving hind foot touches down and there is generally no period of flight before the diagonal pair of legs is grounded again. A slow canter with no flight period is shown in Figure 2.6. This is a very stable gait because there are periods where three legs are on the ground creating a support triangle, as in a walk, and periods where two diagonal legs are on the ground creating a support line under the body, as in a trot. However, there are also phases with only one or zero legs on the ground, which makes this a fast gait due to the high average number of legs in return.

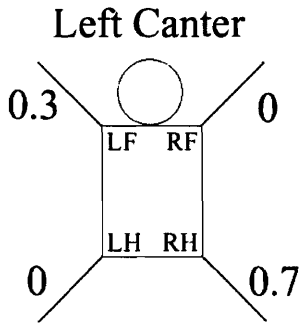

Figure 2.7: Leg Phasing for a Left Canter [25]

There are two main forms of the canter: the left canter and the right canter. The difference between the two is merely which diagonal pair of legs operates in tandem and different animals prefer to use the right or left canters much in the same way that people are left or right handed. The leg phasing for a left canter is shown in Figure 2.7. There are also other forms of the canter, such as the cross canter, but they are not very widely used and are not considered "true" canters because their leg phasing is different the traditional canter.

2.1.3 Gallop

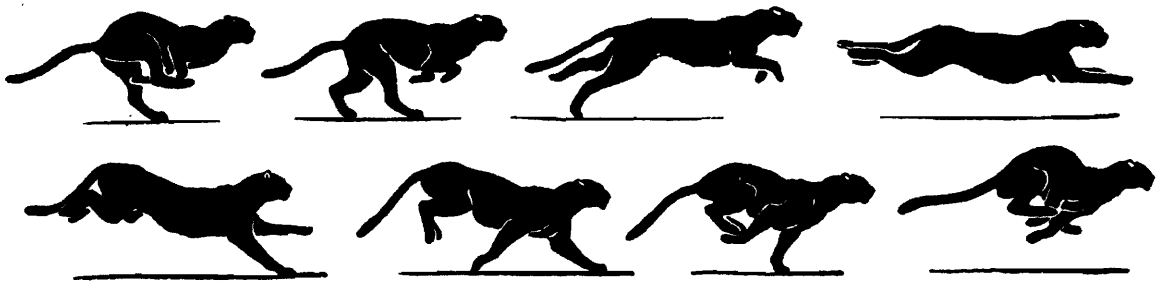

Figure 2.8: Cheetah in a Rotary Gallop [15]

The gallop is a high speed gait that is commonly used by almost all biological quadrupeds, from dogs to horses. It is characterized by the two front feet hitting the ground sequentially followed by the two back feet, which also strike the ground

sequentially. There can be one or two flight phases during the gallop depending on the speed with which the animal is moving. The first flight phase, which is always present, occurs after the front legs lift off the ground and before the hind legs have touched down. This flight phase is called "gathered" because all of the legs are tucked beneath the body. The second flight phase occurs at high speeds and is generally shorter than the gathered phase. It occurs after the rear legs have lifted off and is known as "extended" because all the legs are stretched out forwards and backwards during this phase [15].

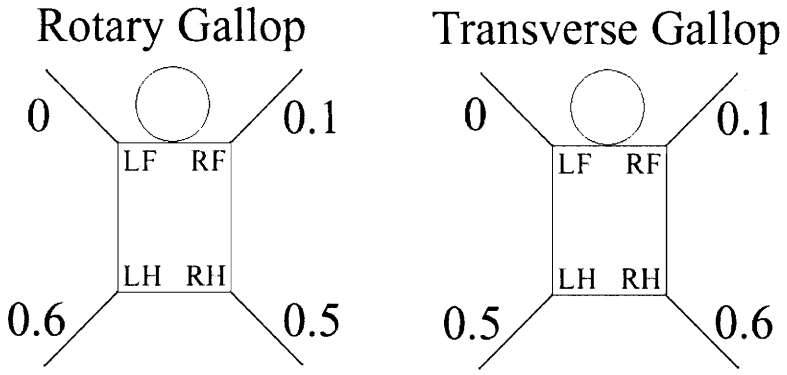

Figure 2.9: Leg Phasing for Rotary and Transverse Gallops [25]

There are two main types of the gallop: the transverse gallop and the rotary gallop. In the transverse gallop, the legs on the same side of the body are the first to touch down when both the front and rear pairs of legs hit the ground. In this gallop the body undergoes two rolls per stride due to the fact that both the fore and the rear legs cause a roll to the same side when they land. In the rotary gallop, the opposite foot strikes the ground first in the front and hind legs pairs, i.e., if the left leg hits the ground first when the front legs touch down, the right leg will hit the ground first when the rear legs touch down. This means that the body only undergoes one roll per stride since the first pair of legs causes a roll to one side and the next pair corrects the roll.

The two gallops seem to be preferred fairly equally among animals, but there is some research that indicates the transverse gallop might be favored more by animals with a stiff spine while the rotary gallop is favored more by animals with a flexible

spine. The rotary gallop also appears to allow for two flight phases to be achieved more easily than the transverse gallop, while the transverse gallop appears to be more stable overall [55]. This means that the rotary gallop is better for achieving high top speeds, while the transverse gallop is better for stable running at lower speeds. The transverse gallop is also less likely to cause leg interference than the rotary gallop since the phase difference between the front and the rear legs is greater [18].

2.2 Current Research on Gaits on Robotics

2.2.1 Gait Research on Aibos

There has actually been a fairly significant amount of research done on trying to make Aibo robots move faster than their standard walk speed. This research has primarily been driven by the RoboCup robotic soccer tournament in which teams of various types of robots play a soccer game against each other. One of the divisions in the competition is an all Aibo division and since all the robots being used are the same, the competitors can only beat each other by having better control software than their opponents. Obviously teaching one's Aibos to run faster than everyone else's would give you a sizable advantage in a game where getting to the ball first is an important factor in winning.

In the beginning, most speed improvements to the Aibos were made by creating a set of parameters to define a gait and then hand-tuning them in order to achieve the best possible speed. As time went on however, researchers discovered that by using their hand-tuned gaits as a base parameter set and then applying a learning algorithm of some sort they could develop much faster gaits than were previously possible.

One of the first attempts to use an evolutionary algorithm to learn a gait for an Aibo was performed by Hornby *et al.* [24] while helping Sony to try to develop a trotting gait for use in the first consumer version of the Aibo in 1999. They used a

steady-state evolutionary algorithm in their trials and were moderately successful. A steady-state evolutionary algorithm works by choosing one of two operators, mutation or combination, and applying them on the current set of gaits to obtain the new generation. If mutation is chosen, the algorithm randomly chooses two individuals from the set of gaits, mutates the values of the one with the highest score, and replaces the lower scoring gait with the new mutated gait. If combination is chosen, the algorithm randomly picks three individuals, combines pieces from the two with the top score, and replaces the lowest scoring individual with the new gait. This type of evolutionary algorithm keeps a large number of individuals from the previous generation every time it creates a new generation, so it doesn't evolve as fast as some other learning algorithms do. Nevertheless, the researchers were able to develop a pacing gait that moved at 17 cm/s, which was much better than any previous gait that Sony had been able to develop via hand-tuning. A pace is a gait in which the legs on each side of the body move in concert with each other and the left pair of legs has a phase offset of 0.5 from the right pair of legs.

The main benefit to come out of this research however, was the evaluation program that was used to test parameter sets and give fitness scores. The program the researchers developed was almost fully autonomous, thus greatly decreasing the amount of time needed to run a series of trials. For each parameter set, the robot would use its onboard camera to face itself towards a colored strip placed on the opposite wall. It would then load the gait to be tested, travel for an allotted period of time, and see how far it got. Distance traveled was calculated using the robot's onboard distance sensor, which measured the distance to the far wall at the beginning and end of the trial to get the distance covered by the gait. The robot then used its head camera to determine what angle it was at in relation to the color strip once the trial finished and from that calculated how straightly it had traveled. The speed and straightness values were then multiplied to come up with a fitness score for the gait. Finally, the robot turned around to find the colored strip on the wall in the direction it had just come from, centered itself on it, and evaluated the next individual. Versions of this evaluation program were used for most subsequent machine learning gait research using Aibo robots.

Figure 2.10: Aibo Learning Setup [24]

The next major attempt at improving an Aibo's speed was made by the University of South Wales in preparation for the 2003 RoboCup [28]. In this experiment, the leg phasing was fixed and only the leg path was modified. The leg path started out as a simple rectangle and there were twelve parameters that allowed the algorithm to move each of the four corners of the rectangle in three dimensions to create a new leg path. The path for the front legs and the rear legs could be modified separately, which resulted in a total of 24 parameters for the algorithm to optimize. The algorithm they chose to use was Powel's (direction set) method for multidimensional minimization [52]. This method minimizes along each direction in a multidimensional space and notes the effectiveness of each direction. It then uses that information to derive new directions in order to minimize the function as fast as possible.

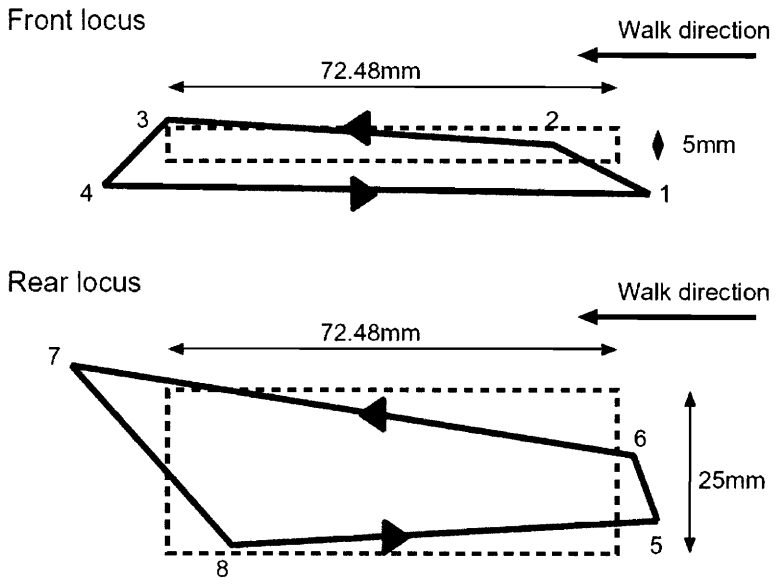

Figure 2.2.11: Evolved Leg paths for the Aibo [28]

This team developed a gait that moved at 27 cm/s, a moderate improvement over the speed of the previous best hand-tuned gait at 25 cm/s and a significant improvement over the speed of the previous best learned gait at 17 cm/s. The UNSW learned gait was also much more stable than any of the previous hand-tuned gaits. The leg trajectories that were developed by the algorithm can be seen in Figure 2.10.

After seeing the success of the UNSW team, other teams began using learning methods to try and optimize their gaits as well. The University of Texas at Austin decided to use a more constrained half-elliptical foot path for their gait, which consisted of the foot stepping forward in a smooth arch and moving backwards in a straight line. This decreased the overall search space for their problem and allowed them to add in parameters that controlled overall stride time and fraction of time on the ground without significantly increasing the convergence time for their algorithm. The algorithm that the UT Austin team chose to use was gradient-climbing reinforcement learning. This algorithm takes a base parameter set and creates variations on it by randomly adding or subtracting a set amount to each parameter. It then evaluates each of the new parameter sets and based on that information adjusts the base parameter set to move it along the perceived steepest upward gradient of the

multidimensional search space. Using this method, UT Austin was able to find a stable gait with a speed of 29.1 cm/s [30].

Carnegie Mellon University also decided to see if it could use machine learning to improve the speed of its Aibos for the RoboCup competition [7]. However, rather than specify their control parameters in terms of the leg trajectories, they decided to focus on controlling the trajectory of the robot's body. They did this by simply specifying the place that each foot would be set down and then saying that once the foot was on the ground it had to pass through its "neutral position" at a certain time that was calculated to maintain the current trajectory of the body. The set down coordinates, neutral position coordinates, foot velocities, stride times, and leg phasing, were all part of the parameters that could be set. The CMU team decided to use a type of genetic algorithm to learn the parameters for the fastest gait. Their genetic algorithm worked somewhat like the evolutionary algorithm used by Hornby *et al.* in the first attempt at applying learning to the problem of developing gaits for the Aibo [24]. It started with a set of individual gaits and then applied one of two operators to them: mutation or crossover. Mutation took a single individual and modified it to create a new individual, while crossover took two individuals, split them along a random point and crossed the pieces to create two new individuals. These operations were performed until an entirely new generation was made, but the previous generation was still kept as well. The gaits from the new generation were then evaluated and both the new generation and the old one were combined into one large list. The worst M individuals from the list were removed in order to return the list to the specified generation size and the process was then repeated. Another feature of the CMU algorithm was "radiation". Whenever too many individuals were clustered in a tight region, radiation was applied that greatly increased the mutation and crossover rates, spreading the individuals out and widening the search area. Radiation was used in the first phase of the algorithm to make sure the search covered a wide range of possible gaits, and then was turned off for the second phase so that the algorithm could focus on improving the best gait it had found during its search in phase one. Using this method, the CMU team was able to find a gait with a speed of 29.6 cm/s.

The gaits found by the CMU and UT Austin teams were impressive in comparison to the previous speeds that the Aibo was capable of, but were decidedly lackluster in comparison to those achieved by biological quadrupeds. Using the Froude numbers of 0.5 for a trot and 2.5 for a gallop and with a leg length of 14 cm, theoretically the Aibo should be starting to trot at speeds of 82.9 cm/s and begin galloping at speeds of 185.3 cm/s. Most of the reason that the best Aibo speeds are so low has to do with the design of the Aibo itself; the lack of springs on the legs to absorb impacts, the slow movement and low torque of the motors, the inflexible spine, and the design of the legs prevent it from running full out as an actual animal would. However, a large part of the reason for the slow speeds also stems from the constraints that were placed on the learning algorithms by the researchers. These constraints were added because the researchers emphasized stability over speed and because they were doing their learning on actual Aibo robots rather than in simulation and didn't want to break them. In addition, the RoboCup researchers wanted the Aibo to remain in a crouched position on its front legs while moving since this position allows the Aibo to more easily catch the ball. Unfortunately, it also greatly decreases the stride length and freedom of movement for the front legs resulting in a large decrease in the speed that the Aibo can achieve.

2.2.2 Galloping Robots

One of the few robots actually capable of bounding and galloping is the Scout II robot. It uses an extremely simple design with two degrees of freedom per leg: an actuated pivot at the shoulder and a passive linear spring damper in line with the leg. In addition, its controller is only set to read whether the foot is in the air or on the ground. Yet using simple open loop control systems, researchers have been able to get the robot to bound and even implemented a rotary gallop achieving speeds as high as 1.3 m/s [51]. When the foot is in the air, the controller moves the leg to a specified touchdown angle and holds it there until the foot hits the ground. Once the foot is on

the ground, the controller moves the foot backward at maximum torque until it passes a specified take off angle. The controller does not try to stop the leg at this takeoff angle, it just stops applying torque and lets the leg coast backwards until the sensors tell it that the foot is no longer on the ground, at which point the controller moves the leg back to the touchdown angle. The coasting phase during the backward sweep of the leg is important because if the controller tried to stop the leg at the predetermined takeoff angle it would slow the leg down as it approached the angle, thus causing the leg to either slow the robot's forward motion or making the foot drag along the ground, both of which could lead to instability.

Figure 2.12: Scout II Robot [51]

The Scout II is a physical implementation of the Spring Loaded Inverted Pendulum (SLIP) model for legs in a dynamic gait. Seyfarth *et al.* [57], and Chigliazza *et al.* [8], have found that for some leg touchdown angles SLIPs become self-stabilized provided that leg spring stiffness is properly adjusted and a minimum running speed is exceeded. In this case, stabilized means that if the robot's running speed is increased or decreased due to an outside force or small ground variation it will return to the set speed automatically without any intervention from the controller. The higher the running speed, the less sensitive the system becomes to perturbations. Unfortunately, as of yet, the precise reason for this self-stabilizing behavior is not well understood.

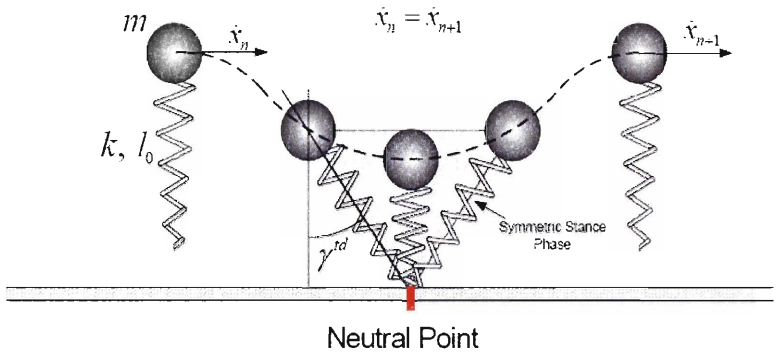

Figure 2.13: SLIP Leg Model [51]

Other work has been done on galloping robots using two-dimensional simulations [44, 31, 33, 38]. One of the most interesting of these studies was done by Krasny and Orin [34]. It is one of the only experiments to use both an asymmetric mass distribution and articulated legs in its robotic model as well as using an evolutionary algorithm to help develop the gait controller. The robot model had perfect actuators at the shoulder and elbow of each leg and the elbow joint had a torsional spring connecting the actuator to the joint so that the leg could bend and absorb the impact energy when the foot touched down. The controller for this simulator worked much like the previously discussed gallop controller [51]. While in flight, the leg would be moved to a specified touchdown angle and when on the ground the shoulder actuator would push the leg backwards at a constant set torque until the leg left the ground. When the foot passed beneath the shoulder the elbow actuator would move a specified amount to add back the energy to the spring that was dissipated due to impact losses with the ground. One of the more unique features of the controller was that the leg actually started moving backwards from the set touchdown angle when the foot was still 5 cm off the ground. This was done so that the leg would be traveling backwards when it struck the ground, thus helping to minimize speed losses due to inadvertent breaking by the impact of the legs.

Using this simulation, Krasny and Orin were able to implement bounding, cantering, and galloping on their robot model. Most impressively, the speeds achieved and energy per distance traveled curves closely resembled what one would expect from biological quadrupeds of similar size. The results are tempered by the fact that

the simulation was two-dimensional, the actuators and sensor data were perfect, and the simulation was initiated with the robot already moving forward in mid-flight rather than having it start from a standing position. Nevertheless, this was one of the few experiments to achieve results that mirrored those seen in real animals.

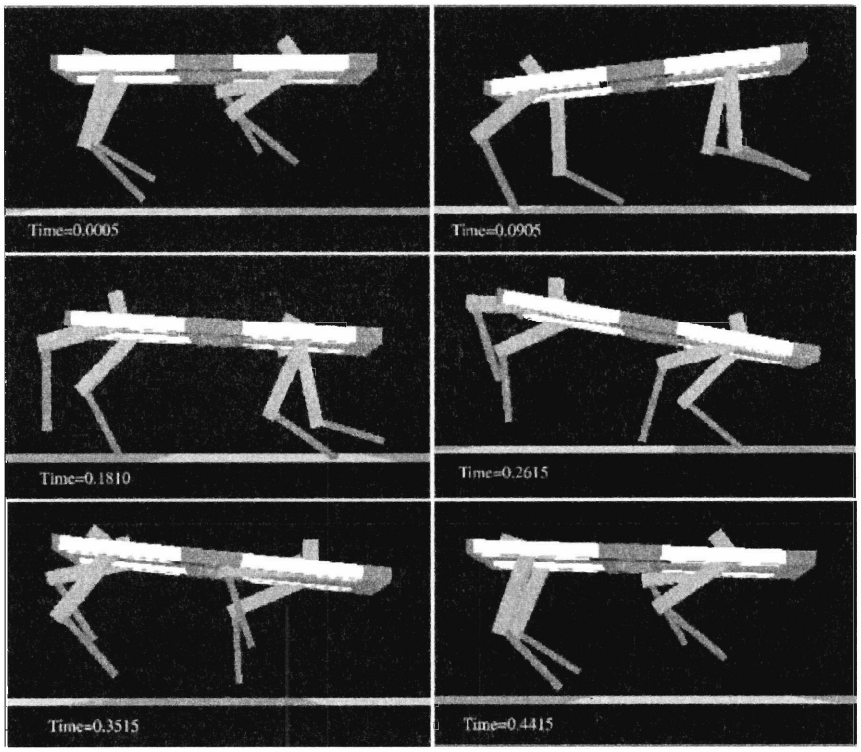

Figure 2.14: Krasny and Orin's Simulated Gallop [34]

Chapter 3 Algorithms

3.1 Simulator

For this research it was decided to use an Aibo ERS-7 robot. The Aibo was chosen because it is one of the most advanced legged robots commercially available and there has been a considerable amount of previous gait research done on it. In addition, most experiments on dynamic gaits have relied on simulations with perfect sensors and other unrealistic assumptions that I wanted to avoid by using a much more realistic model of a currently existing robot. It was decided to use a simulator rather than the actual Aibo robot for this thesis because a simulator allowed for much more controllable conditions than would be possible with real world testing and, more importantly, enabled the learning algorithms to try a wide range of gaits without fear of breaking the robot.

Yobotics Simulation Construction Set was the program that was chosen to create the simulation because it both extremely easy to use and very flexible in terms of how the controller interfaces with the simulator. Yobotics is a fully three-dimensional simulation that automatically calculates the equations of motion and force interactions for any object you create in the simulation. Thus, it allows the researcher to focus on the design of the robot and its controller rather than wasting time concentrating on physics and transformation matrices. The program can quickly and accurately simulate rigid body physics and allows the user to define ground contour and contact models as well as add any obstacles that they desire, such as ramps, gaps, and poles. It also allows access to all joint positions, velocities, and torques and can plot real-time graphs of any variable. Yobotics generates 3D graphics of ones simulation as it runs and gives the user texture mapping and camera controls. The simulations can be stopped, recorded, or rewound at any time and the parameters of the simulation can be modified while the simulation is running. The simulations can also be expanded using the Java-based Application Programmers Interface (API).

To create a model in Yobotics, one merely enters the masses and inertias of the various links that make up the robot and then specifies where the links connect and what kind of joint is at the connection. Each degree of freedom in a joint automatically has an actuator associated with it and these actuators can either be controlled and used as simulated servos or left as uncontrolled free-swinging joints. Ground contact points that will interact with the chosen ground contact model can be added anywhere on the robot as needed.

The measurements of the links and the sweep ranges of the joints for the simulated Aibo were obtained from the Sony manual on model information for the ERS-7 [61]. The masses and inertias of the links were estimated from measurements done on actual Aibo robots, but were somewhat limited in accuracy due to the fact that the Aibo could not be taken apart and the pieces weighed separately. The torque limits for the motors were also estimated at 0.7 Nm. One ground contact point was placed on each foot to simulate the fact that the Aibo stands on spheres rather than using flat rectangular feet.

Link	Masses (kg)	I_{xx} (kg m^2)	I_{yy} (kg m^2)	I_{zz} (kg m^2)
Body	1	0.00333	0.00158	0.00333
Hip	0.0085	1.25E-06	1.25E-06	1.25E-06
Upper Leg	0.04165	3.08E-05	3.08E-05	9.00E-06
Lower Leg	0.04165	2.35E-05	2.35E-05	1.10E-05
Neck	0.0272	1.88E-06	1.88E-06	1.88E-06
Head	0.1088	9.02E-05	5.06E-05	9.02E-05
Foot	0.0085	5.00E-07	5.00E-07	5.00E-07

Table 3.1: Simulation Masses and Inertias

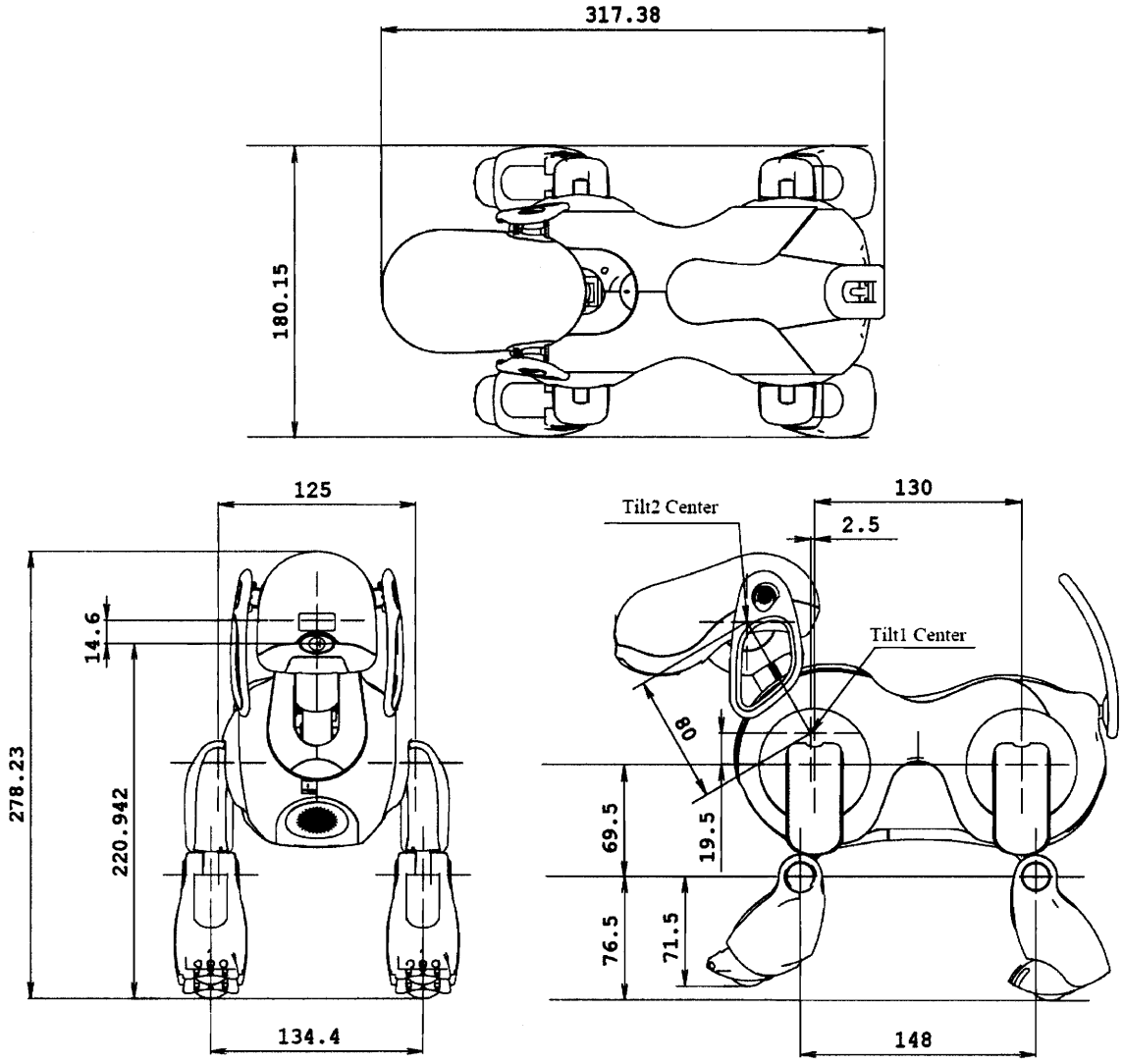

Figure 3.1: Aibo Robot Specifications [61]

The simulation is set up so that the user can set the number of individuals tested per generation and the total number of generations for the learning program to run. The user can also set the length of time for each run; in the case of this research it was set to twelve seconds. Once the time limit for the run is reached or if the robot falls over prior to the time expiring, the program resets and tests the next individual.

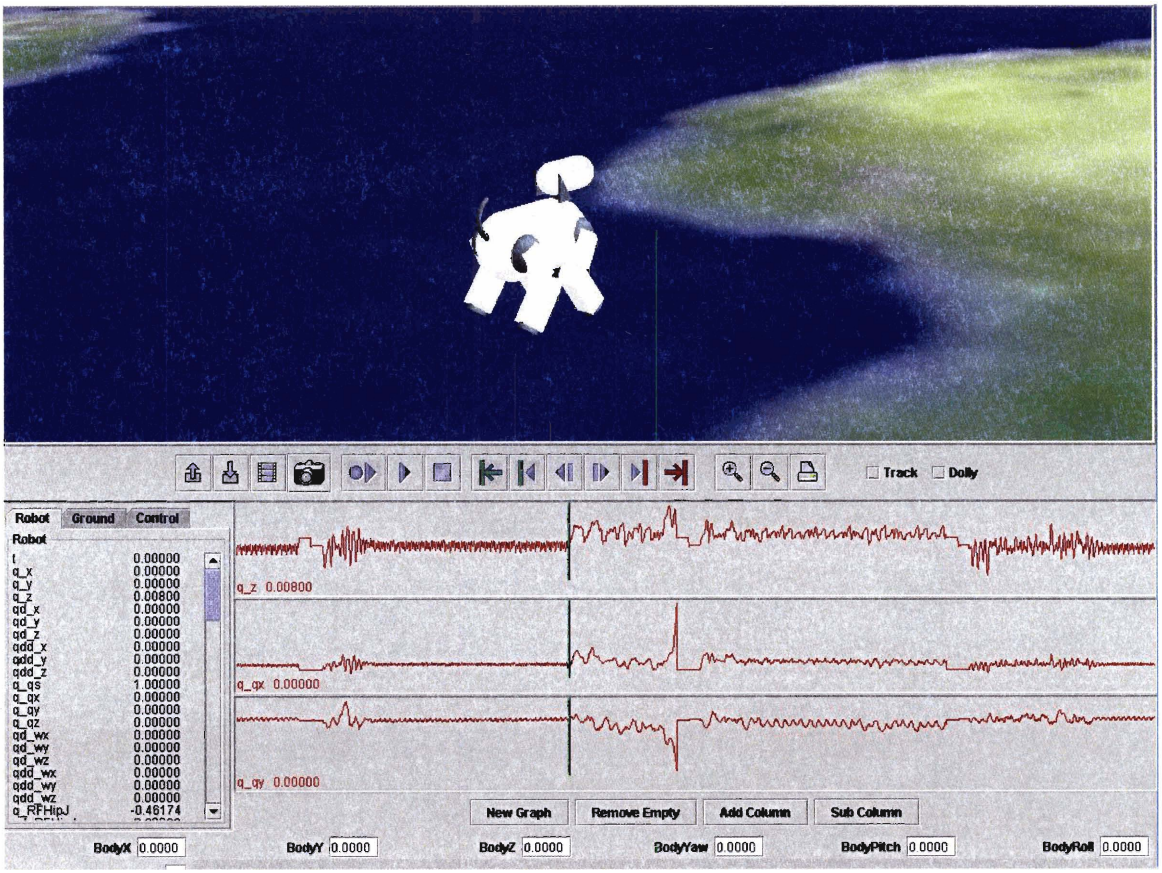

Figure 3.2: Yobotics Simulation

3.2 Gait Program

The gait program takes the 17 parameters shown in Table 3.2 and turns them into a working gait. These parameters set the variables for the basic pieces that make up a gait: the foot path, the leg phasing, the stride time, and the fraction of time spent in each portion of the foot path.

Parameter	Range	Units
Step Height (Front)	1 - 4.465	cm
Step Height (Rear)	1 - 4.465	cm
Cutoff Angle (Front)	0 - 49.5	degrees
Cutoff Angle (Rear)	0 - 49.5	degrees
Cutoff Percentage (Front)	0 - 99	percent
Cutoff Percentage (Rear)	0 - 99	percent
Stride Period	0.2 - 0.8	seconds
Ground Fraction	0.1 - 0.8425	fraction
Phase (Leg 1)	± 0.1	fraction
Phase (Leg 2)	± 0.1	fraction
Phase (Leg 3)	± 0.1	fraction
Y-Coordinate (Front)	0 - 9.9	cm
Y-Coordinate (Rear)	0 - 9.9	cm
Step Length	4 - 19.82	cm
Body Height (Front)	10 - 13.89	cm
Body Height (Rear)	10 - 13.89	cm
Stance Width	± 2	cm

Table 3.2: Gait Parameters

Choosing how the foot path is set up is a fairly important factor in determining how successful the gaits your program can generate will be. Various teams have used various types of foot paths from trapezoids [28] to semi-ellipses [30]. I choose to use a foot path that was a modification of a semi-ellipse. A normal semi-ellipse foot path uses an inverted parabolic trajectory when moving the foot forward and then uses a linear trajectory to move the foot straight backwards. The modified foot path used in this research is shown in Figure 3.3.

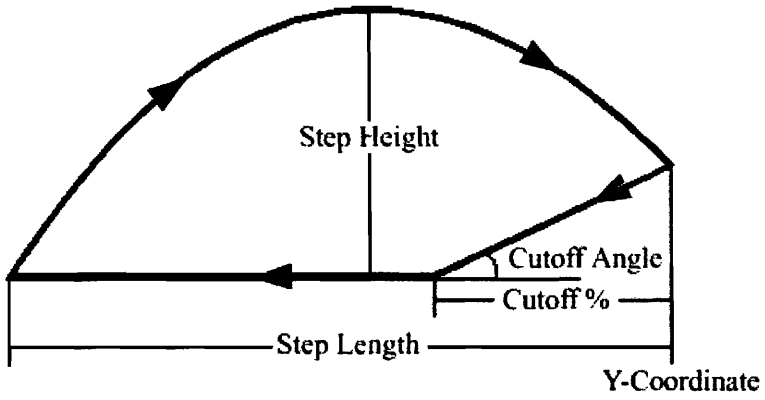

Figure 3.3: Foot Path

As can be seen from Figure 3.3, the modified semi-elliptical foot path has three sections rather than two. The first part of the path is the "step" section, which is an inverted parabolic trajectory, this is followed by the "early retraction" section, where the foot moves back and down along a linear trajectory, and last is the "thrust" section, where the leg moves straight back along a linear trajectory.

The path is traced out using two control primitives written for the Aibo by Greg Andrews called LegLinear and LegQuadratic. LegLinear moves the foot from its current position along a linear trajectory to the specified final position at a constant speed. The parameters for this primitive are the x, y, and z coordinates of the commanded position relative to the hip of the leg being moved, the speed of the foot, and a force threshold that wasn't used in this application. LegQuadratic moves the foot in an inverted parabolic trajectory. The parameters for this primitive are the coordinates of the final commanded position relative to the hip, the lift height of the foot at its highest point relative to the starting position of the foot, the speed of the foot, and a force threshold that wasn't used. The set foot speed for LegQuadratic is based off of the linear distance between the initial and final positions of the foot, not on the distance of the parabolic path that the foot actually travels. For instance, if the foot was commanded to step forward four centimeters with a speed of four

centimeters per second, it would always take one second to complete the movement regardless of the specified lift height. The only restriction on LegQuadratic is that the lift height must be higher than the height of the final commanded position of the foot.

The semi-elliptical foot path was modified to include an early retraction stage for two main reasons: to decrease impact losses and to increase stability. Biological quadrupeds commonly begin moving their legs back and down before they actually touch the ground when they are running. This is done so that their feet are moving backwards at the desired running speed prior to touching down, which helps minimize energy losses from the impact and makes the gait more efficient. Early retraction has also been included in some of the more successful robotic gallop studies [34, 32]. This stage also helps increase the stability of the gaits in the simulation. Without an early retraction stage, the end of the step stage made the foot come straight down onto the ground which ended up occasionally causing the leg to punch the ground, throwing the body up slightly so that the foot was in the air for a good proportion of the thrust phase. This led to wobbling and yawing that would increase until the robot fell over. With the early retraction stage however, the foot is always moving down and back at an angle when it hits the ground rather than straight down or down and forward as it would be if the foot struck the ground during the step stage. This decreases the tendency for the robot to push itself into the air when its foot touches down and makes the all the gaits more stable.

The stride period and ground fraction variables are used to calculate the velocities of the primitives for each stage of the foot path. Stride period is the total amount of time, in seconds, it takes for the foot to traverse the entire foot path and ground fraction is the fraction of the stride period that the foot is not in the step stage. So, if the stride period was one second and the ground fraction was 0.6, the step stage would last for 0.4 seconds and the total time for both the early retraction and the thrust stages would be 0.6 seconds. This means that the ground fraction variable is not the actual fraction of time the foot is on the ground since the foot touches down part way through the early retraction stage. The true fraction of the stride period that the foot is on the ground is therefore always somewhat less than the ground faction variable.

When the simulation is initialized, the robot starts out in a semi-crouched position as is shown in Figure 3.2. After one second has passed so that the robot and its actuators can settle properly, the gait program starts, and the robot begins moving. The left front leg is the leg that all of the other leg phase shifts are keyed off of, so its phase is always zero and it always begins each run by stepping forward. The other legs begin by moving backwards and then lift off and step forward at the appropriately calculated time so that when they begin the early retraction stage they have the correct phasing with respect to the left front leg. The set phase differences are maintained by varying the velocity of the legs during the step stage to correct for any errors that have occurred. This ensures that the legs remain phase locked throughout the run.

To make the simulation more realistic, the controller was run with an update rate of 32 ms, which was the actual update rate of the controller on the real Aibo. This update rate greatly complicated the implementation of the controller since the fastest gaits generally had stride periods of 0.2 seconds, which meant that the controller was only updating one to three times per stage as the robot's legs traveled along the commanded foot path. This introduced errors into the gait, which is the reason that the velocities of the legs had to be adjusted during the step stage to maintain the proper phasing. It also meant that a gait wouldn't run perfectly every time; slight variations in the point during the stride at which the updates occurred combined with the natural oscillations of the robot's body as it ran could cause a leg to hit the ground at different times than it should have and could result in it falling over. The randomness in the simulation, which appeared to be caused by a combination of the slow update rate and the interaction of the robot with the ground contact model as it ran, meant that no two runs were ever exactly the same. These variations in the runs had to be taken into consideration when designing the learning programs used in my research.

Chapter 4 Genetic Algorithm

4.1 Overview

Genetic Algorithms (GA) are a powerful yet simple optimization method that can perform well in high-dimensional, non-linear, and largely unknown parameter search spaces. They are a direct search method that uses principles based upon the Darwinian evolution of DNA, such as selection, crossover, and mutation, to modify individual parameter sets in order to maximize their fitness with respect to some evaluation function over the course of several generations [4].

In the field of robotics, evolutionary algorithms, of which genetic algorithms are a subset, have become a popular tool for developing complex behaviors that would be difficult to program manually. However, much of this research has focused on developing controllers such as neural networks for statically-stable wheeled robots that can adapt to changing environments [11, 47, 65]. Another area of interest has been artificial life, where evolutionary methods are used to develop complex interactive behaviors for multiple virtual or robotic agents in an artificial environment in order to provide insight into the behavior of biological systems [62].

There has been only limited interest in using genetic algorithms to develop dynamically stable gaits for legged robots even though they seem well suited to the task. This might, in part, be due to the fact that there is a significant bootstrapping problem with dynamic legged systems, where only a small fraction of the possible parameter sets lead to actual stable gaits. In most evolutionary learning algorithm studies the researchers start off with a gait or system that is guaranteed to be stable and then search within that stable space for new and interesting solutions to their problem. With dynamic legged systems however, a large portion of the time and effort involved is used just trying to locate an area within the parameter space that is somewhat stable. The difficulty of this problem may have encouraged most

researchers to focus on developing statically stable gaits instead, though there have been a few that have attempted to tackle the problem [51, 34, 32].

4.2 Implementation

The genetic algorithm starts off with a population of N parameter sets that make up the first generation. In the case of this research, N was chosen to be 35 in order to provide an appropriate amount of diversity within each generation so as to improve the quality of the results. The initial generation can either be entirely random or consist of user loaded parameter sets. If the user provides an array of parameter sets that is smaller than the chosen population size, the algorithm fills in the extra slots by breeding together the provided parameter sets until the population is full. All of the parameter sets that are added, whether they are randomly generated or bred from user provided parameter sets, are run through a program to test them for feasibility before they are added to the population. This program just checks to make sure that all the points on the foot path fall within areas that the foot can actually reach, none of the sections of the path require extraordinarily fast or slow speeds, and that the step height is higher than the early retraction stage starting height as well as at least 5 cm below the shoulder height. It also checks to make sure that during the thrust stage the front feet go back to at least the point directly below the shoulders and that the rear feet go back to a point at least one centimeter behind the hips. This condition was found to greatly increase the proportion of stable gaits discovered by the algorithm.

Once the population has been created, the simulator tests each individual and records its fitness score. The fitness score was calculated by taking position checks every tenth of a second, adding up the total distance traveled in centimeters, and subtracting out a minor directionality penalty if necessary. The robot started out facing the positive y-direction, but if the robot got turned around so that it was traveling in the negative y-direction, the linear distance it traveled in the negative y-direction was subtracted from the fitness score as shown in Equations 4.1 and 4.2.

$$fitness = \sum_{k=1}^{n} \sqrt{x_k^2 + y_k^2} \quad \text{if } y \geq 0 \tag{4.1}$$

$$fitness = \sum_{k=1}^{n} \sqrt{x_k^2 + y_k^2} + y_k \quad \text{if } y < 0 \tag{4.2}$$

The reason that the total distance traveled was used as the fitness measure was because maximizing it maximizes both speed and stability as well. If pure speed were used as the fitness function, the program would probably just learn to leap forward at maximum torque and crash on its face since that would achieve the highest speed, therefore some measure of stability needed to be included in the fitness function as well. The faster a gait is, the more distance it will cover in a given amount of time and the more stable the gait is, the longer it will stay upright and consequently the more distance it will cover during the run. A very slow stable gait will get a low fitness score using this system, but so will a very fast unstable gait.

The direction penalty was put in to encourage straighter gaits. The reason that it is such a weak penalty is that for many gaits the Aibo bounced around a lot at the beginning when it started moving and ended up facing in a different direction by the time the gait settled into stability. If gaits were punished because the robot started out facing the wrong direction due to bouncing at the beginning, it would greatly retard the learning process since many good gaits would be unnecessarily punished. In addition, many good gaits had a slight yaw to them so that the robot ended up running in a large circle. This behavior needed to be punished somewhat, but not to the point where the algorithm would not properly learn from a gait just because it ran in a circle, especially since other researchers have found small curves during the course of a gait easy to correct for using yaw control [32]. Gaits that fell over in less than 3 seconds were given a flat fitness score of five.

Once the all the parameter sets in the population had been tested and their fitness scores recorded, likelihoods were calculated for each individual. It was experimentally determined that proportional likelihoods gave the best overall performance for this application. With proportional likelihoods, each individual has a

chance to be selected based on its relative fitness in relation to the summed fitness scores of the entire generation and this fraction is the individual's likelihood score. So the likelihood score of individual j is its relative fitness r_j as given by:

$$r_j = \frac{F_j}{\sum_{k \in D} F_k}, j \in D \tag{4.3}$$

where D is the domain of all the individuals in the generation and F is the fitness score for a given individual.

Once all the fitness scores have been recorded and the likelihoods have been calculated, the algorithm creates the next generation of individuals. To do this, the algorithm first takes the top M individuals with the highest fitness scores and copies them over to the next generation unchanged. This is called elitism and is used to preserve particularly good individuals so that their influence in the gene pool is maintained through the generations. Initially, these elites were retested each generation and the new scores overwrote the older ones, but, as was mentioned in Chapter 2, the simulator had an element of randomness to it which would sometimes make the gaits work very well and sometimes make the robot fall over right away. This meant that the elite's scores did not always stay high when they were retested, which diminished their influence on the gene pool and negated the entire purpose of having elites. This lead to periodic setbacks in the learning process as can be seen in Figure 4.1, which slowed the learning rate down. Thus, the program was changed so that the elites were still retested each generation, but their fitness scores were only changed if they were higher than they were before.

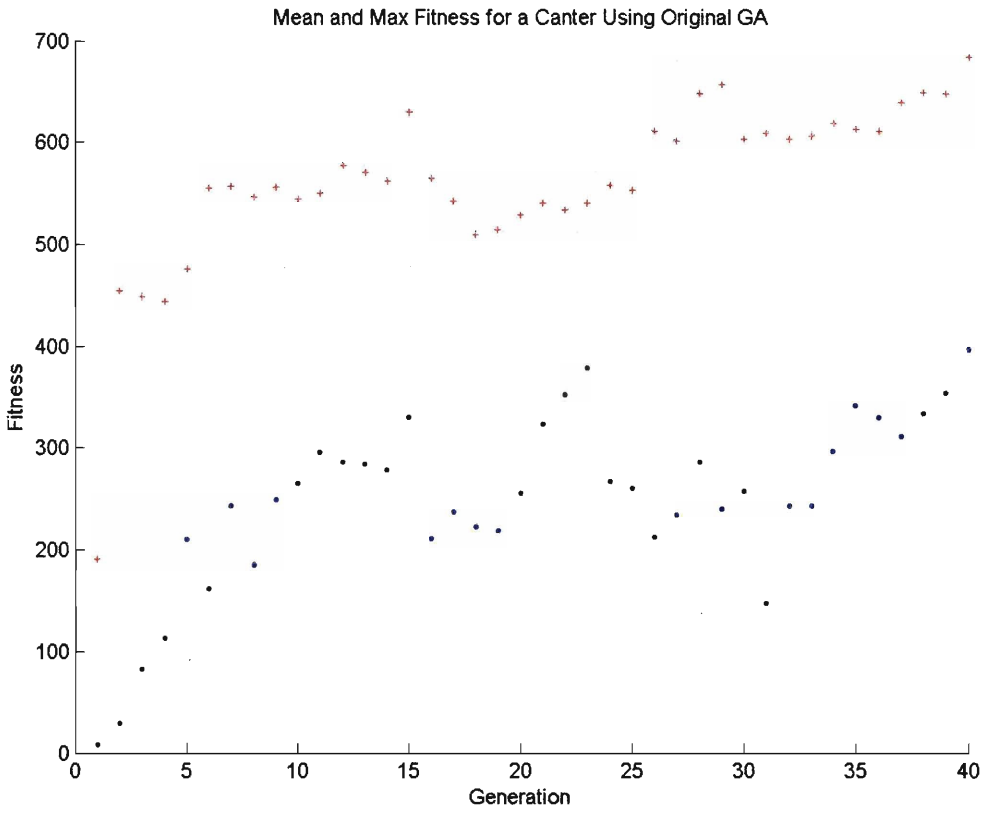

Figure 4.1: Mean and Maximum Fitness Scores for a Run Using the Original GA

After the elites were added, all the rest of the slots in the next generation were filled in by breeding the individuals from the previous generation. Breeding begins by selecting two individuals based on their likelihood scores i.e., a parameter set with a likelihood score of 0.1 would have a 10% chance to be chosen. Once two parents are chosen they are mated together using a simple crossover procedure. During the crossover one of the parents is randomly chosen as Parent 1 and the other is designated Parent 2. A random point is chosen along Parent 1 and from that point onward all the parameters are deleted and replaced with the corresponding parameters from Parent 2, creating a single new individual that is the child of Parent 1 and Parent 2. An example of the crossover operation being performed to create two children can be seen in Figure 4.2.

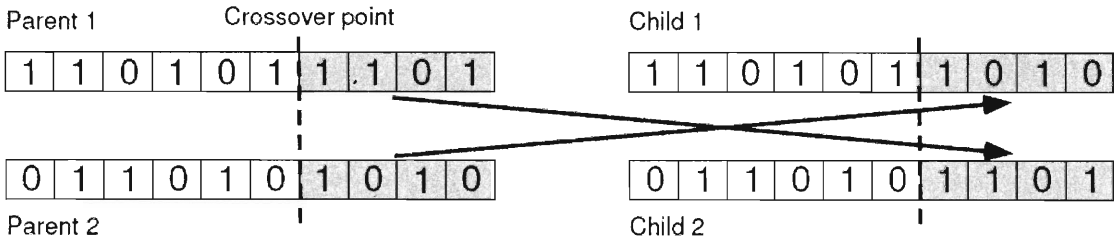

Figure 4.2: Example of Single-Point Crossover [27]

After the new child has been created, a mutation operator is applied to introduce some random variation and prevent the solutions from clustering too much. The amount of mutation that occurs is based on the mutation percentage, which was set to 15%. This meant that for each parameter in the child parameter set, there was a 15% chance that the mutation operator would be applied to it. If the mutation operator is applied to a parameter, the parameter can either be reset to a random number or its current value can be varied by a random amount. In the case of this paper's research, all the parameters were normalized so they could be any number from 0 to 99 and the mutation operator varied the chosen parameter from its current value by some random number between ±10. Having the mutation operator change the parameter to an entirely new number is generally the preferred method because it forces the algorithm to explore a wider search space, but with dynamic gaits the parameters were so sensitive that any large change in their value generally completely destroyed the parameter set's viability. Thus, a small variation of the current parameter value was used instead to provide some randomness to the algorithm without massively inhibiting the learning rate.

Once the child had been bred and mutated, it was checked for its feasibility using the checking program that was detailed earlier. If the child passed the testing it was added to the next generation, if it did not, new parents were chosen and a new child was bred. This process continued until the next generation was filled, at which point the new generation was sent to the simulator for testing so that its fitness scores could be recorded. This process was repeated for 50 generations during each trial and a total of 3 trials were completed for the trot, the canter, and the gallop to get the initial results.

4.3 Initial Results

4.3.1 Trot

The trot is usually the slowest dynamic gait as well as being one of the most stable. For these trials the base phase offsets were set as in Figure 2.5, though the algorithm could modify them by ±0.1 to try out different variations on the trot. The number of elites was set at 5 and the mutation rate was 15%.

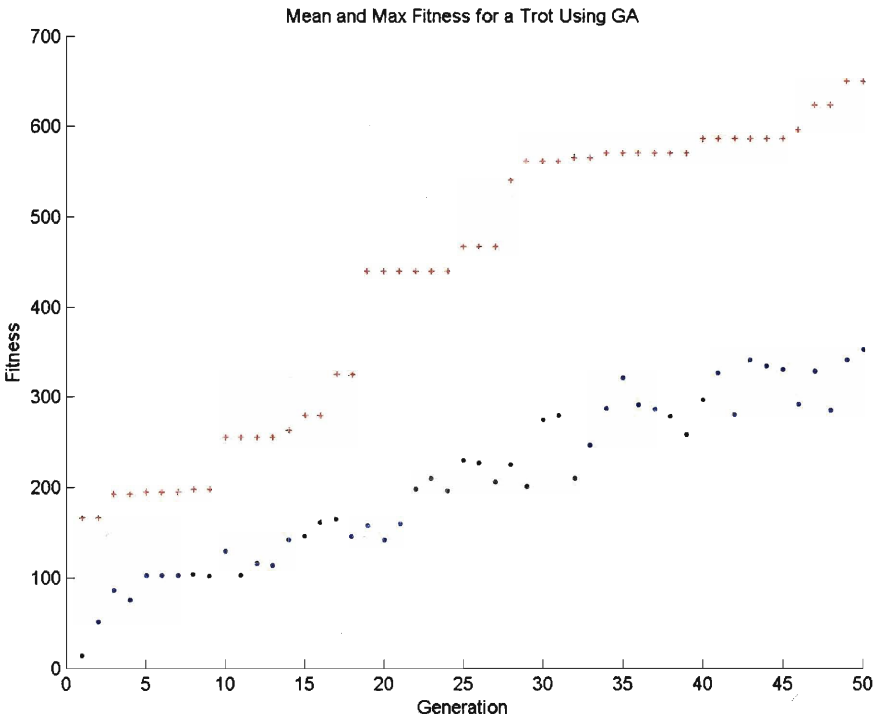

Figure 4.3: Plot of the First Run for the Trot Using the GA

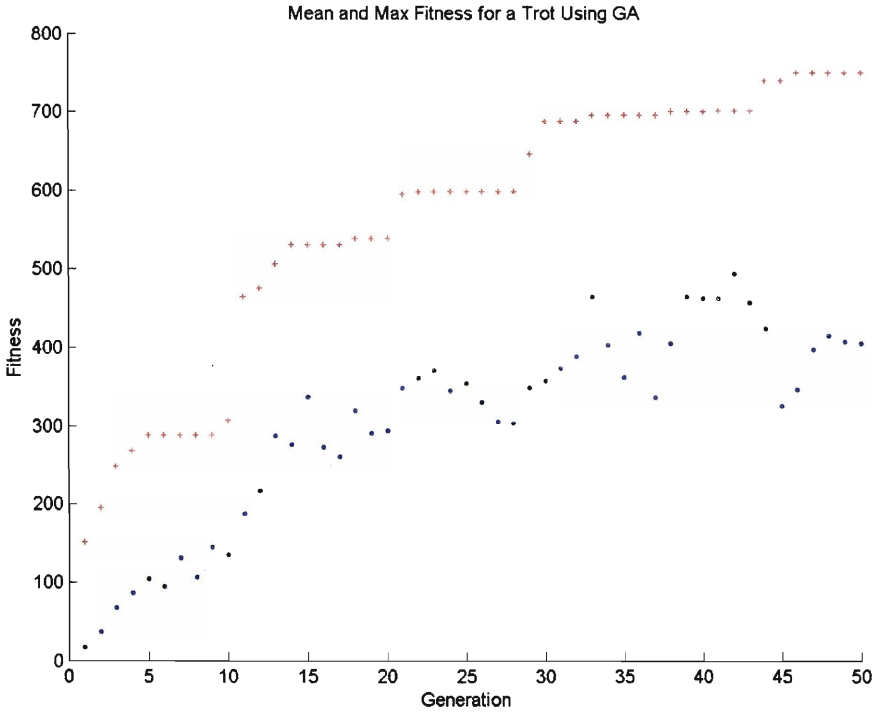

Figure 4.4: Plot of the Second Run for the Trot Using the GA

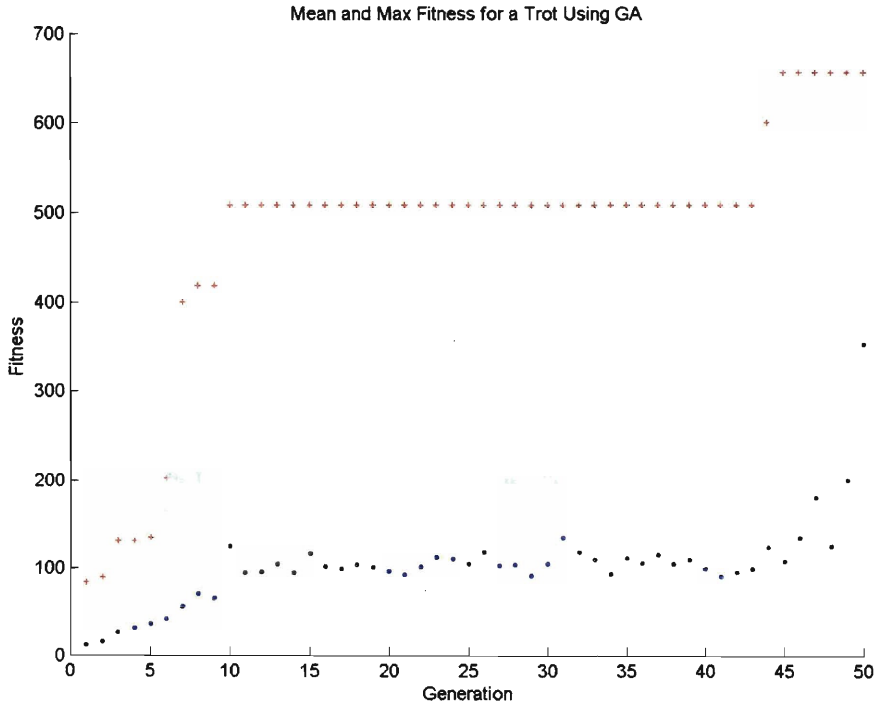

Figure 4.5: Plot of the Third Run for the Trot Using the GA

The first trial shows a slow, almost linear progression in the fitness indicating that there is still considerable room for improvement in the gaits that it found. Normally, one would expect the plot of the fitness scores to flatten out as the gaits reached their maximum values since no further improvements could be made. Since the first trial shows no flattening of the fitness line, these gaits have not yet approached their highest fitness values.

The second trial came closer to maxing out the fitness of its gaits as can be seen from the way that the fitness climbs rapidly and then begins to taper off, but it has not yet flattened out entirely. The reason that trial two climbed to a much higher fitness level than trial one is because in trial two the GA happened upon a region of very stable fast trots. The trots in that area of parameter space seem to have consistently high fitness values and are not nearly as sensitive to varying the parameters as most gaits seem to be. This kept the averages for each generation high meaning that more individuals were contributing to the genetics of the next generation rather than having a few elites dominate everything. Thus, the GA was able to more fully explore this stable region and find better gaits than it would have if it was forced to only search in tight proximity to a few elite gaits as it would in a more unstable region where the average score of each generation was low.

The third trial was interesting because it climbed extremely fast in the beginning, then leveled off for a long time, and finally, made an abrupt jump upward. This happened because the GA became stuck in an unstable region around generation 10. Virtually every variation it tried in this region resulted in extremely poor fitness values as demonstrated by the low fitness averages for each generation. Thus, the elites came to dominate the gene pool more and more, but since the elites were within the unstable region, as their influence on the gene pool grew they locked the GA into that particular area of the search space. Luckily, the mutation factor eventually broke the GA out of that area and allowed it to move to a much more stable one, resulting in a massive jump in maximum fitness and an even larger one in the average fitness. However, the delay in learning caused by becoming stuck indicates that significant improvement can probably still be made to the gaits in this trial.

4.3.2 Canter

The canter is a transition gait between a trot and a gallop. It is generally a fast yet extremely smooth gait with none of the jarring variations in body height that are present in the trot and, to a lesser extent, the gallop. For these trials the base phase offsets were set as in Figure 2.7, though again the algorithm could modify them by ±0.1 to try out different variations on the canter. The number of elites was kept at 5 and the mutation rate was kept at 15%.

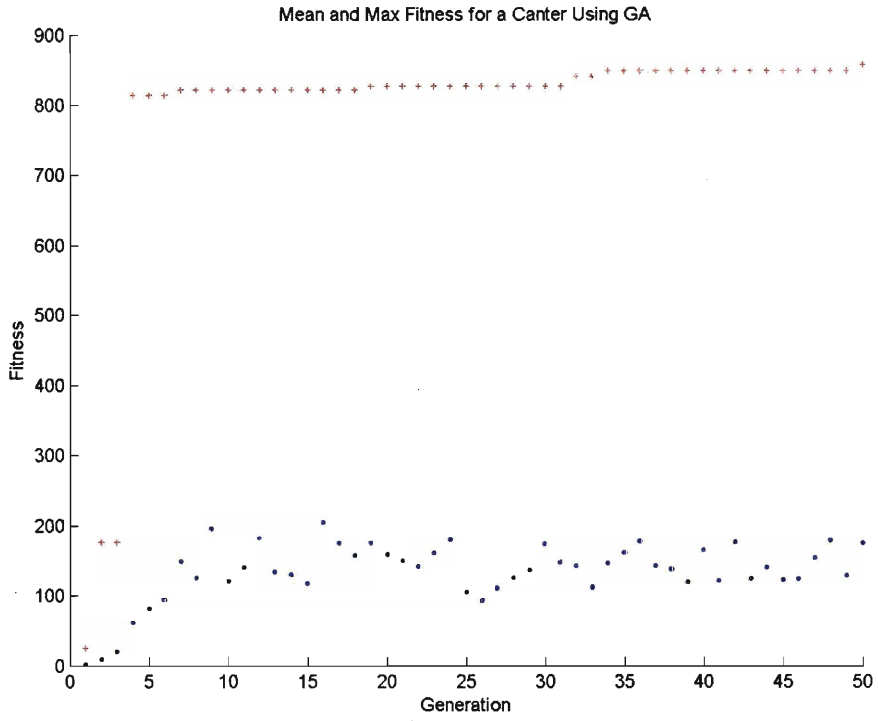

Figure 4.6: Plot of the First Run for the Canter Using the GA

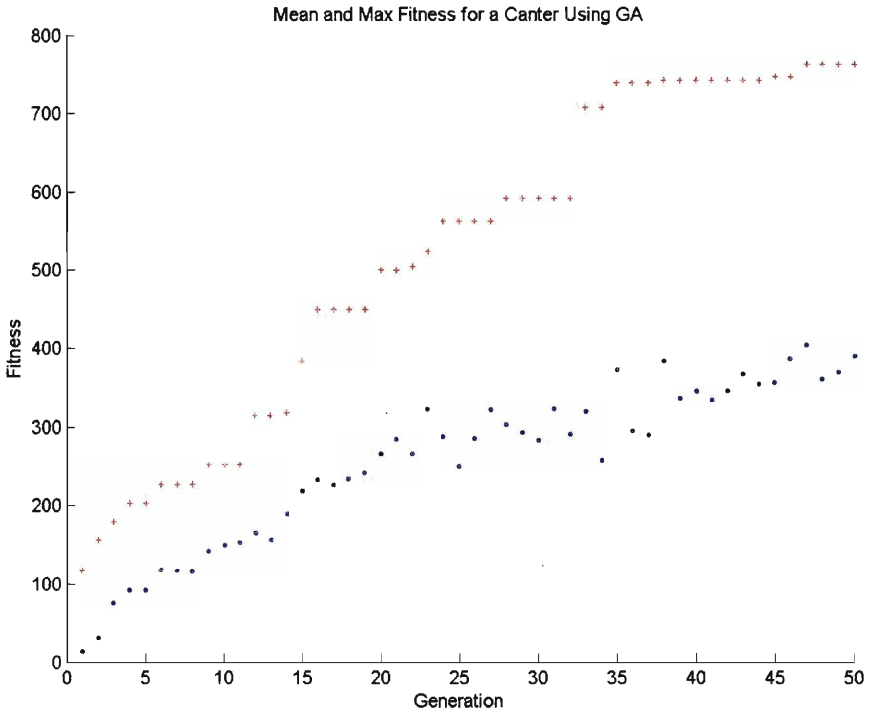

Figure 4.7: Plot of the Second Run for the Canter Using the GA

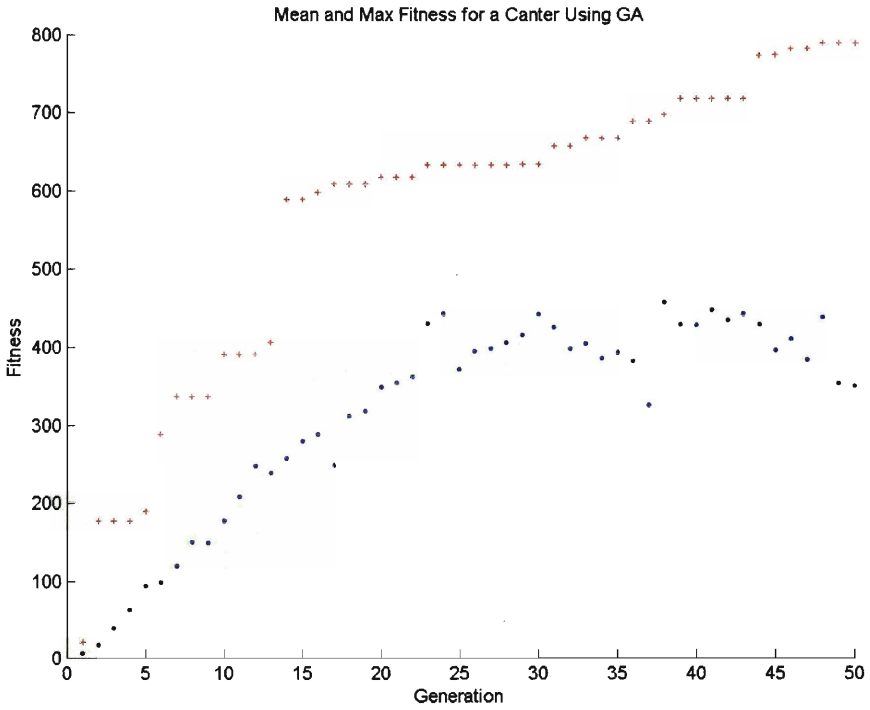

Figure 4.8: Plot of the Third Run for the Canter Using the GA

In the first trial for the canter, the GA became stuck in an unstable region just as in the third trial for the trot. However, this time the fitness climbed even more rapidly and achieved a much higher fitness value. It is obvious that the region is extremely unstable due to the very low average fitness scores for each generation, which led to the GA get locked into that area of the parameter space, but unlike in trial three of the trot, this time the mutation factor was unable to break the GA out of the unstable region. The maximum fitness the GA had found was just so high that mutation could never find anything better and the GA plateaued. The GA was able to make some minor improvements to the fitness as time went on, but it remained in the unstable area for virtually the entirety of this trial and there is probably not much improvement that can be made on these gaits.

Trial two climbed steadily until generation 35 or so and then began leveling off. The gaits it found were all fairly stable and it appears that there is some room for both improving them and finding an even more stable region due to the fact that the maximum fitness is still increasing slightly and the average fitness continues to climb steadily as well. When the GA becomes stuck in a particular area, the average fitness tends to level off and unless the mutation factor can break the GA out of that area the maximum fitness will soon level off as well once the GA finds the best gait within that region. Since the average fitness has not yet leveled off in trial two, the gaits can most likely still be significantly improved upon.

In the third trial the average fitness flattens out around generation 30 though the maximum fitness continues to climb. The fitness of these gaits can still be improved, but probably not by a significant amount since the GA is in a stable region with a high maximum fitness and thus will probably not be able to break out and find a better region.

4.3.3 Gallop

The gallop is normally the fastest of all the dynamic gaits for a quadruped. For these trials the base phase offsets were set for a transverse gallop as in Figure 2.9. The reason that the transverse gallop was chosen instead of the rotary gallop was that in nature it seems that animals with more rigid spines, such as horses and cattle, tend to favor the transverse gallop [55]. Since the Aibo has no ability to flex its spine at all, this type of gallop seemed the most appropriate. The number of elites was kept at 5 and the mutation rate was kept at 15%.

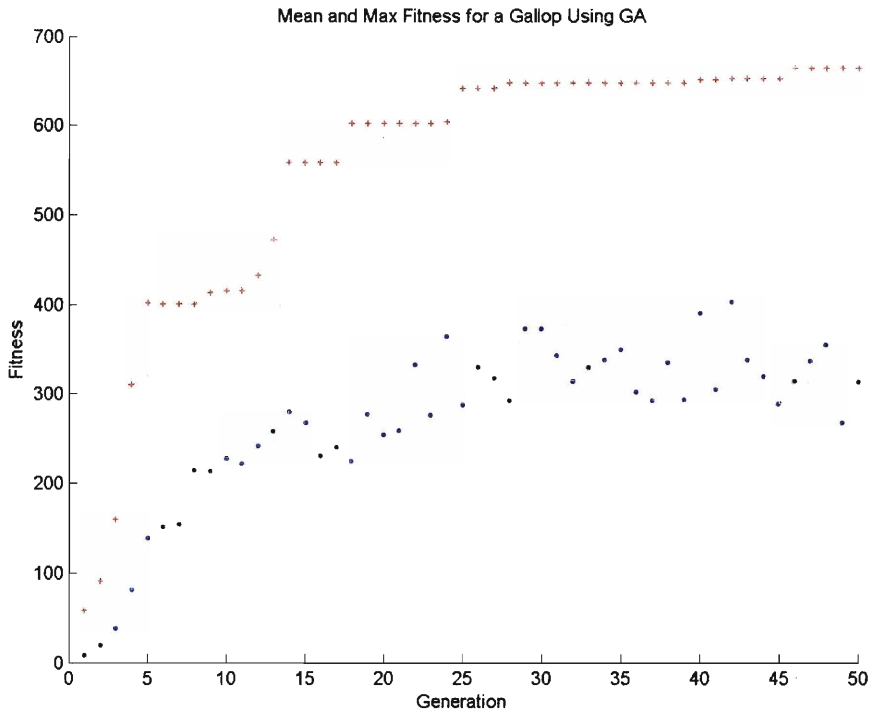

Figure 4.9: Plot of the First Run for the Gallop Using the GA

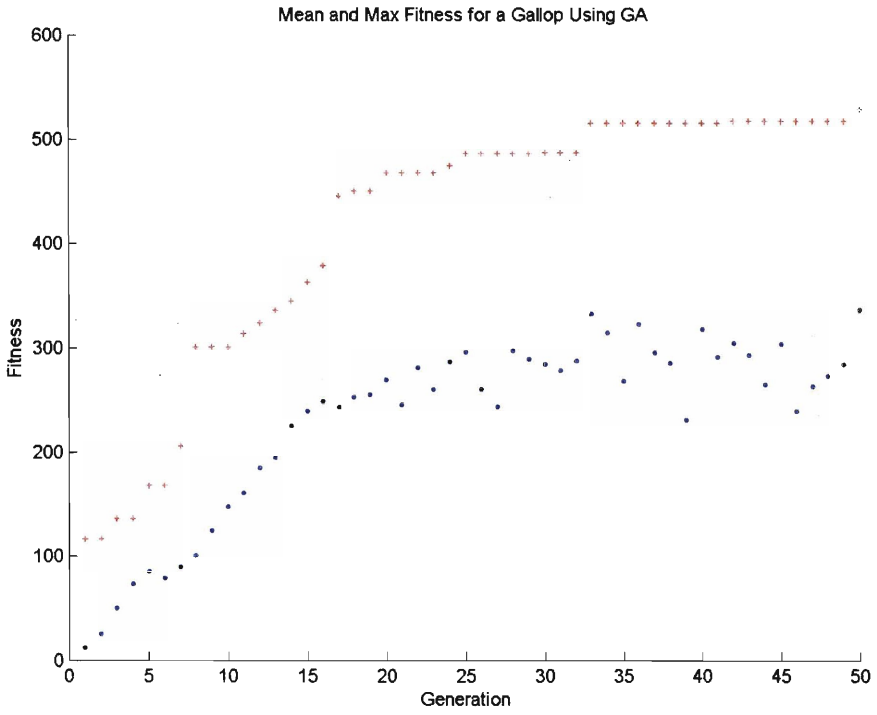

Figure 4.10: Plot of the Second Run for the Gallop Using the GA

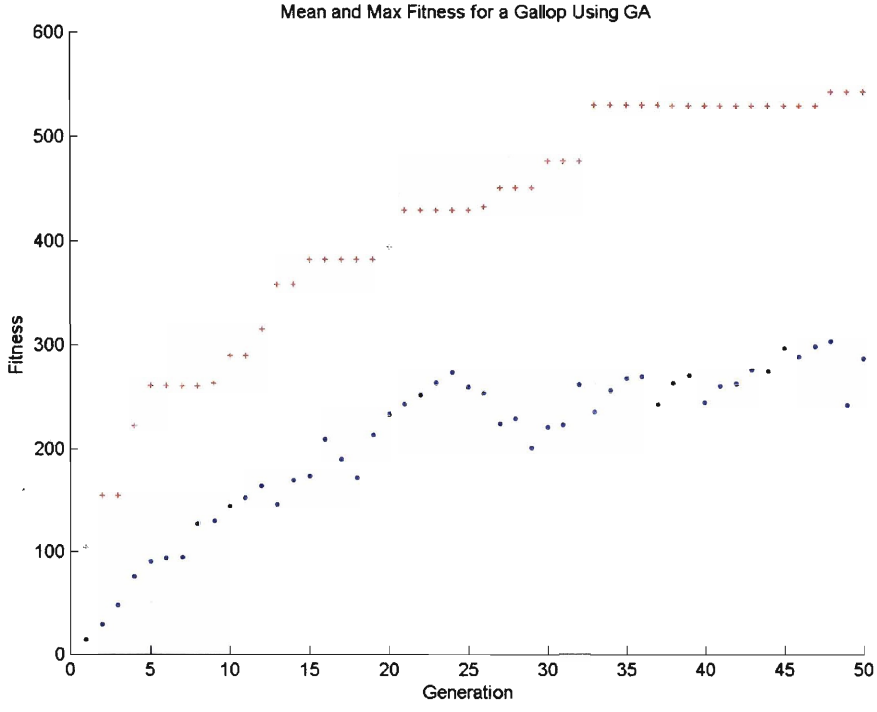

Figure 4.11: Plot of the Third Run for the Gallop Using the GA

Both trials one and two show a typical GA progression with the maximum and average fitness scores climbing rapidly at first and then leveling off. The averages seem moderately high as well, indicating that the GA isn't just stuck in a region of instability. This means that there is probably little further improvement to be made on the gaits in these two trials.

In the case of trial three, neither the maximum or average fitness scores have leveled off yet, so some improvement can still be made to these gaits. Again, the fact that the average fitness scores are still going up slightly shows that the GA is not trapped in an unstable region.

Unfortunately, the fitness scores for all three of these trials are actually lower than the fitness scores found for the canter and even most of the trots. The reason for this most likely has to do with the design of the Aibo robot, which makes it difficult to initiate flight stages. Without large flight stages the only way a gallop can remain stable is to increase its ground fraction so that it becomes more of a lurching walk rather than a fast gallop. These issues will be discussed in more detail in the analysis section.

4.4 Secondary Results

For the second phase of learning for the genetic algorithm the gaits found in the initial phase of learning were used to try and find even better gaits for the trot, canter, and gallop. To this end, for each of the three types of gaits, the best five individuals were taken from each of the three trials that were done with the GA during the first phase of learning. These fifteen parameter sets were combined with the best single gait from each of the three trials that were done with the gradient-climbing reinforcement learning algorithm during the first phase of learning. This gave a total of eighteen gaits, which were used to seed the first generation of the GA for the second phase of learning. The remaining seventeen individuals of the first generation

were created by breeding together the eighteen seed gaits based on their fitness scores using the same breeding function and likelihood selection method that was used to create each new generation as is described in Section 4.2. Once the initial population was generated, the GA ran just as it did in the initial round of learning, but in this case, since the seed gaits were being used as the first generation rather than randomly generated gaits there was only one trial for each the trot, the canter, and the gallop. The mutation factor was kept at 15% for these runs, but the number of elites was increased to seven because it was found that this aided the learning process by keeping more of the gaits from the first generation that were known to be good.

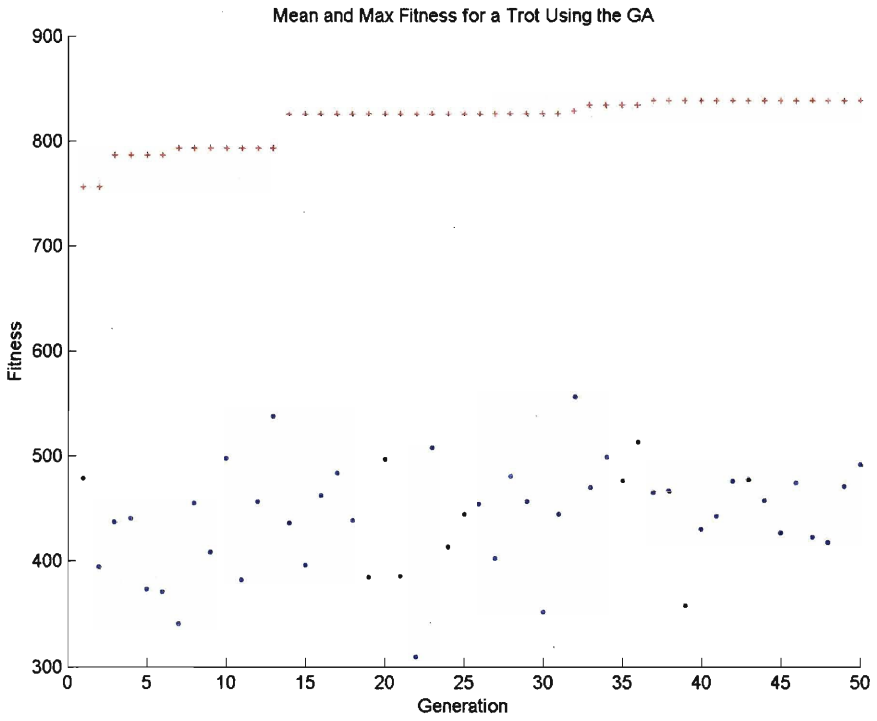

Figure 4.12: Plot of the Second Phase of Learning for the Trot Using the GA

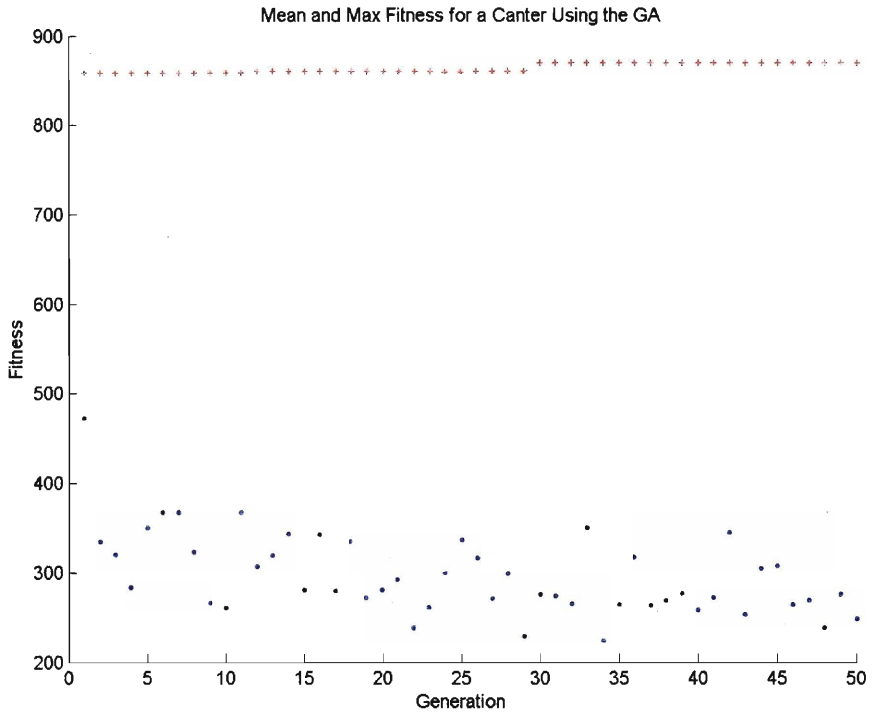

Figure 4.13: Plot of the Second Phase of Learning for the Canter Using the GA

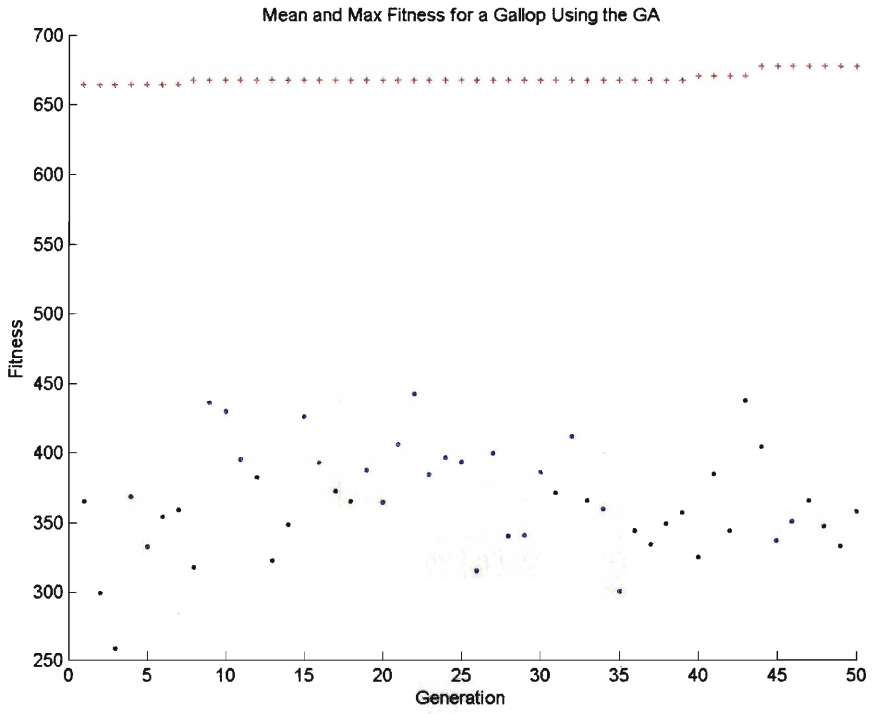

Figure 4.14: Plot of the Second Phase of Learning for the Gallop Using the GA

The trot showed the most improvement, while the maximum fitness of the canter and the gallop didn't increase much at all. However, though the maximum fitness scores did not significantly increase during the phase two trials, the best gaits did become much more consistent as the generations passed. As mentioned previously, the simulator did not generally give the same results twice when testing a gait due to errors in timing caused by the slow update rate of the controller. The best gaits found during phase one of the learning tended to be extremely inconsistent, occasionally doing very well, but usually falling over quite quickly. The gaits found in the second phase of the research were only slightly better in terms of fitness, but gave much more consistently high scores and fell over less when they were repeatedly tested.

4.5 Analysis of Gaits

4.5.1 Trot

Front Step Height (cm)	3.1
Hind Step Height (cm)	2.6
Front Cutoff Angle (degrees)	11.5
Hind Cutoff Angle (degrees)	8.5
Front Cutoff Percentage (%)	1.0
Hind Cutoff Percentage (%)	42.0
Stride Period (sec)	0.2
Ground Fraction (%)	60.3
Leg 1 Phase	0.484
Leg 2 Phase	0.568
Leg 3 Phase	0.956
Front Y-Coordinate (cm)	0.9
Hind Y-Coordinate (cm)	1.1
Step Length (cm)	7.9
Stance Width (cm)	0.4
Front Body Height (cm)	11.4
Hind Body Height (cm)	11.4

Table 4.1: Parameters for the Trot Learned by the GA

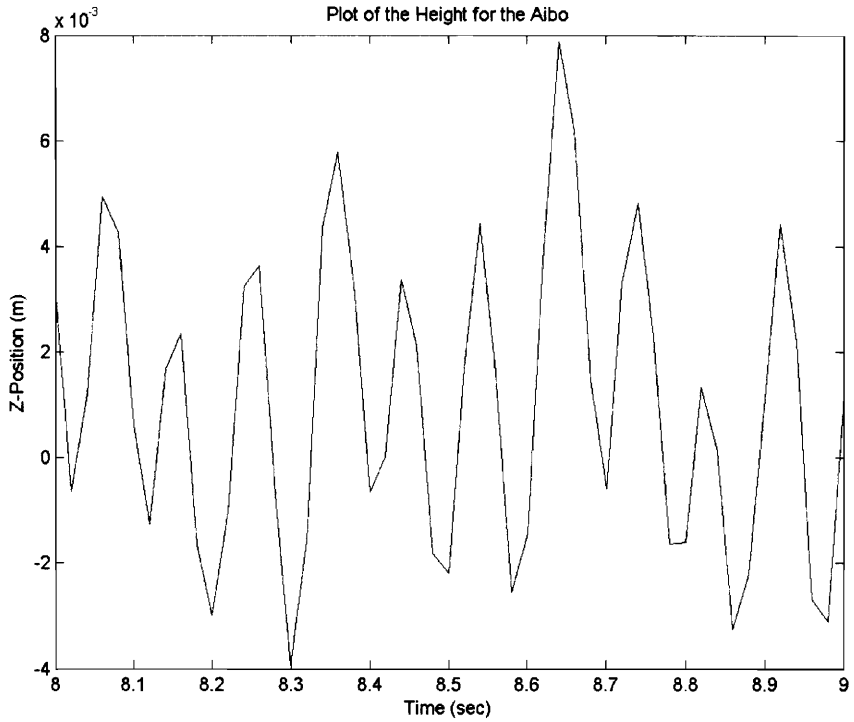

Figure 4.15: Height for the Aibo During the GA Trot

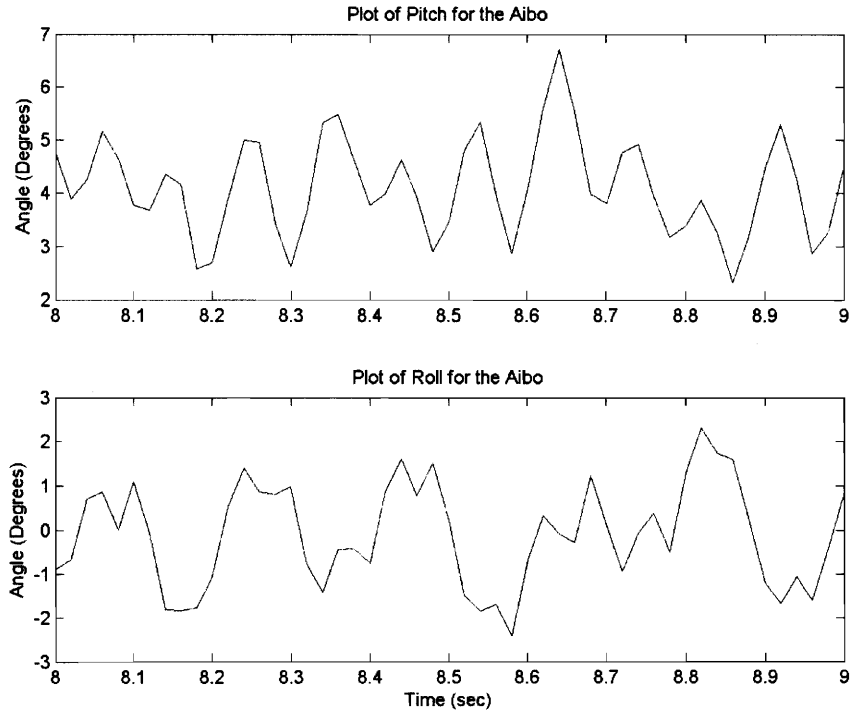

Figure 4.16: Pitch and Roll for the Aibo During the GA Trot

As expected, the trot found by the GA was a fairly steady gait with relatively small variations in height, pitch, and roll. Height measurements were taken from the center of the chest, pitch was taken from the difference in the heights of the front and hind hips on the right side, and roll was taken from the difference in the heights of the left and right sides of the chest. As the robot runs, the height fluctuates by about 1 centimeter, the pitch by 3 degrees, and the roll by 4 degrees.

For a normal trot, the diagonal pairs of legs work in tandem, but the trot found by the GA is a little different because the back feet are on the ground for a longer amount of time than the front feet, which means that there are brief periods where only one back foot is on the ground and all three other feet are in the air. The results of this can be seen in Figures 4.15 and 4.16. Initially, the body is tilted to the left with a roll angle of about -2 degrees as the left front and right back legs touch down. The Aibo then begins rolling to the right because its center of mass is on the right side of the support line since the weight of the robot's head means that the Aibo's center of mass is slightly forward of the middle of the body which the support line runs under. The left front leg is then picked up and only the right back foot is on the ground causing the robot to roll to the right briefly. Next, the right front foot and left back foot pair hits the ground causing a small roll back to the right from the impact followed by a large roll to the left because now the center of mass is to the left of the support line. The robot pitches up every time a diagonal leg pair touches down and then falls back its minimum value of about 2.5 degrees before the next diagonal leg pair hits causing it to pitch upwards again. Both the pitch and the height reach their minimum values whenever a pair of feet hit the ground and their maximum values right before the front foot lifts off the ground. The roll on the other hand, cycles at about half the rate of the pitch and height. The reason the data curves become distorted after 8.7 seconds is that the robot stumbles a bit at that point.

The reason that the Aibo's path curves slightly to the right is unknown. It might have something to do with the bouncing observed in the back legs of the robot while it's trotting or with the fact that the front feet stay on the ground for a shorter time than the back feet.

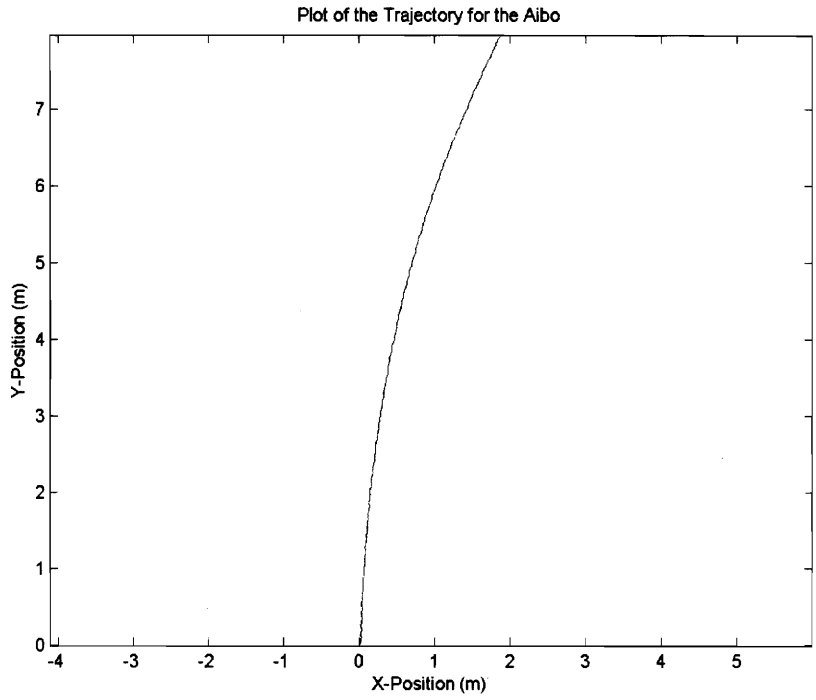

Figure 4.17: Trajectory for the Aibo During the GA Trot

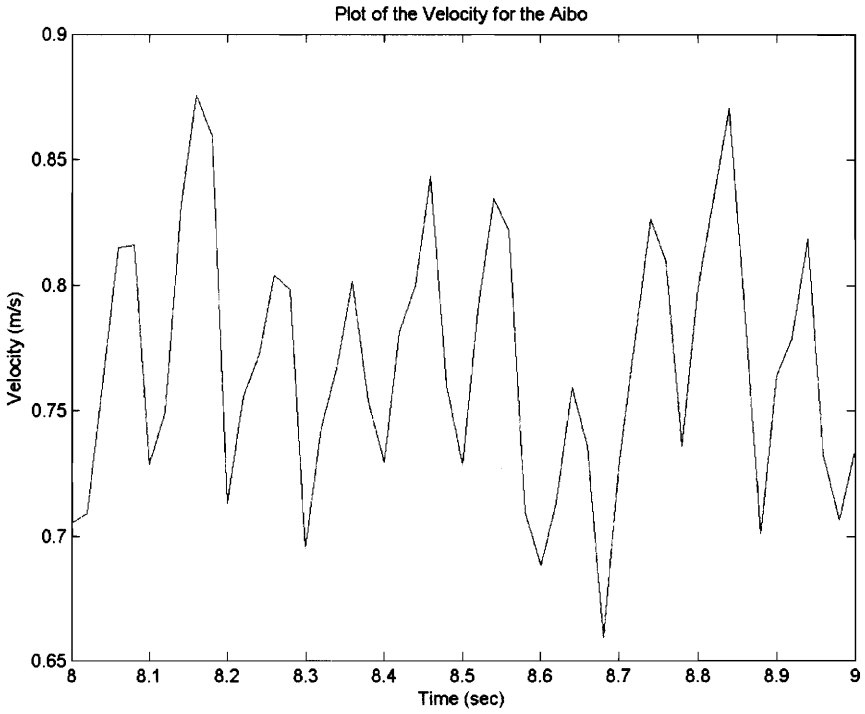

Figure 4.18: Velocity for the Aibo During the GA Trot

The average velocity for this trot was 75.8 cm/s and varied at the same frequency as the height and the pitch. When a pair of legs touches down, they accelerate backwards increasing the speed. Then the front leg is picked up and the back leg begins to slow down as it approached its target position, causing the robot's velocity to drop rapidly. Finally, the other pair of legs touches down and the robot speeds up again.

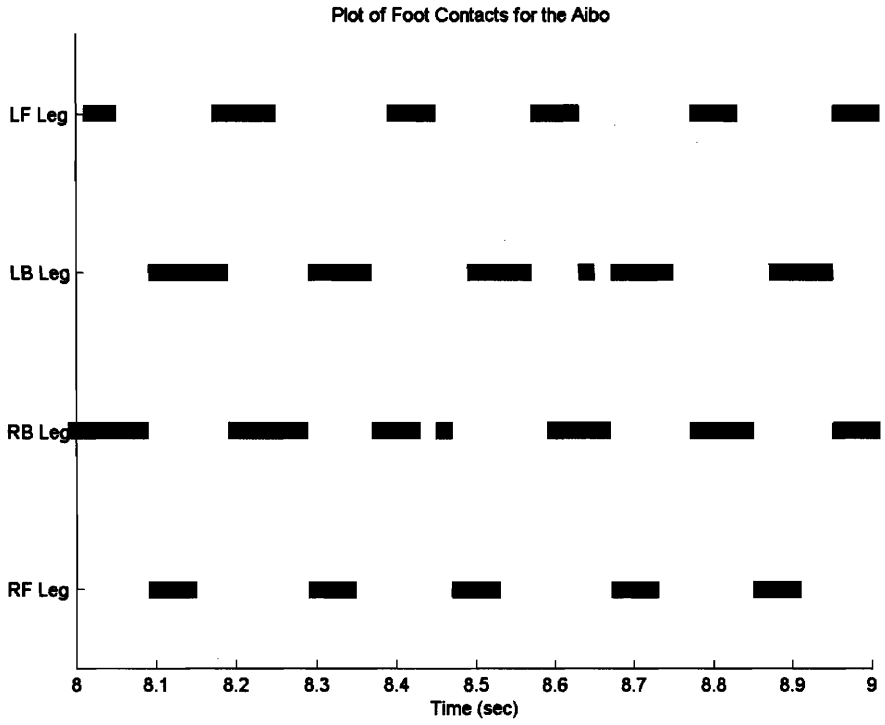

Figure 4.19: Foot Contacts for the Robot During the GA Trot

For the trot found by the GA, the front left and back right legs are supposed to hit the ground together, then when they lift up the right front leg is supposed to land followed closely by the back left leg. This phasing is actually similar to that of a canter though in a canter the delay between the front right and back left leg is larger. As the Aibo runs however, the phasing shifts somewhat due to the slow update rate of the controller, so the phasing isn't always perfect as can be seen in Figure 4.19. The

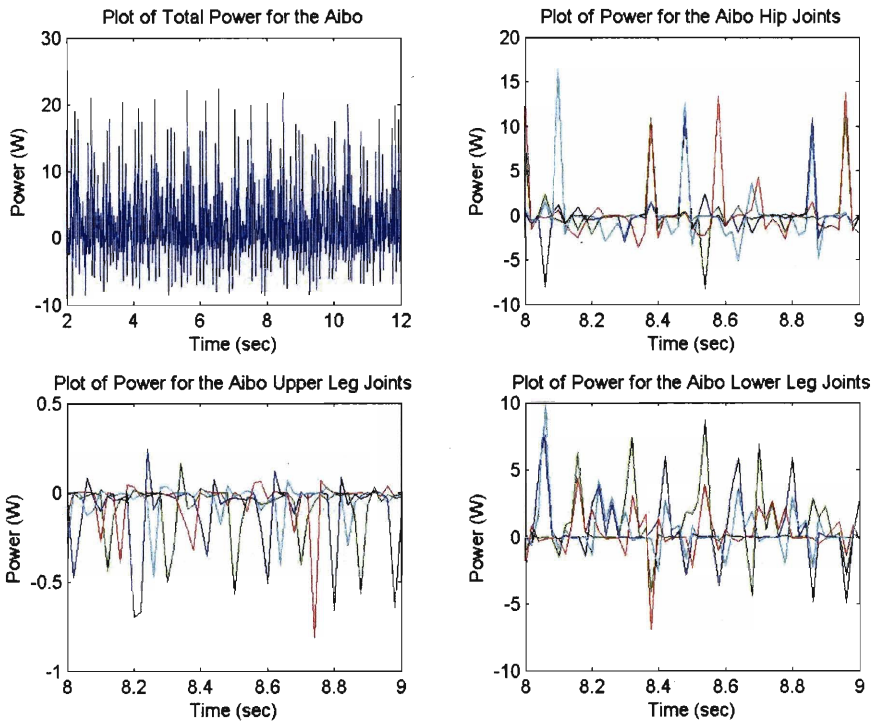

Figure 4.21: Power Used by the Aibo During the GA Trot

In Figures 4.20 and 4.21, hip joints refer to the joints that swing the legs forward and backward, upper leg joints are the joints that spread the legs outward or inward, and the lower leg joints are the knee joints. Figure 4.21 also uses the same legend as 4.20 for plotting the individual legs. The power plots were found by multiplying the speeds and torques of each joint.

For the hip joints, the maximum torque and power come during the period that the foot is first picked up and begins to swing forward. This is because for this gait the leg must complete the step forward section of the leg path in 39.7% of the total stride time so the foot is traveling fastest during this section and the hip torque needs to ramp up to accelerate the foot. The step forward section is also the part of the stride in which the phase errors are compensated for so the torque varies every stride as the required speed varies. Naturally, since the torques and speeds are highest in that section the power requirements are highest there too.

In the knee joints, the highest torques come when the foot hits the ground and the leg begins moving backwards, while during the step forward section of the foot path

controller compensates for these errors to bring the phasing back to its correct values, but cannot make the phasing perfect for every stride.

The rear legs are on the ground for 44.9% of the stride while the front legs are on the ground for only 32.3% of the stride. The hind legs also have a tendency to bounce a bit on the ground leading to the breaks in the foot contacts seen in Figure 4.19. The reason that the back legs stay on the ground longer than the front legs and tend to bounce is unknown. Looking at the gait parameters one would expect the back legs to be on the ground less than or equal to the amount of time that the front legs are on the ground. This discrepancy may have something to do with the fact that the back legs are pushing the Aibo forward while the front legs are pulling it forward or it may be because there is more weight over the front legs due to the mass of the head.

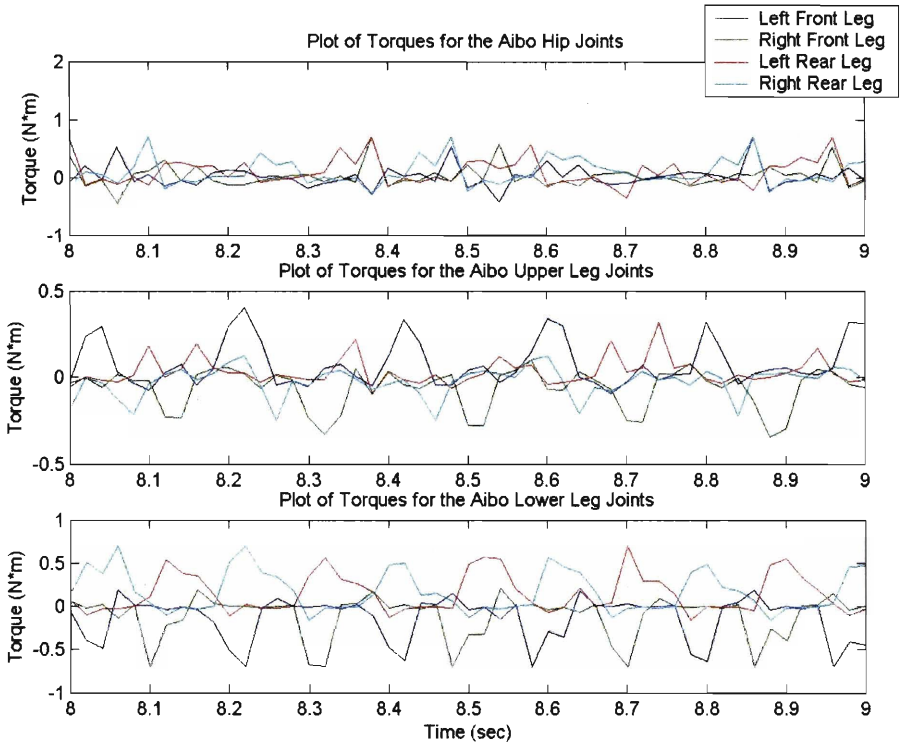

Figure 4.20: Torques for the Aibo During the GA Trot

they are nearly zero. This indicates that the knee joints are actually the ones providing the most thrust in this stage rather than the hips. The reason that the torques for the front legs are negative while the torques for the back legs are positive during the thrust stage is because the knees bend in opposite directions.

The torque and power required for the upper leg joints is minimal throughout the gait because the stance width is almost zero and the joints just have to maintain that.

4.5.2 Canter

Front Step Height (cm)	2.6
Hind Step Height (cm)	3.3
Front Cutoff Angle (degrees)	7.0
Hind Cutoff Angle (degrees)	8.0
Front Cutoff Percentage (%)	35.0
Hind Cutoff Percentage (%)	39.0
Stride Period (sec)	0.2
Ground Fraction (%)	60.3
Leg 1 Phase	0.332
Leg 2 Phase	0.770
Leg 3 Phase	0.000
Front Y-Coordinate (cm)	2.0
Hind Y-Coordinate (cm)	0.4
Step Length (cm)	9.2
Stance Width (cm)	0.0
Front Body Height (cm)	10.6
Hind Body Height (cm)	10.6

Table 4.2: Parameters for the Canter Learned by the GA

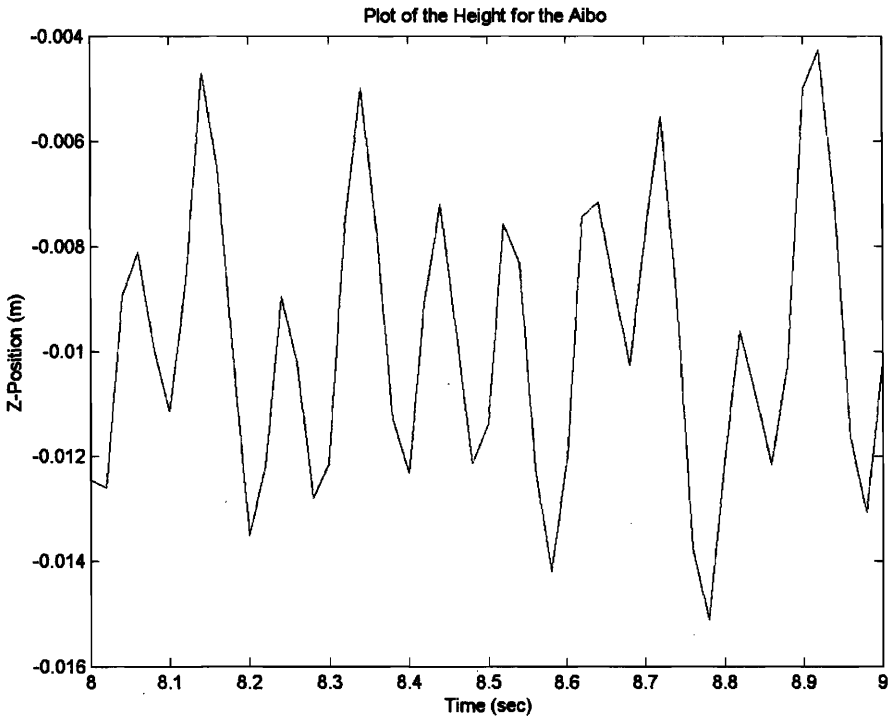

Figure 4.22: Height for the Aibo During the GA Canter

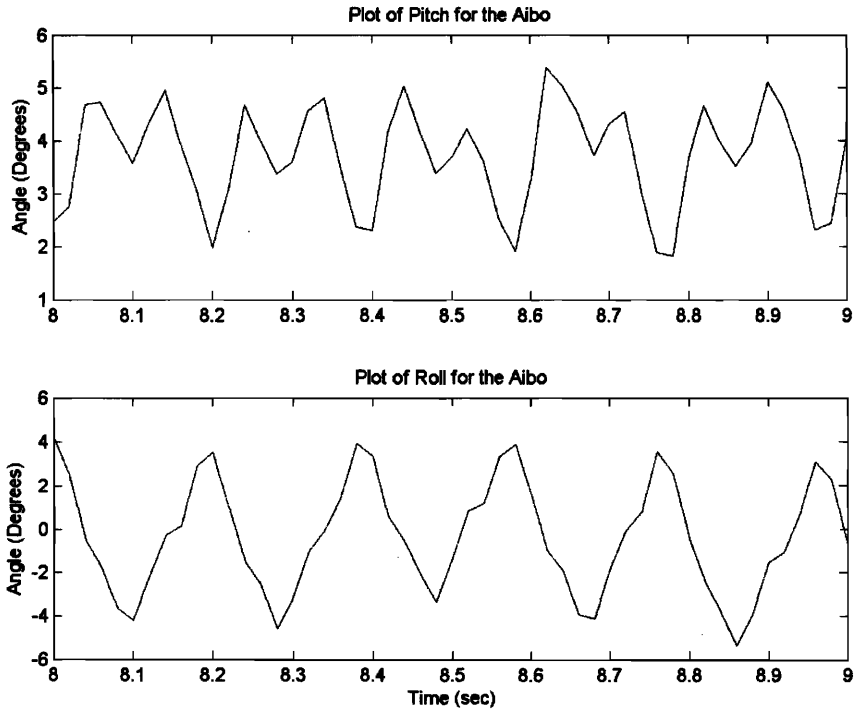

Figure 4.23: Pitch and Roll for the Aibo During the GA Canter

The roll variation for the canter is about 9 degrees, which is more than double the roll variation for the trot, though the height and pitch variations are about the same at 1 centimeter and 2.5 degrees respectively. The average roll remains near zero, but the average pitch is actually less than the average pitch for the trot at 3.3 degrees. As with the trot, the reason for the slight tilt to the upwards is unknown since the front and back leg heights are set to be the same.

The maximum roll angle of 4 degrees occurs just before the right front foot touches down. When the right front foot hits the ground, it acts as a break slowing the speed of the Aibo. This causes the leg to hit the ground hard, bouncing the robot up so that its pitch increases sharply and making the robot roll to the left. Shortly after the Aibo pitches upward, it begins to fall back down again. About halfway through the fall, when the robot is tilted to its minimum roll value of -4 degrees, the left front leg hits the ground and also acts as a break, making the robot bounce up again and tilt back to the right. The bounce is not as severe in this case because the robot was higher off the ground when the left front leg hit than it was when the right front leg hit, so the left front leg didn't push as hard against the ground since it was already near its maximum extension when it touched down. After the bounce from the left front leg, the robot falls back down to its minimum pitch value of 2.2 degrees as it continues rolling right. Finally the right front leg hits the ground again and the process is repeated. The back legs don't seem to contribute much to either the pitch or the roll of the gait.

The canter is the straightest of the gaits overall, though the robot does make a large turn to the right at the beginning before it properly settles into the canter. The tendency to curve to the right is still present, but it is less pronounced, perhaps due to the higher speed creating a larger momentum for the Aibo and making it more difficult to turn.

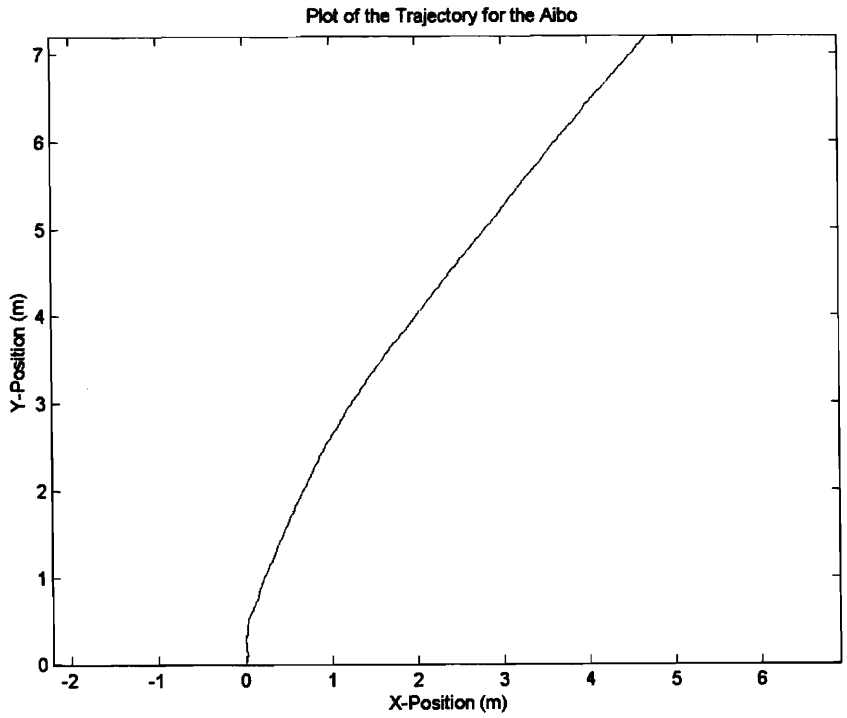

Figure 4.24: Trajectory for the Aibo During the GA Canter

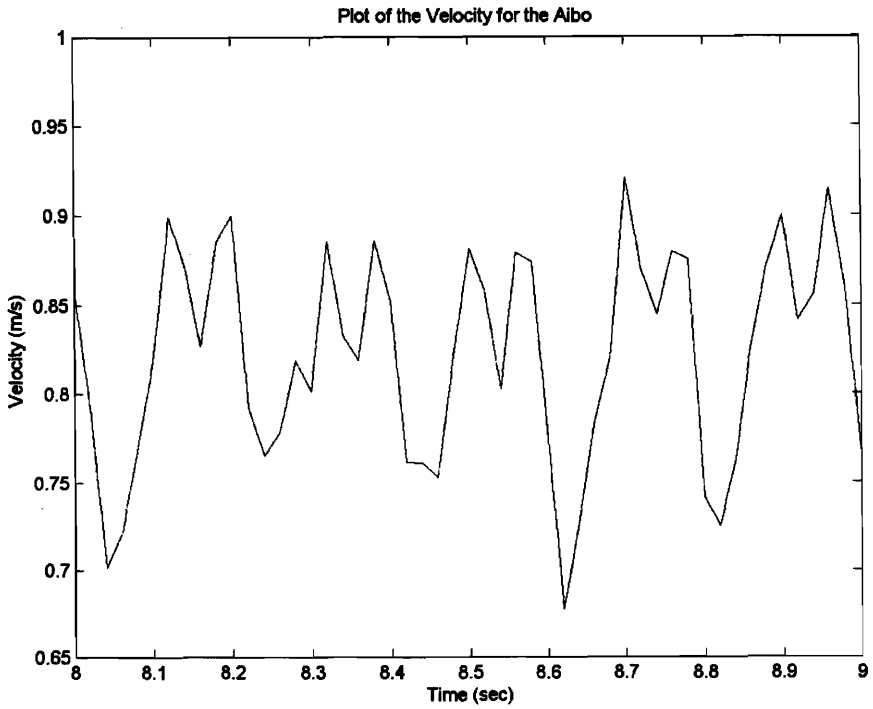

Figure 4.25: Velocity for the Aibo During the GA Canter

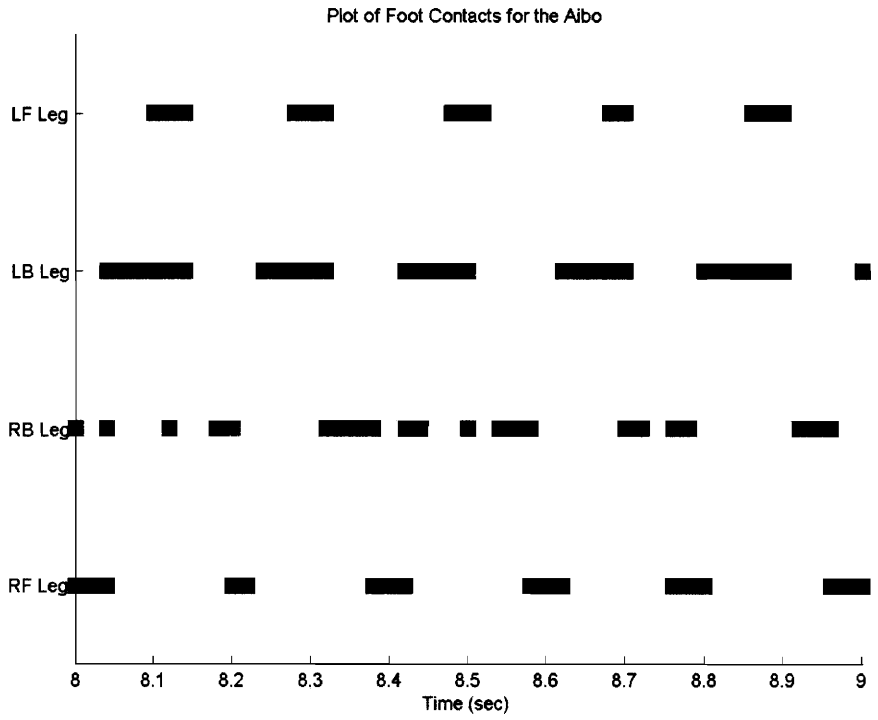

Figure 4.26: Foot Contacts for the Aibo During the GA Canter

The average velocity for the canter developed by the GA was 80 cm/s. The velocity profile of the Aibo while it is cantering is somewhat odd due to the problems that occurred with the right back foot during the gait. That foot did not hit the ground with the correct phasing and when it was on the ground it bounced a lot. The reason for this slight stuttering along the ground is unknown, but it appears to be related to the large bounces caused by the front legs when they touch down.

When the front right foot touches down, it acts as a break and the velocity drops off massively while the body begins to roll left from the impact. The touch down of the back left foot then accelerates the robot forward. Since the body is tilted to the left, the front left leg hits the ground early so it is not properly in phase with the right back leg. In addition, because it hits early, the leg is still in the tail-end of its step forward phase so it is coming almost straight down when it hits. This causes the front of the Aibo to bounce, meaning that the front left leg stays on the ground a shorter time than it should and making the robot's roll rightwards. It also briefly bounces the

back right leg off the ground causing the velocity to dip again. Then the back right foot lands on back the ground and continues accelerating the Aibo, making the velocity rise to its previous value as the pitch of the Aibo falls back to its minimum. Finally, the front right foot hits the ground hard due to the low pitch angle and high positive roll angle and the process repeats itself.

The bouncing of the front feet due to their acting as breaks to the Aibo's forward motion account for the reason that the front feet are only on the ground for 30.9% of the stride while the rear feet are on the ground for 47.9% of the stride.

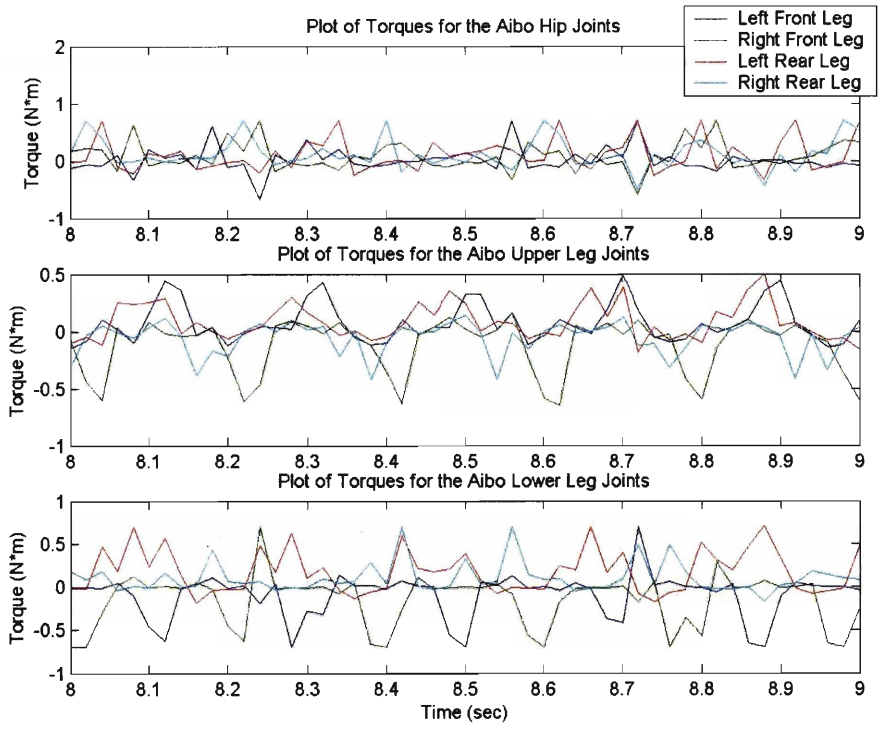

Figure 4.27: Torques for the Aibo During the GA Canter

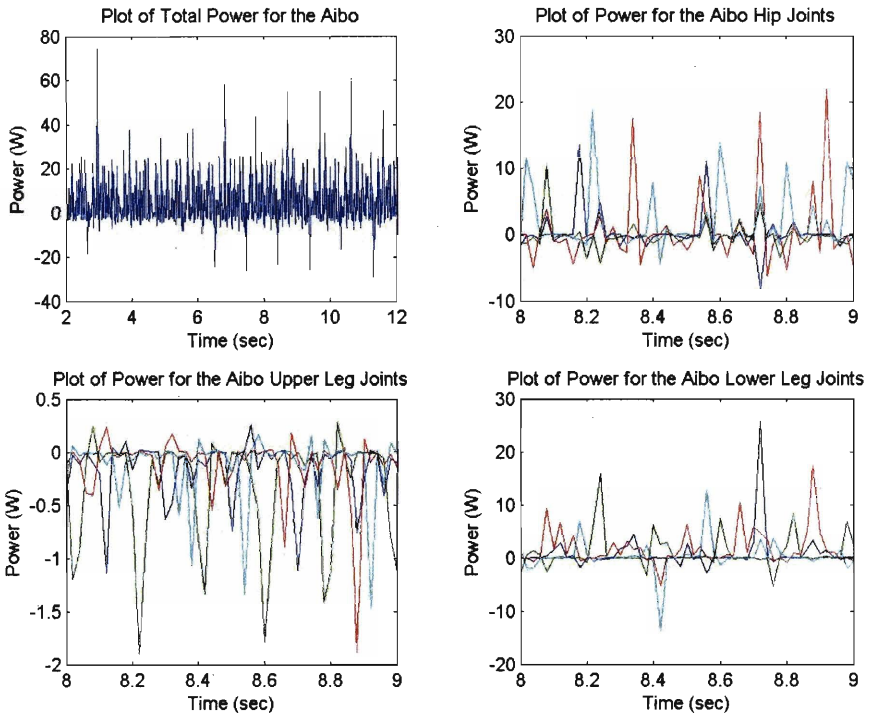

Figure 4.28: Power Used by the Aibo During the Canter

For the hip joints, again most of the torque is applied during the step forward stage just as in the trot; however, for the canter the velocities required during this stage are higher because the step length is 9.2 cm as opposed to 7.9 cm while the completion time is the same as it was in the trot. Thus the duration of the high torques is longer and the power levels needed are much larger. The high speed required during the step forward stage also leads to somewhat higher torque and power values for the knee joints than were necessary for the trot, though the torques for the thrust stage are very similar. The upper leg joint torques are also somewhat higher because there are many points in the gait where only one leg is on the ground so a higher level of torque is required to keep the legs from splaying outward than in the trot where two legs on opposite sides of the body were down at the same time. Overall, this gait, while fast is not very power efficient due to the large step forward speed required.

4.5.3 Gallop

Front Step Height (cm)	1.7
Hind Step Height (cm)	3.1
Front Cutoff Angle (degrees)	28.0
Hind Cutoff Angle (degrees)	30.0
Front Cutoff Percentage (%)	35.0
Hind Cutoff Percentage (%)	32.0
Stride Period (sec)	0.2
Ground Fraction (%)	63.3
Leg 1 Phase	0.160
Leg 2 Phase	0.452
Leg 3 Phase	0.676
Front Y-Coordinate (cm)	1.5
Hind Y-Coordinate (cm)	1.6
Step Length (cm)	6.1
Stance Width (cm)	-0.6
Front Body Height (cm)	12.5
Hind Body Height (cm)	12.6

Table 4.3: Parameters for the Gallop Learned by the GA

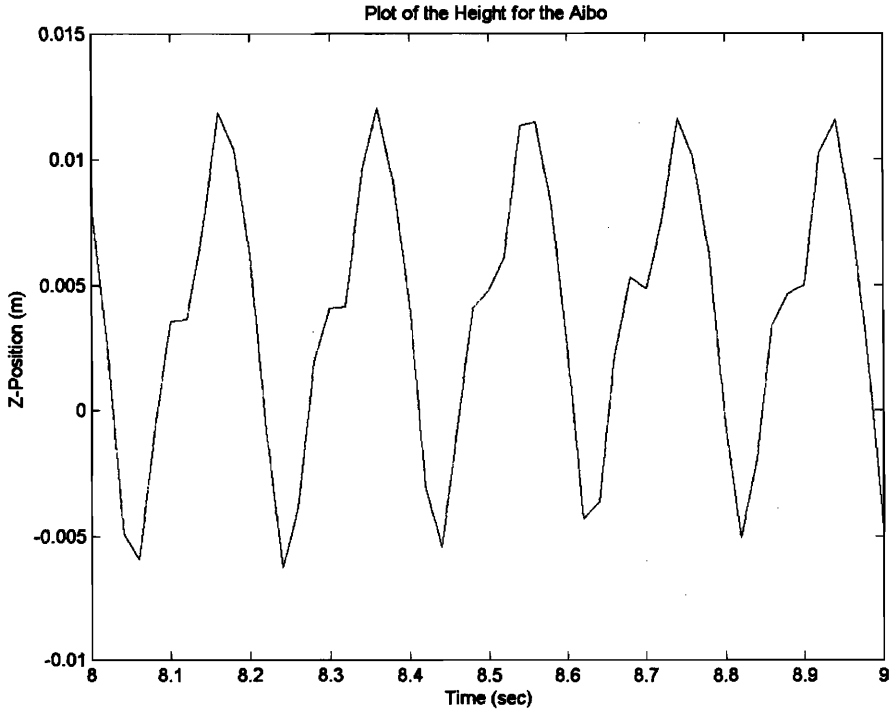

Figure 4.29: Heights for the Aibo During the GA Gallop

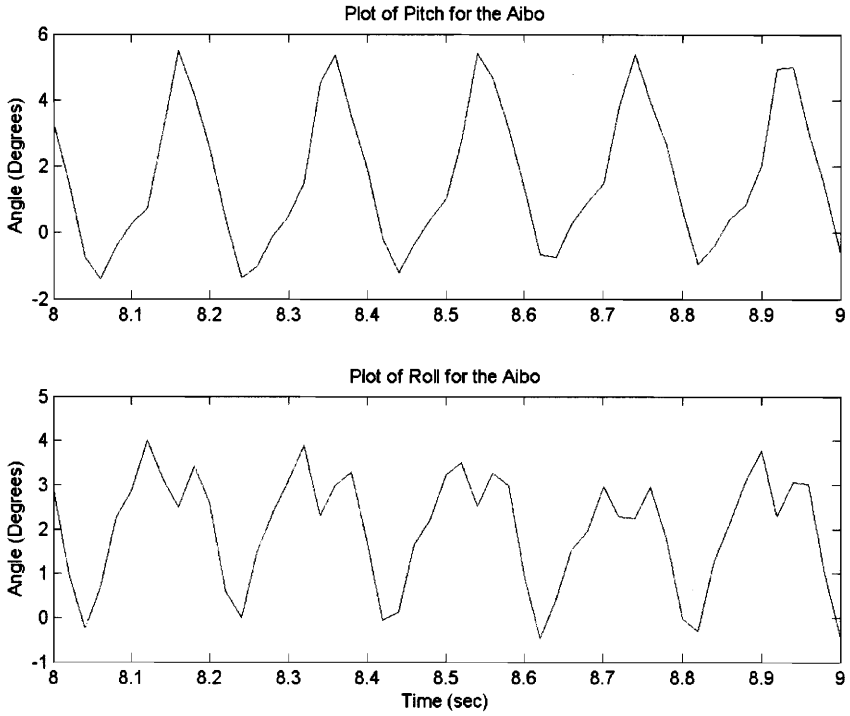

Figure 4.30: Pitch and Roll for the Aibo During the GA Gallop

For the GA transverse gallop the height, pitch, and roll peaks all match up together fairly well unlike with the canter and the trot where the height and pitch changed at twice the rate of the roll. The height and pitch variations are about double those of the trot at 1.8 cm and 7 degrees respectively, while the roll variation is about the same as for the trot at 4.2 degrees. Experiments with animals [42] and with other galloping robot simulations [32] have shown that the height variation is typically about 10% of a quadruped's standing height. In the case of the Aibo, the standing height is 12 cm so the height variation in this gait is somewhat larger than is typical, but that is not surprising given that this gait is too slow to be called a true gallop.

The robot starts out perfectly level and then the left front leg touches down causing the robot to make a large roll to the right. This is followed by the right front leg coming down causing the Aibo to roll slightly to the left. Next, the left back leg hits the ground and the robot rolls slightly to the right and finally, the right back leg touches down, inducing a large roll to the left. The robot pitches up from a minimum of about -1 degrees as the left front leg comes down and then pitches up even faster

after the right front leg touches down. Once the left back leg hits the ground the Aibo begins pitching down again and continues to do so after the right back leg hits. The cycle then repeats itself. As before, the height follows the pitch curve fairly closely.

The transverse gallop has the sharpest curve to its trajectory out of the three types of gaits, as can be seen from Figure 4.31. This is to be expected considering that the gallop has the highest average roll angle of the three types of gaits at 1.7 degrees and that for both the front and the rear pairs of legs the left leg always hits first followed by the right, leading to a slight yaw that makes the robot's path curve.

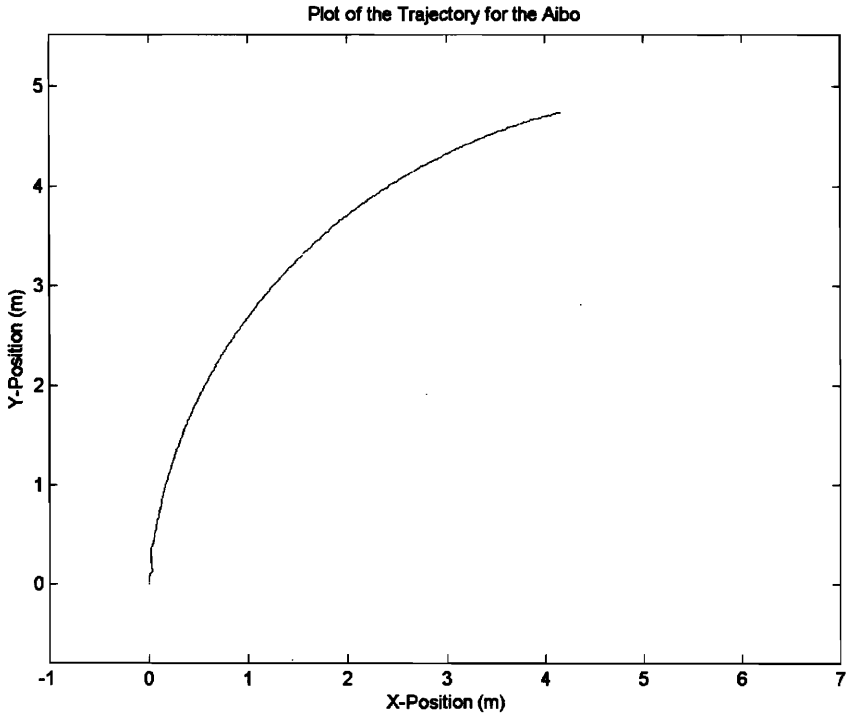

Figure 4.31: Trajectory for the Aibo During the GA Gallop

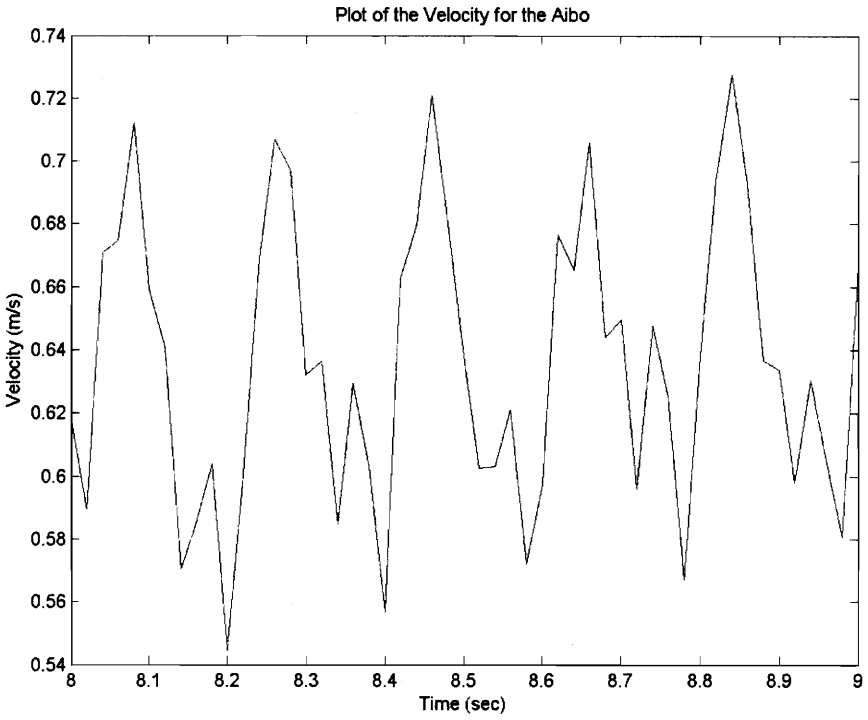

Figure 4.32: Velocity of the Aibo During the GA Gallop

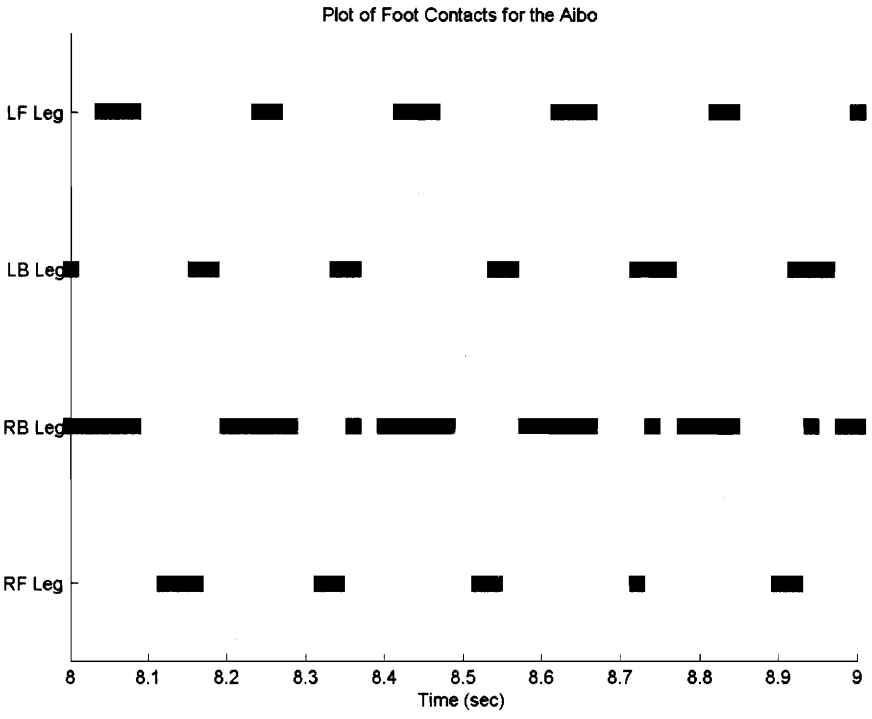

Figure 4.33: Foot Contacts for the Aibo During the GA Gallop

The average velocity for this gait was 61.8 cm/s, which is considerably slower than the speeds of the trot or the canter. This may have been because the lower limit of 0.2 seconds that was imposed on the stride time was too high and the torque limits too low for the Aibo to perform a true galloping gait with significant flight times. The gallop developed by Krasny [32] had a stride time of less than 0.2 seconds even though the robot he simulated was much larger than the Aibo, so the Aibo would probably need a stride time that was even lower. Unfortunately, this was not feasible given the torque limits on the motors and the slow update rate of the controller.

While running, the large jumps upwards in the velocity in Figure 4.32 are due to the thrust from the right back leg. Next, the velocity drops off until the left back leg touches down whereupon it rises briefly again and then resumes its drop until the back right leg touches down again. The front legs seem to have only minimal contribution to the velocity profile.

The foot contacts for this gait are very strange because the right back foot is on the ground 52.5% of the time while the other feet are on the ground for only 27.2% of the time. In addition, the phase difference between the left front and right front legs is much larger than it should be. The brief touchdown periods for three of the four legs may be due to bouncing effects like those that occurred in the canter, but this was unable to be determined for certain. There were short periods of flight for this gait, lasting 15.1% of the stride period on average, but these flight periods actually occurred between the liftoff of the front left leg and the touch down of the right front leg rather than during the period between the liftoff of the front right leg and the touch down of the rear left leg as one would expect. The flight periods happened because of the unusually large phase difference between the two front legs and the very brief ground contact periods for both of them.

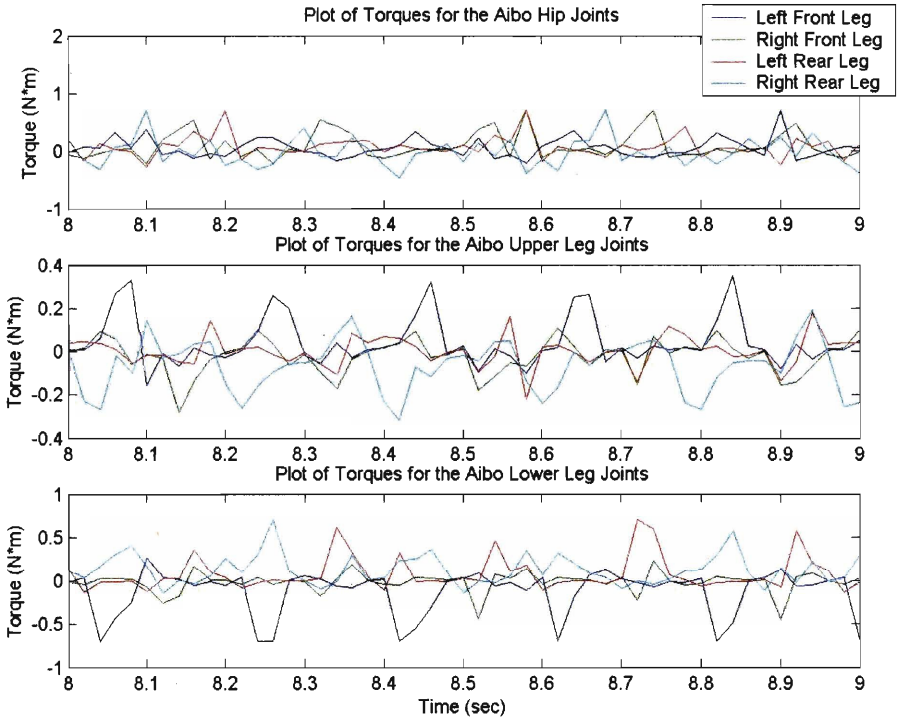

Figure 4.34: Torques for the Aibo During the GA Gallop

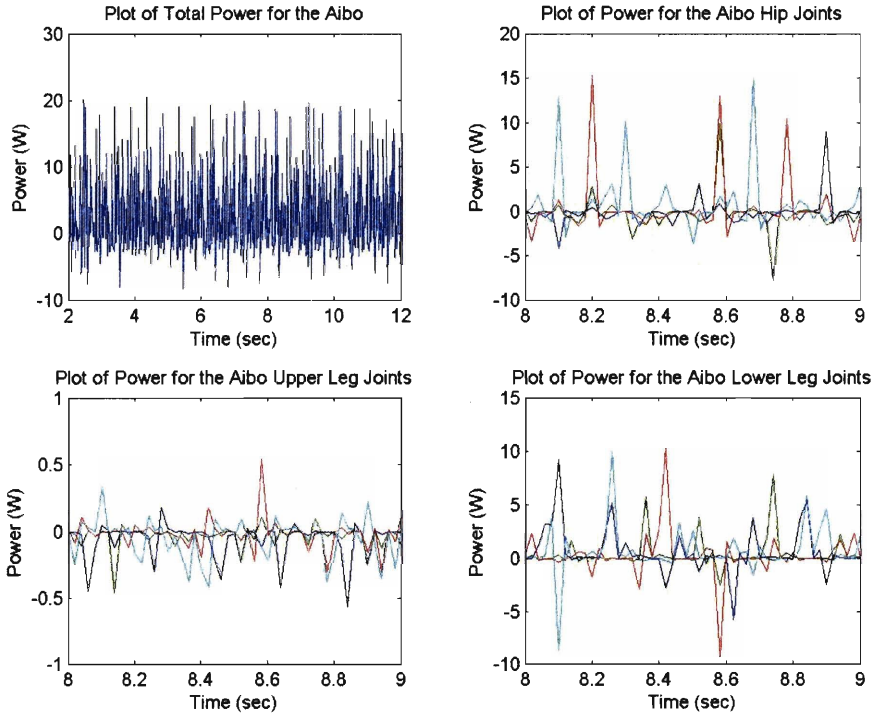

Figure 4.35: Power Use for the Aibo During the GA Gallop

As with all the other gaits, most of the power was used by hips while trying to swing the legs forward during the step stage of the foot path. The torques and power required is slightly lower than for the trot because the forward step speed is also somewhat lower. The torques for the lower legs again peak during the thrust stage, though the torque for the front right leg is strangely lower than the rest, perhaps due to the fact that it is on the ground less than the other legs and so doesn't have time to ramp up to full torque. The upper leg torque is rather small because the legs are bent slightly inwards so there isn't as much torque trying to spread the legs out as there is with the other gaits.

Chapter 5 Gradient-Climbing Reinforcement Learning Algorithm

5.1 Overview

A gradient-climbing reinforcement learning algorithm (GCRL) as implemented by [30] can be considered a degenerate form of standard policy-gradient reinforcement learning techniques [64, 5]. These gradient-climbing algorithms converge to local optima rather than global optima and so the starting point of the algorithm is important.

The algorithm begins by generating a number of candidates through small random perturbations of some provided seed parameter set and then evaluating those candidates to find their fitness scores. Next, the algorithm looks at all the candidates and for each parameter n, calculates the average reward function of the candidates in which the value of n was increased, the average reward function of the candidates in which the value of n was decreased, and the average reward function of the candidates in which the value of n was not changed. Based on this set of three averages for n, the algorithm calculates an adjustment number for the parameter n. Once adjustment numbers have been calculated for all the parameters, the adjustments are added to the original seed parameters to create a new seed set. An example of this process for estimating the gradient in a single dimension can be seen in Figure 5.1

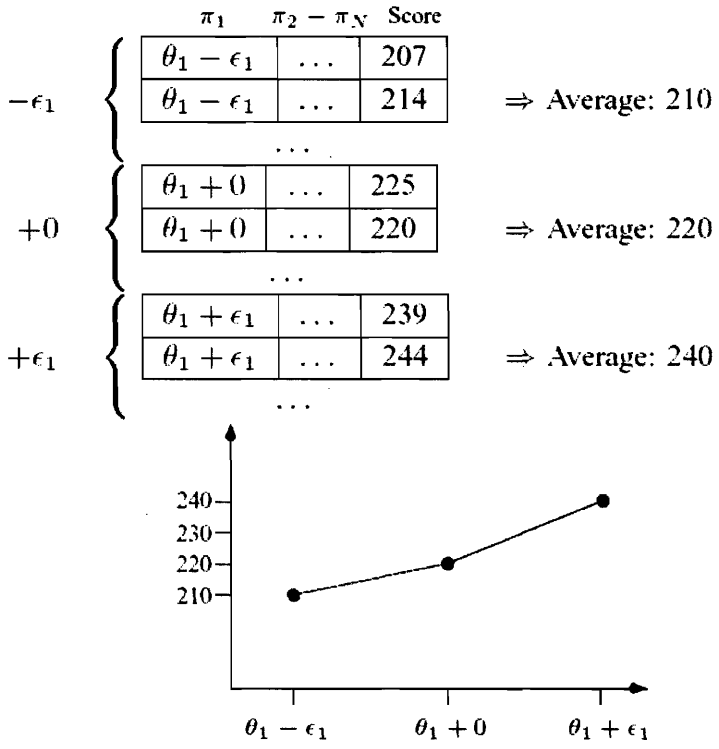

Figure 5.1: Example of the Process for Estimating the Gradient in a Single Dimension [30]

5.2 Implementation

The algorithm first starts by generating a random population of parameter sets in the same way that the random populations were generated in for the GA as explained in Section 4.1. The same program as before was used to make sure each of the randomly generated parameter sets was feasible before it was added to the first generation. The parameter sets were also normalized just as in the GA so that each parameter was a number between 0 and 99. Once the first generation was created, it was evaluated by the simulator using the fitness function from Equations 4.1 and 4.2. The best parameter set from this generation was used as the first seed set for the GCRL. The user could also skip this first generation and just load a seed set into the program if desired.

Once the first seed set had been chosen, the GCRL started up. The GCRL was initially implemented as described in Section 5.1, but the results obtained were decidedly lackluster as can be seen in Figure 5.2. The best trots developed by the GA in Chapter 4 had fitness scores in the 600-800 range, so having the best trots produced by the GCRL have fitness scores below 150 was unacceptable.

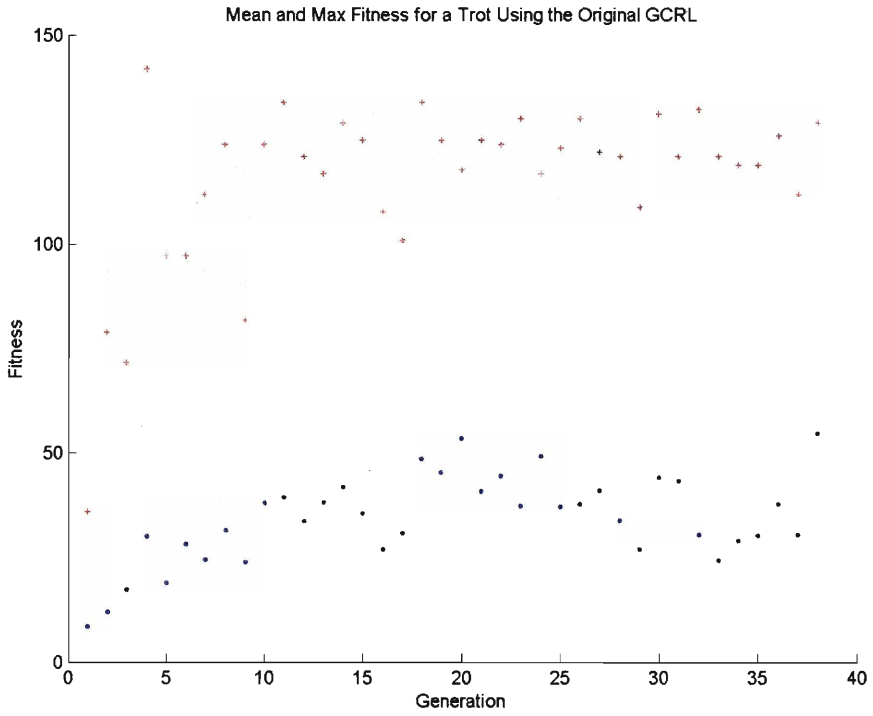

Figure 5.2: Mean and Maximum Fitness Scores for a Run Using the Original GCRL

Thus, several modifications were made to the algorithm in an attempt to improve its performance. It was theorized that a large part of the reason that learning plateaued so fast in the original GCRL was that it treated all the variables in the parameter set as independent and evaluated them across all the gaits in the generation to determine how best to modify the seed set. In reality however, while some of the variables are independent, others are not, so the variables cannot be effectively evaluated separately. Therefore, the algorithm was changed so that instead of independently checking each variable it just picked the parameter set with the highest score and used that as the new seed set for the next generation. If no parameter set from the current

generation was better than the score of the current seed set then that current seed set was kept and used to create the next generation.

The original GCRL also used a fixed step size and uniform distribution for creating the variations on the seed set that made up the population of each generation. However, since the algorithm was now choosing the best individual from each generation rather than trying to estimate the gradient and then modify the seed set, it was decided to use a Gaussian distribution with a varying standard deviation instead.

The new GCRL took the first seed set that was either generated from the random population or loaded by the user and then varied all the parameters in the seed set by a Gaussian random distribution with an initial standard deviation of 12 in order to create a new individual. This process was repeated until the entire population of 35 individuals was created and then each member of the population was evaluated in the simulator using the fitness function from Equations 4.1 and 4.2.

If a gait had been found with a higher fitness than that of the seed set, that new gait became the seed set for the next generation and the standard deviation for the distribution of the next generation was reduced by 2, making the new standard deviation 10. Every time a generation found an improved gait and replaced the seed set, the standard deviation for the next generation was reduced by 2 until the standard deviation hit a lower limit of 2. This was done so that the algorithm started off searching a fairly wide range of possible variations, but then focused its search more and more as the gaits improved.

If no gait was found with a higher fitness than the seed set, the seed set was used again to create the next generation using the same standard deviation as before. After 10 generations had passed with no improvement to the seed set, the standard deviation increased by 2 in order to widen the search area and hopefully find a better parameter set. This process was repeated for 50 generations to complete a single trial just as in the GA. Also like the GA, a total of 3 trials were run for each the trot, the canter, and the gallop to obtain the initial results.

5.3 Initial Results

5.3.1 Trot

Once again the base phase offsets for the legs were set as in Figure 2.5. The initial standard deviation for the GCRL algorithm was 12.

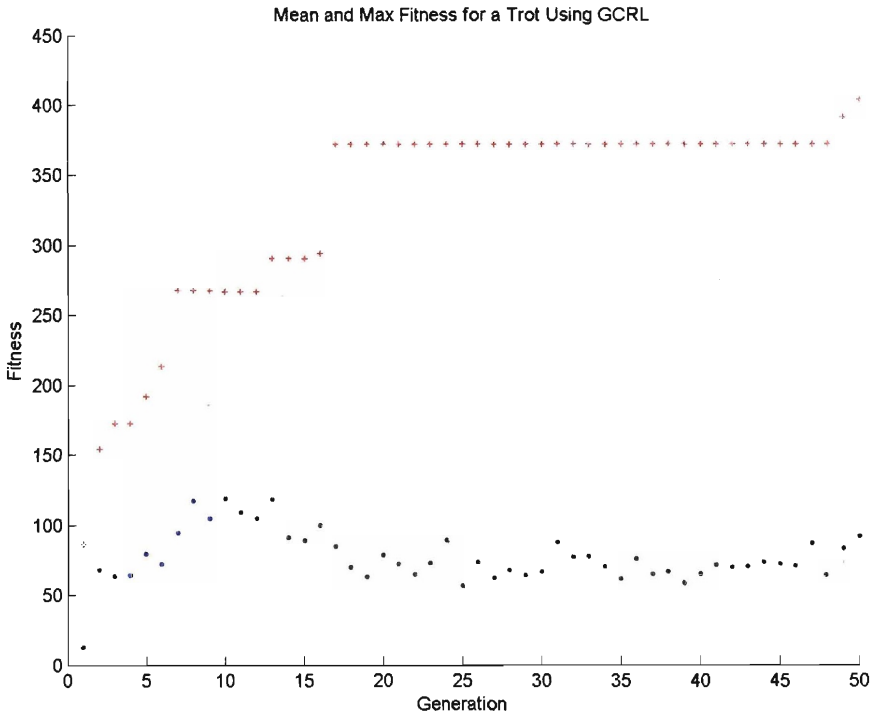

Figure 5.3: Plot of the First Run for the Trot Using the GCRL

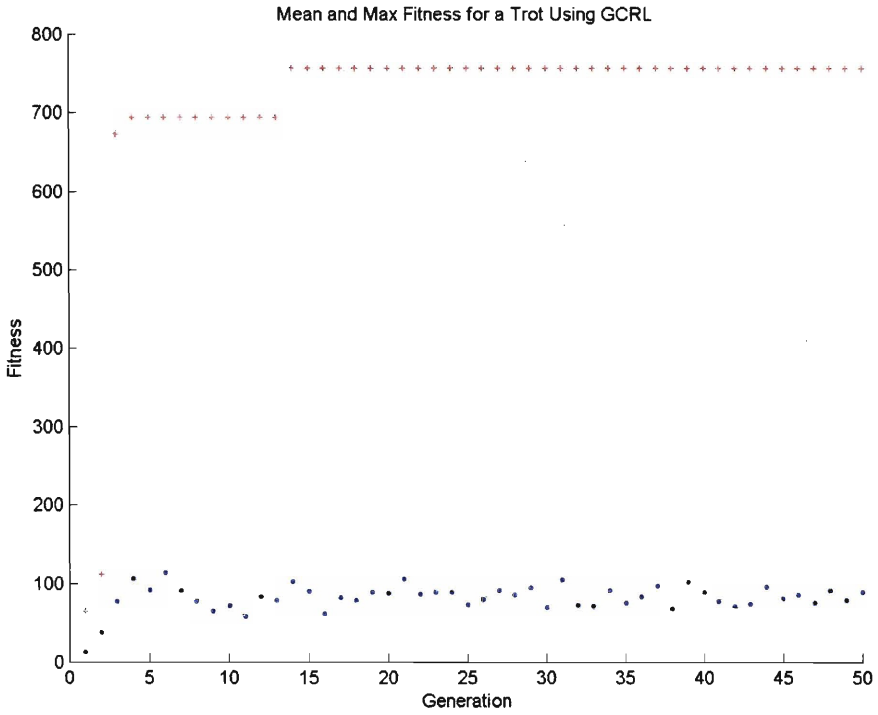

Figure 5.4: Plot of the Second Run for the Trot Using the GCRL

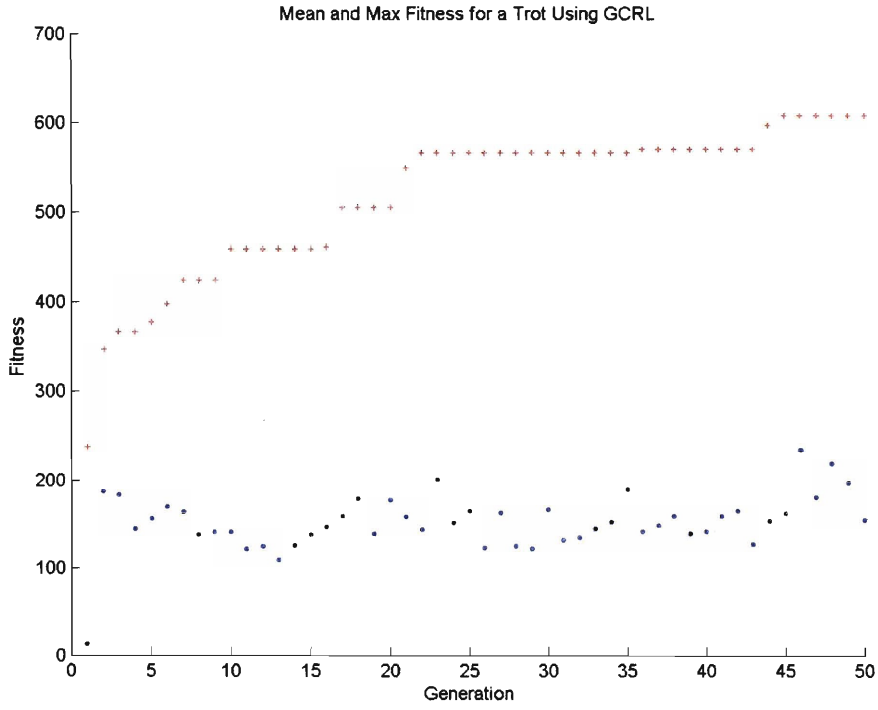

Figure 5.5: Plot of the Third Run for the Trot Using the GCRL

The GCRL appeared to be much more prone to getting stuck in regions of instability and being unable to break out of them than was the GA. This is most likely due to the fact that the GCRL does not have a mutation function to enable it search a wider range of areas, though if the standard deviation becomes large enough and the search is run long enough it can sometimes escape these regions. When the algorithm does not get a good starting seed, this inability to escape unstable regions can cause it to perform very poorly.

The first trial for the trot is a good example of this. The trial starts off fine with the mean and maximum fitness scores growing steadily, but then the means begin to go down as the search enters an unstable area and shortly thereafter both the mean and max fitness scores flat-line. However, since the maximum fitness leveled off at such a low number the algorithm is able to finally escape the region near the end of the run after the standard deviation has expanded due to the long period with no improvement. Unfortunately, this escape occurred in generation 49 and so the algorithm wasn't able to improve the gait much further, though given more time it certainly would have.

The second trial is much like the first run for the canter with the GA. The fitness quickly jumps to a very high value, but then becomes stuck in an area of instability and is unable to break free even when the standard deviation becomes larger because the current maximum fitness is so high. This trial did however produce the highest scoring gait out of all three trials for the trot in the GCRL, so it was not a bad run; it's just very unlikely that the gait could be improved upon even if the algorithm was given more time.

Trial three shows the typical progression one would expect from the GCRL algorithm with a sharp climb in the beginning which gradually flattens out as time passes. The average fitness scores stop increasing fairly early, but the maximum fitness is still climbing slightly when the trial ends so some improvement could probably still be made to this gait.

5.3.2 Canter

The base phase offsets for the legs were set as in Figure 2.7 and again the initial standard deviation for the GCRL algorithm was set to 12.

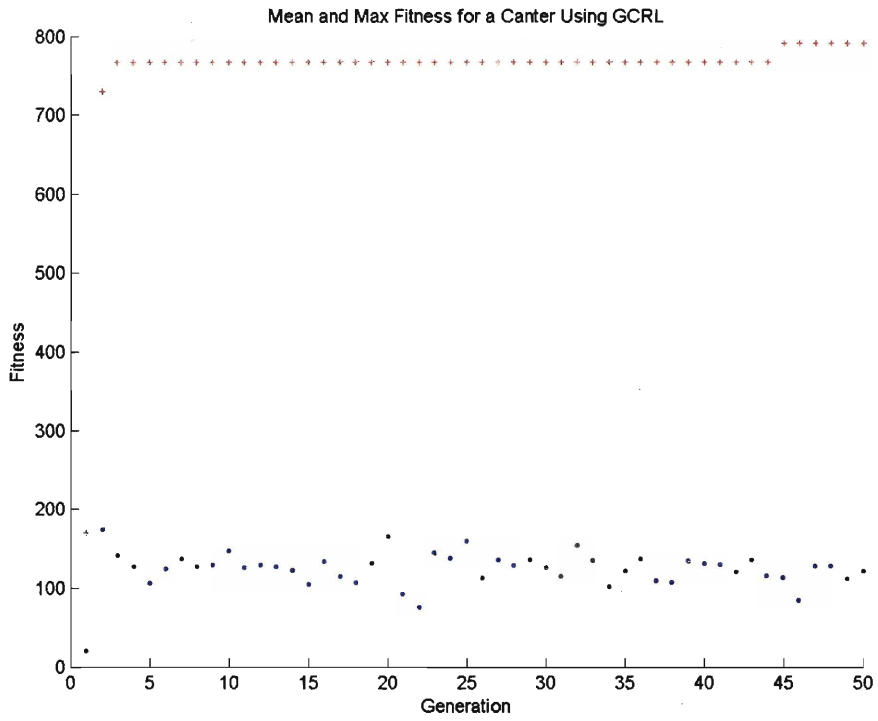

Figure 5.6: Plot of the First Run for the Canter Using the GCRL

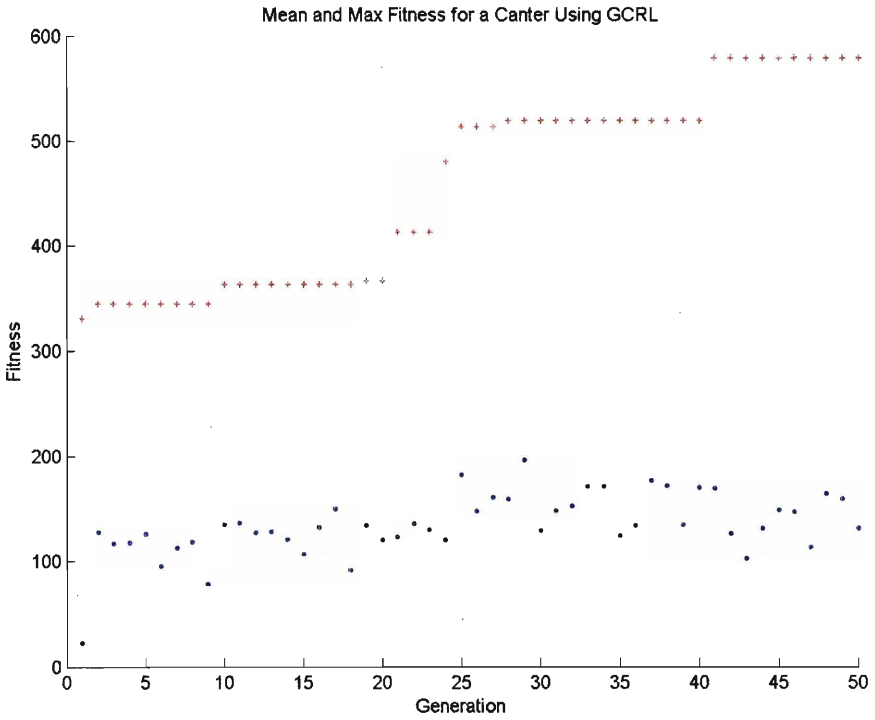

Figure 5.7: Plot of the Second Run for the Canter Using the GCRL

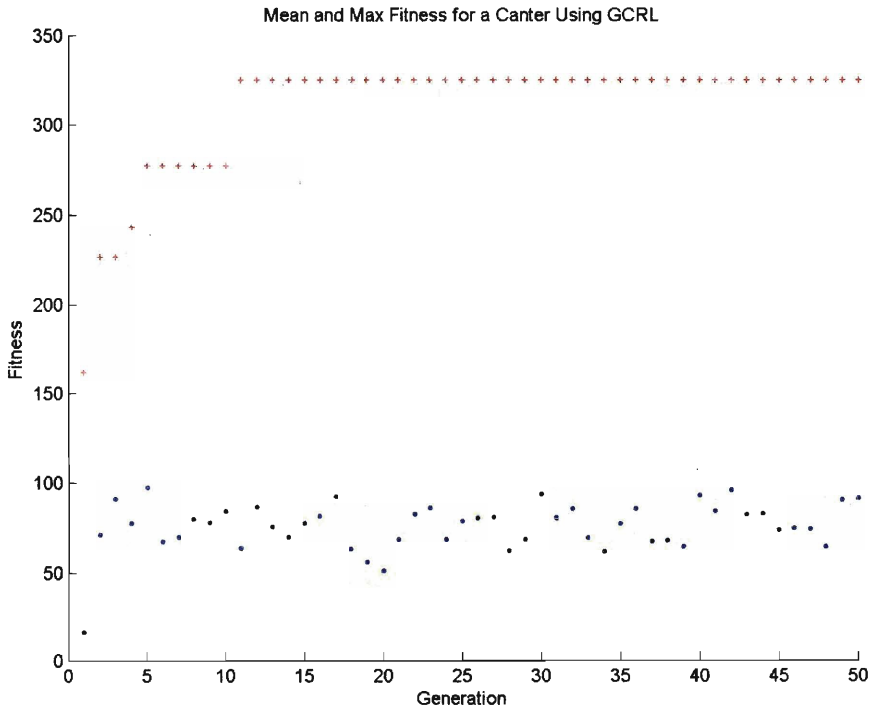

Figure 5.8: Plot of the Third Run for the Canter Using the GCRL

In the first trial, the algorithm got lucky and jumped up extremely rapidly to a high fitness value where it then stayed for most of the run. Near the end of the trial the expanded standard deviation allowed the algorithm to find a slightly better gait, but it is unlikely that the GCRL would improve the gait much more.

The second trial shows the fitness leaping up, flat-lining, and then leaping upward again. The large leaps upward roughly correspond to the points where the standard deviation increased allowing the GCRL to escape the region it had become stuck in, however once the fitness increases the standard deviation grows smaller and the algorithm becomes stuck again. This indicates that for this trial the minimum standard deviation may have been too low for the area that the GCRL was searching. If it had been higher the algorithm would have probably advanced considerably faster, so there is definitely room for improvement in this gait.

For the third trial the plot is typical of what one would expect from the GCRL with rapid advancement at the beginning and then a gradual leveling off, but the maximum fitness achieved by the trial is extremely low. Unfortunately, the fact that the fitness stopped increasing at such a low value despite the increasing standard deviation means that the algorithm just got unlucky and ended up trapped in a region with a very low maximum. It could be broken out of if the standard deviation were made large enough, but making it that large would almost be the same as redoing the random first generation and starting the entire run over. Thus, for all intents and purposes, it is doubtful that any further improvements can be made to this gait.

5.3.3 Gallop

The base phase offsets for the legs were set for the transverse gallop as in Figure 2.9 and the initial standard deviation for the GCRL algorithm remained 12.

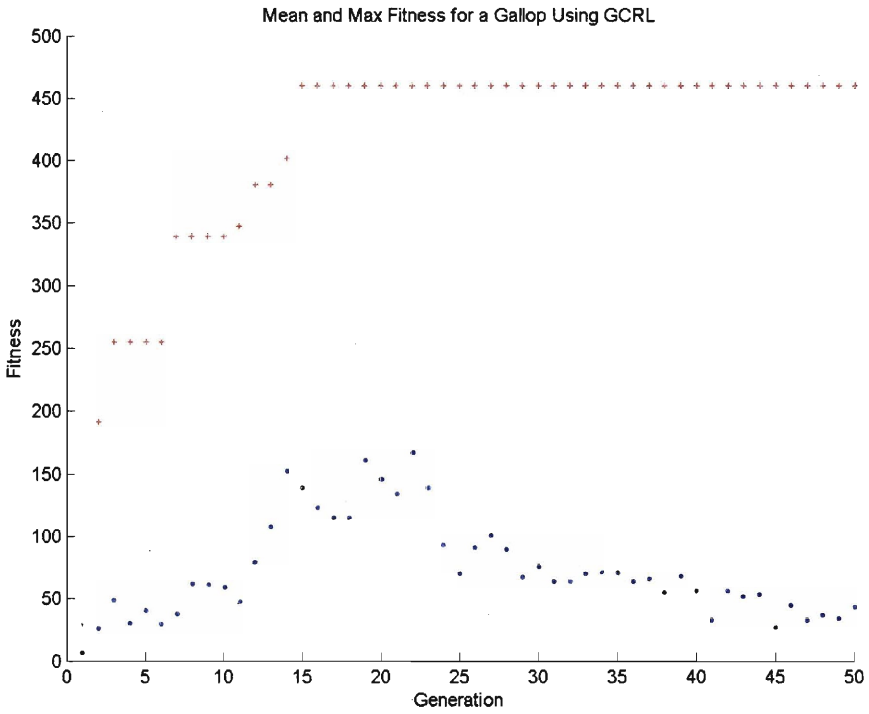

Figure 5.9: Plot of the First Run for the Gallop Using the GCRL

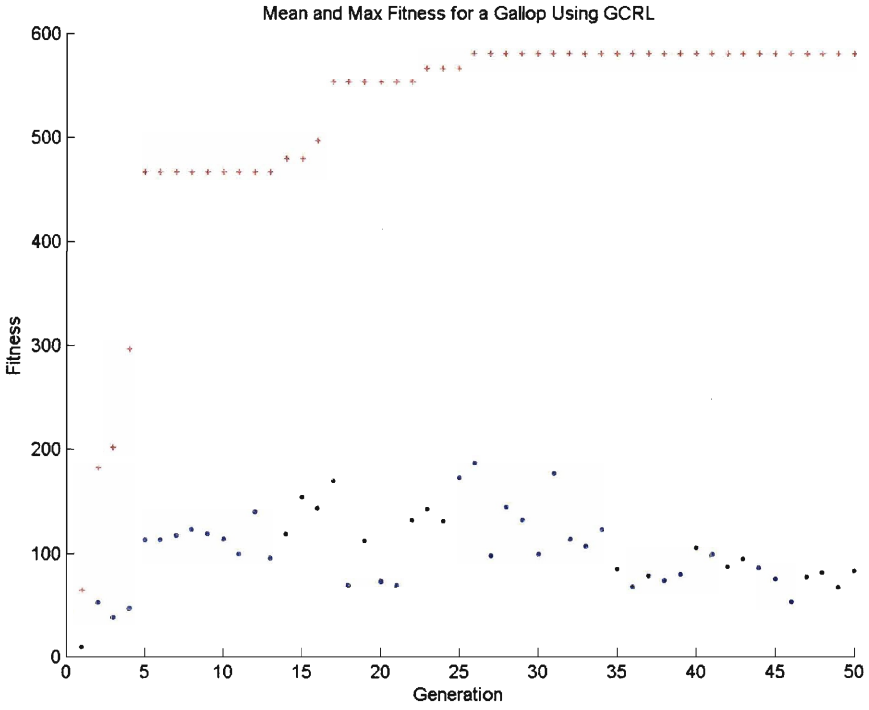

Figure 5.10: Plot of the Second Run for the Gallop Using the GCRL

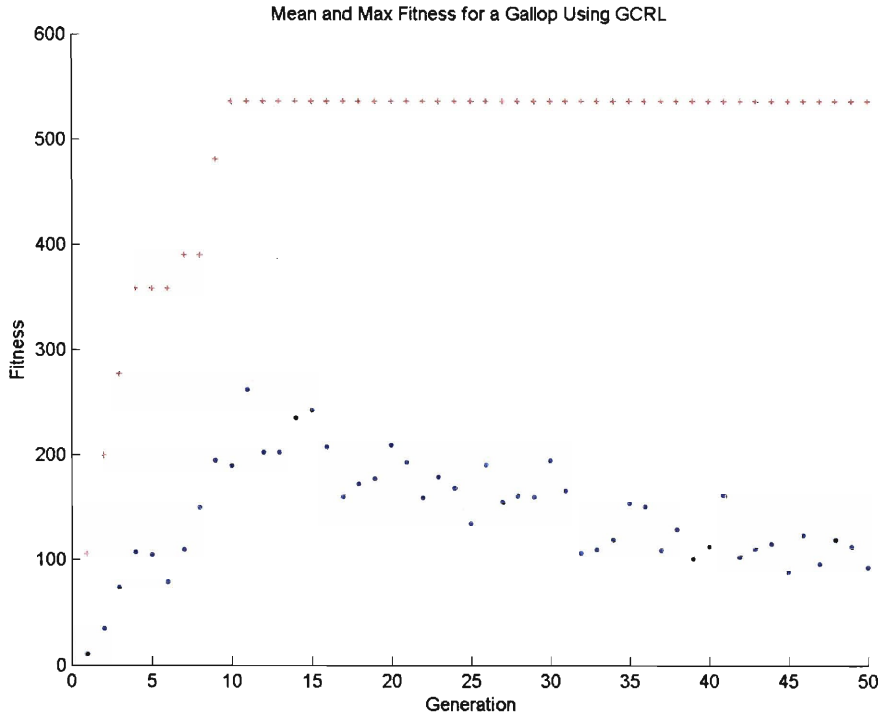

Figure 5.11: Plot of the Third Run for the Gallop Using the GCRL

In the case of the gallop, all three trials matched the typical plot for a GCRL learning algorithm, though the maximum fitness for the first trial was somewhat low in comparison to trials two and three. One can clearly see in trials one and three how the average fitness for each generation decreases after the fitness plateaus and the standard deviation begins to increase. Unfortunately, in every case no further improvements were made despite the widening of the search area, indicating that it is not very likely that any better gaits will be found in the regions that these three trials cover. As stated in Chapter 4, the reason that the maximum fitness is considerably lower for the gallop than for the trot or canter has to do with the design of the Aibo robot and will be discussed in more detail in the analysis section of this chapter.

5.4 Secondary Results

Phase two of the testing for the GCRL was like phase two for the GA in that the results from phase one were used as a starting point in phase two, but because the GCRL only uses a single parameter set to create each new generation rather than using the entire population as the GA does, the testing had to be approached somewhat differently. Since multiple gaits couldn't be used as the seed set, the highest scoring gait out of all the phase one trials for both the GA and the GCRL was used. Once the seed set was loaded the algorithm ran just as before, with the only exception being that the standard deviation started out at a value of 2 rather than 12. This was done because the seed gait was already known to have a high fitness so the search could focus closely around it in the beginning of the trial. As time passed if no improvements were made to the gait, the standard deviation could still expand outward as before.

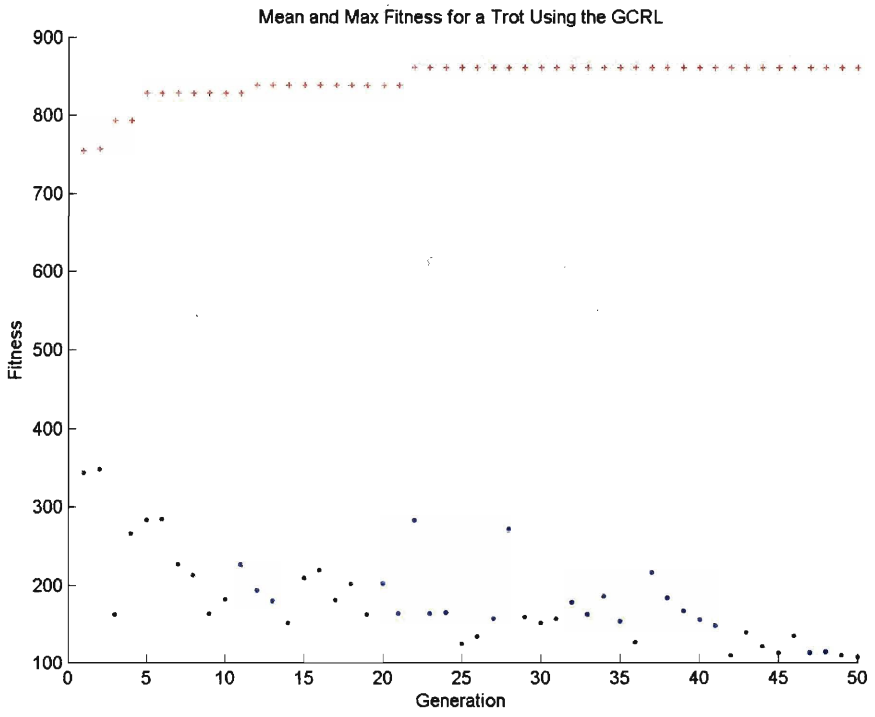

Figure 5.12: Plot of the Second Phase of Learning for the Trot Using the GCRL

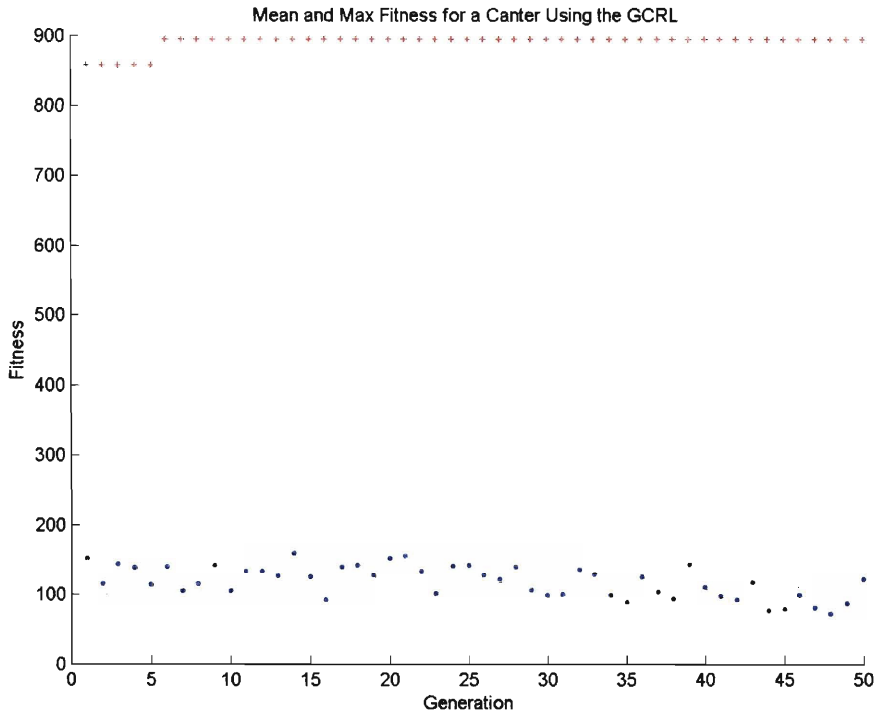

Figure 5.13: Plot of the Second Phase of Learning for the Canter Using the GCRL

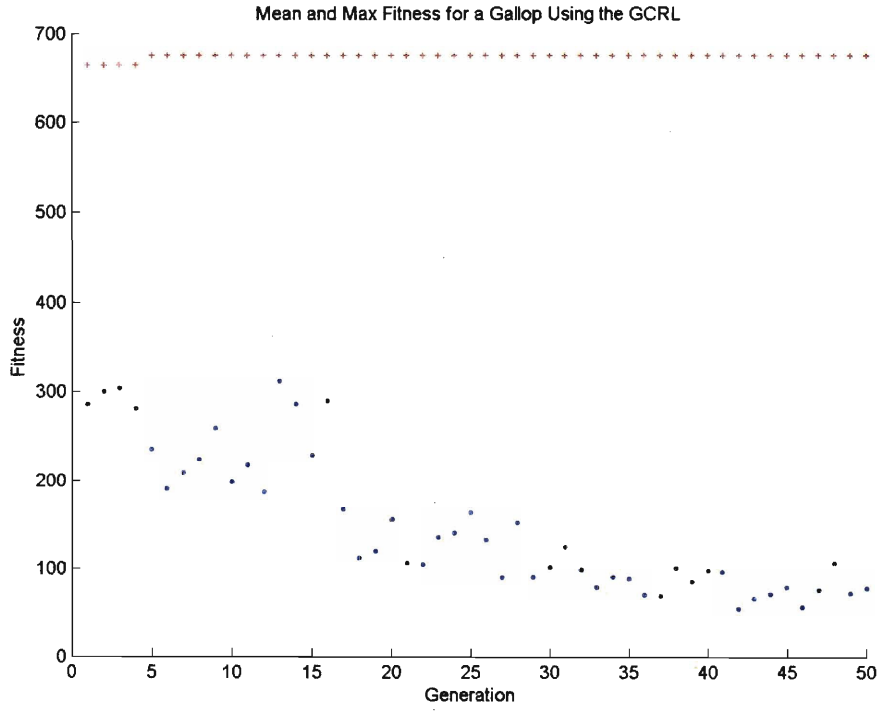

Figure 5.14: Plot of the Second Phase of Learning for the Gallop Using the GCRL

During the second phase of learning with the GCRL, the fitness of the trot gait increased markedly while the canter and the gallop only became moderately better. This is very similar to the results obtained from the GA, though the GCRL found higher fitness values for the trot and the canter than the GA did. One major difference between the results from the GA and the GCRL is that the average fitness scores for each generation during the GA runs were much higher than for the corresponding runs with the GCRL. This is probably due to the observed tendency for the speed of a gait to be proportional to its sensitivity to variations. Since the GCRL only used the fastest gait from phase one of the learning rather than using multiple gaits it was much more likely to come across a large amount of poorly performing gaits during its search than the GA. The gallop is the gait that was actually the least sensitive to variations in its parameters as can be seen from the fact that the average fitness started out very high and only started dropping off after the standard deviation of the GCRL had widened considerably. This is expected because even though the gallop is inherently a more dynamic gait than the trot and the canter, in the case of this research the gallop was much slower than the other two gaits and thus more stable.

5.5 Analysis of Gaits

5.5.1 Trot

Front Step Height (cm)	3.2
Hind Step Height (cm)	2.4
Front Cutoff Angle (degrees)	5.0
Hind Cutoff Angle (degrees)	9.5
Front Cutoff Percentage (%)	3.0
Hind Cutoff Percentage (%)	42.0
Stride Period (sec)	0.2
Ground Fraction (%)	58.0
Leg 1 Phase	0.458
Leg 2 Phase	0.568
Leg 3 Phase	0.964
Front Y-Coordinate (cm)	0.3
Hind Y-Coordinate (cm)	1.4
Step Length (cm)	8.1
Stance Width (cm)	0.5
Front Body Height (cm)	10.5
Hind Body Height (cm)	10.8

Figure 5.15: Parameters for the Trot Learned by the GCRL

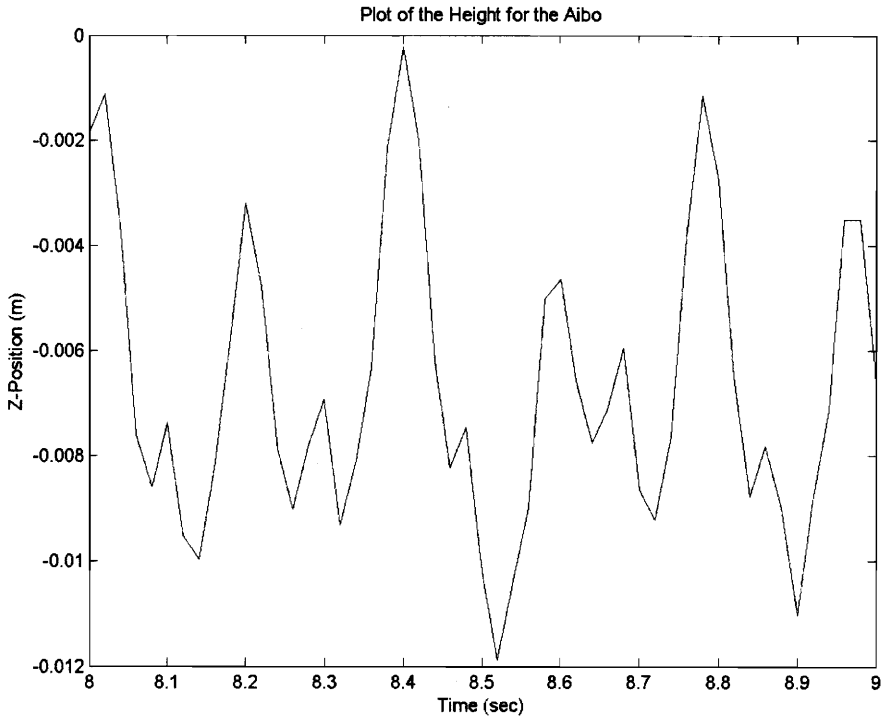

Figure 5.16: Height for the Aibo During the GCRL Trot

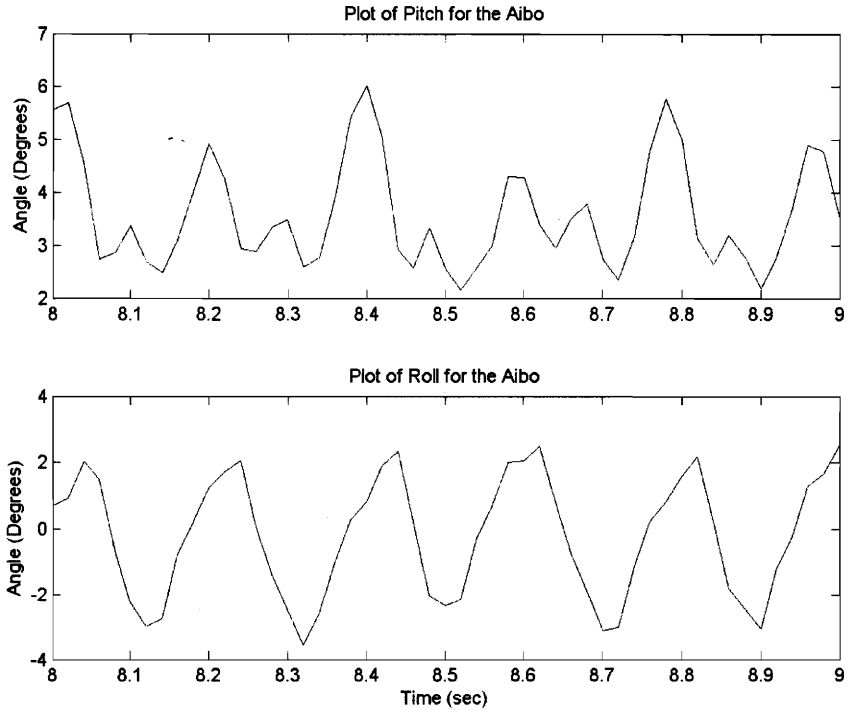

Figure 5.17: Pitch and Roll for the Aibo During the GCRL Trot

The height variation for this trot is 1.1 cm, the pitch variation is 3.9 degrees, and the roll variation is 5.5 degrees. These values are slightly higher than those for the GA trot, but not significantly so.

The robot starts out pitched up at about 3 degrees and then the pitch bounces up even further as the left front and right rear feet hit the ground. After the Aibo pitches up from the initial strike of the feet on the ground it falls back to its minimum value of 3 degrees. At that point, the right front and left rear feet touch down causing a smaller bounce up in the pitch before the robot again returns to its minimum pitch value and the process repeats. The height, as usual, mirrors the pitch and both the height and the pitch oscillate at twice the frequency of the roll, just as in the GA trot.

The body also starts with a roll value of about -3.5 degrees then tilts to the right as the left front and right rear legs come down because its center of mass is to the right of the support line. Next, the Aibo rolls all the way back to the left after right front and left rear legs hit the ground because during this stage the center of mass is to the left of the support line.

The trajectory for this trot is significantly more curved than the trajectory of the GA trot. This may be due to the fact that the right front leg is generally on the ground for less time than the left front leg which could induce a rightward curve to the Aibo's path.

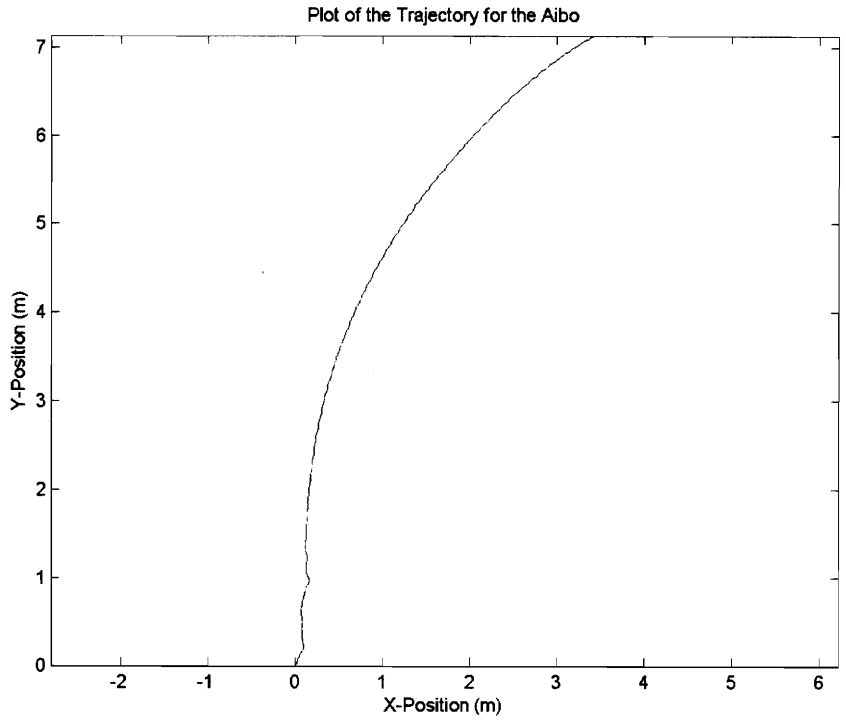

Figure 5.18: Trajectory for the Aibo During the GCRL Trot

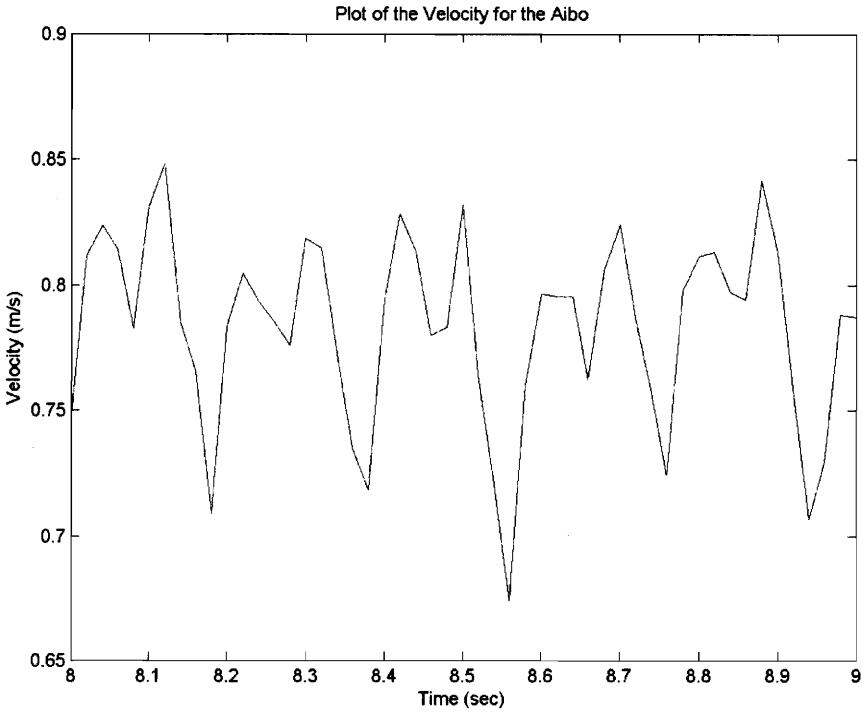

Figure 5.19: Velocity of the Aibo During the GCRL Trot

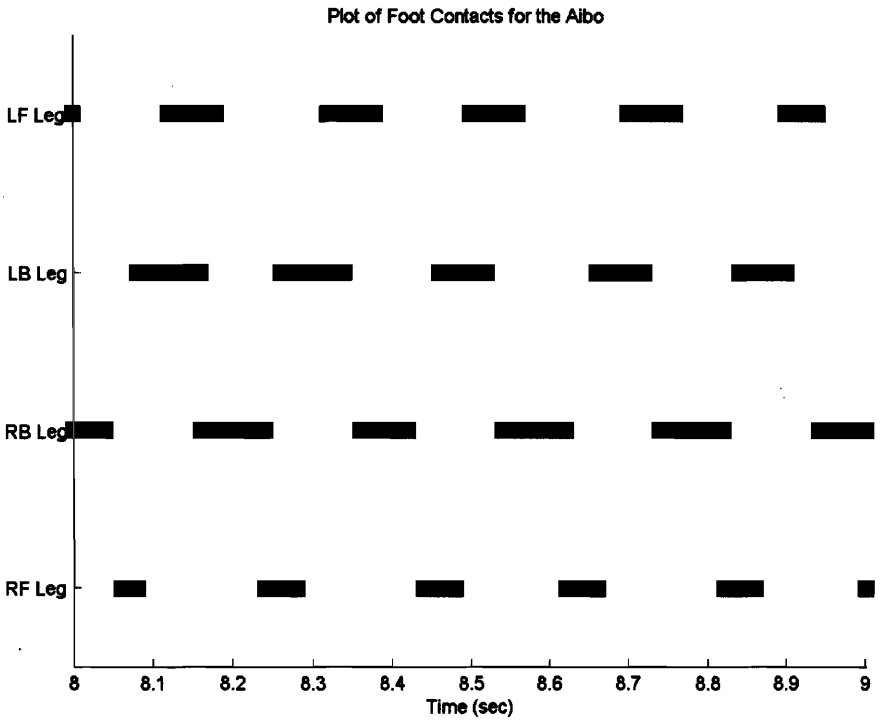

Figure 5.20: Foot Contacts for the Aibo During the GCRL Trot

The average velocity for this gait is 76.5 cm/s which is slightly faster than the GA trot. The velocity profile takes a steep dive when the front left leg hits the ground then starts rising again once the back right leg touches down. A similar pattern occurs for the other diagonal leg pair, but on a smaller scale with the velocity dropping moderately when the right front leg hits and then rising after the back left leg hits. This shows that the front legs act as a break to slow the forward acceleration of the Aibo so that it doesn't fall on its face and then the back legs increase the acceleration again as they thrust the robot forward.

The leg contacts are fairly normal, though the right back leg touches down slightly after the front left leg instead of slightly before it as it should because the slow update rate of the controller prevents it from correcting phase errors that are that small. The only unusual thing about the contacts is that the right front leg is only on the ground 32.4% of the time, while the other legs are all on the ground 47.9% of the time.

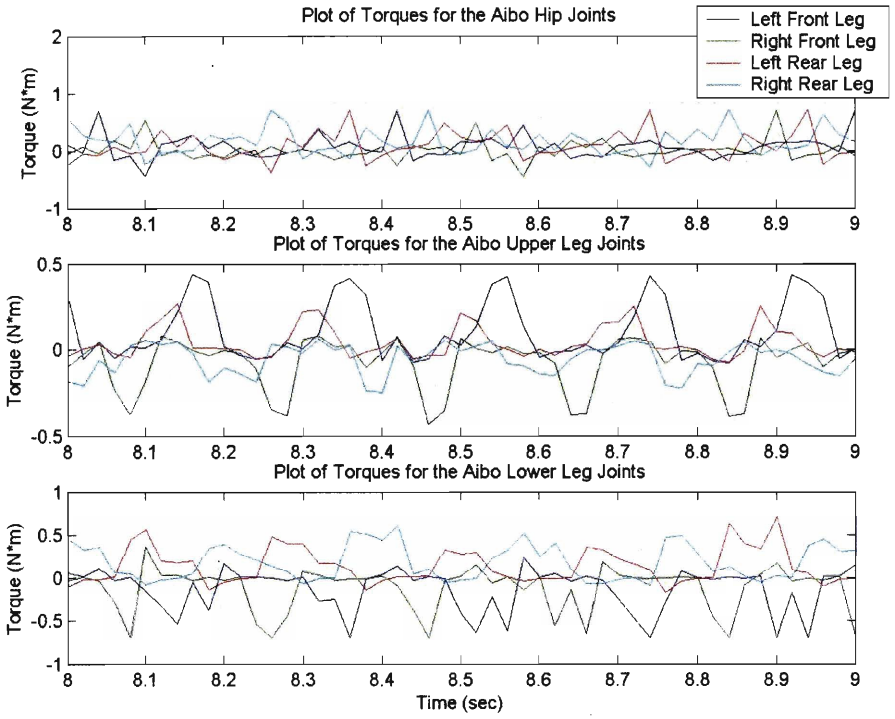

Figure 5.21: Torques for the Aibo During the GCRL Trot

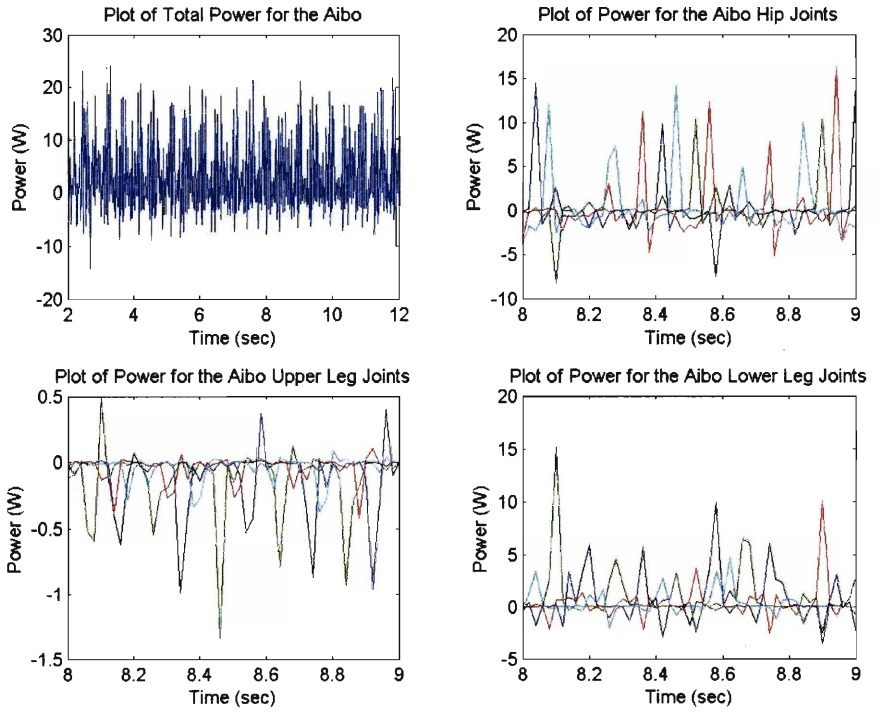

Figure 5.22: Power Use for the Aibo During the GCRL Trot

Just as in the GA trot most of the power is used by the hips swinging the legs forward, with the second most power being used by the knees as the legs move back to thrust the body forward. The power usage in this gait is somewhat higher than in the GA trot because the step length is longer while the stride period is the same, so the speeds required are higher. The upper leg joints use a minimal amount of power in order to keep the legs from splaying outwards. Most of the torque for these joints is used by the front legs since they are slowing the robot down and thus the inertia of the body causes a greater impact force on these legs and the controller requires a larger amount of torque to keep them at their specified width.

5.5.2 Canter

Front Step Height (cm)	2.2
Hind Step Height (cm)	3.3
Front Cutoff Angle (degrees)	7.0
Hind Cutoff Angle (degrees)	6.0
Front Cutoff Percentage (%)	41.0
Hind Cutoff Percentage (%)	41.0
Stride Period (sec)	0.2
Ground Fraction (%)	58.8
Leg 1 Phase	0.334
Leg 2 Phase	0.768
Leg 3 Phase	0.000
Front Y-Coordinate (cm)	1.9
Hind Y-Coordinate (cm)	0.4
Step Length (cm)	9.4
Stance Width (cm)	0.0
Front Body Height (cm)	10.4
Hind Body Height (cm)	10.3

Figure 5.23: Parameters for the Canter Learned by the GCRL

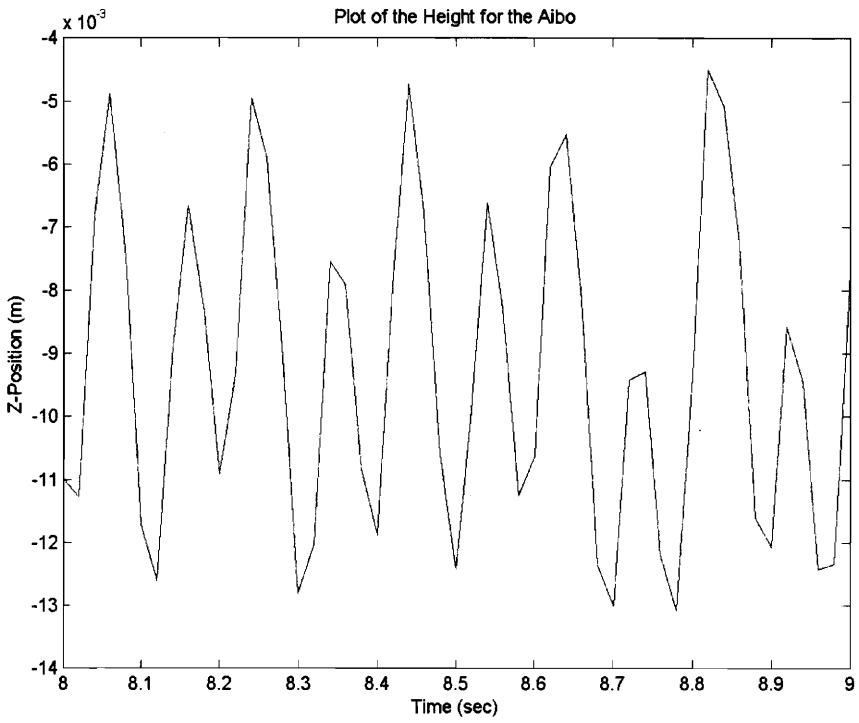

Figure 5.24: Height of the Aibo During the GCRL Canter

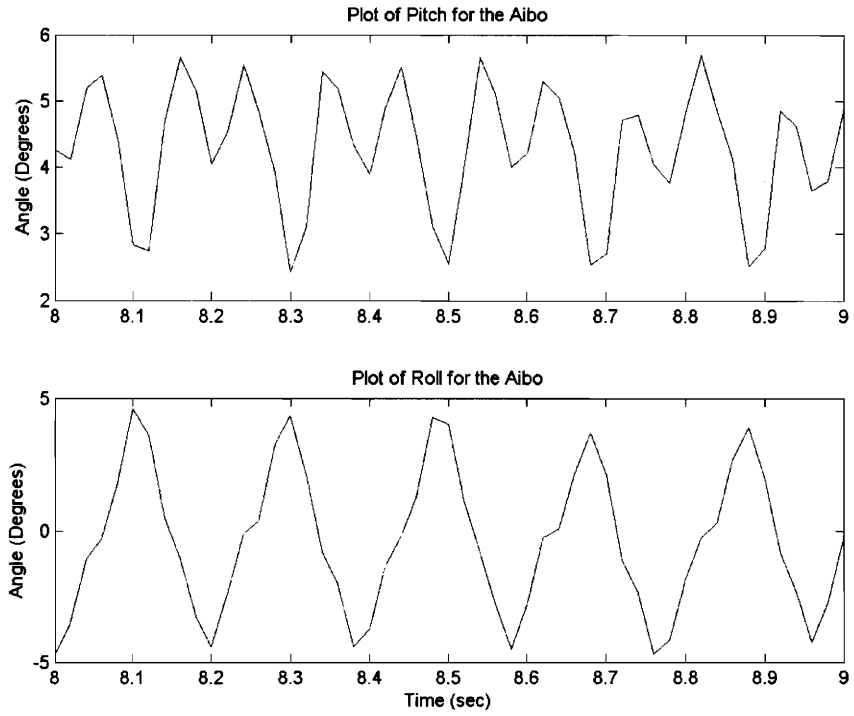

Figure 5.25: Pitch and Roll for the Aibo During the GCRL Canter

This canter is almost exactly the same as the GA canter just a little faster and straighter. The height varies by 0.85 cm, the pitch varies by 3.1 degrees, and the roll varies by 9.6 degrees. So, the height variance is a little smaller than in the GA canter and the pitch and roll values are a little higher. The average pitch is higher at 4.1 degrees and the trajectory is actually straighter, perhaps due to the larger inertia from the higher speed making the robot somewhat more difficult to turn or possibly the times the legs are on the ground for the left and right sides are more equal than they are in the GA canter. Once again, there is a sharp turn to the right at the beginning of the run before the Aibo settles in to the canter.

For an explanation on how the footfall sequence relates to the pitches and rolls of the robot see Section 4.5.2 since the GA canter is almost identical to the GCRL canter.

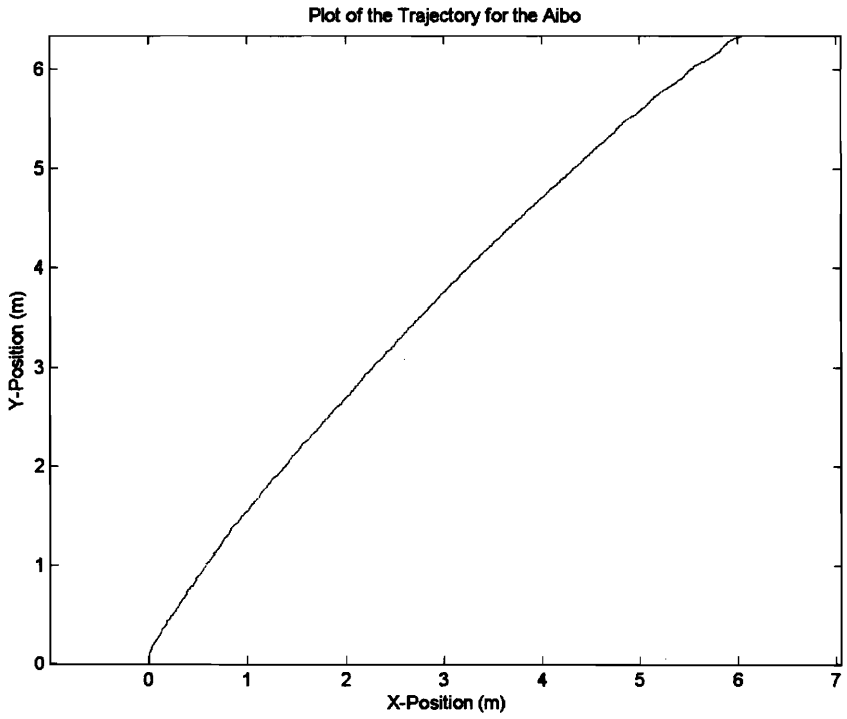

Figure 5.26: Trajectory of the Aibo During the GCRL Canter

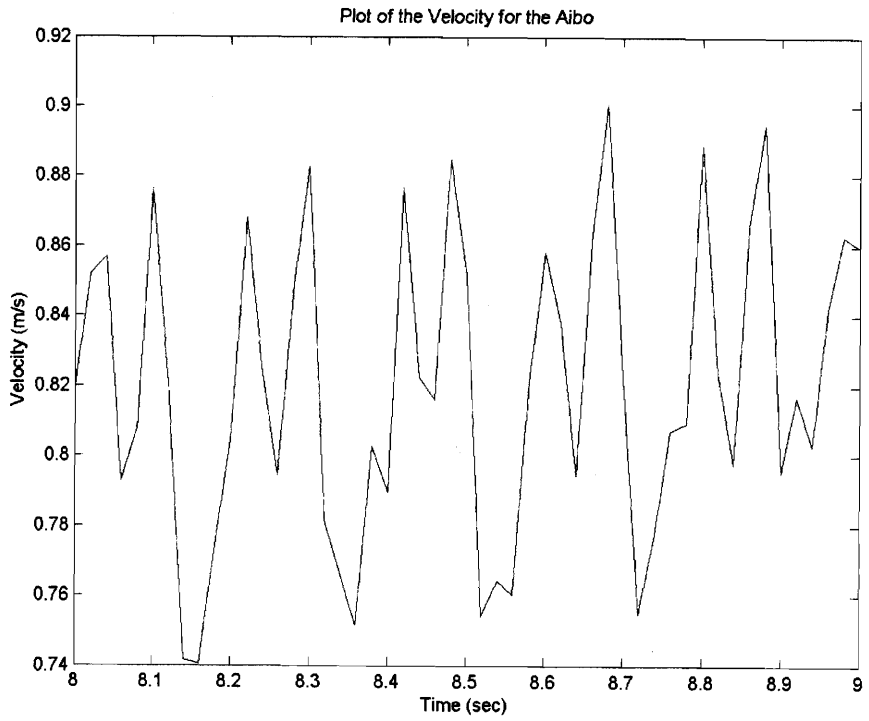

Figure 5.27: Velocity of the Aibo During the GCRL Canter

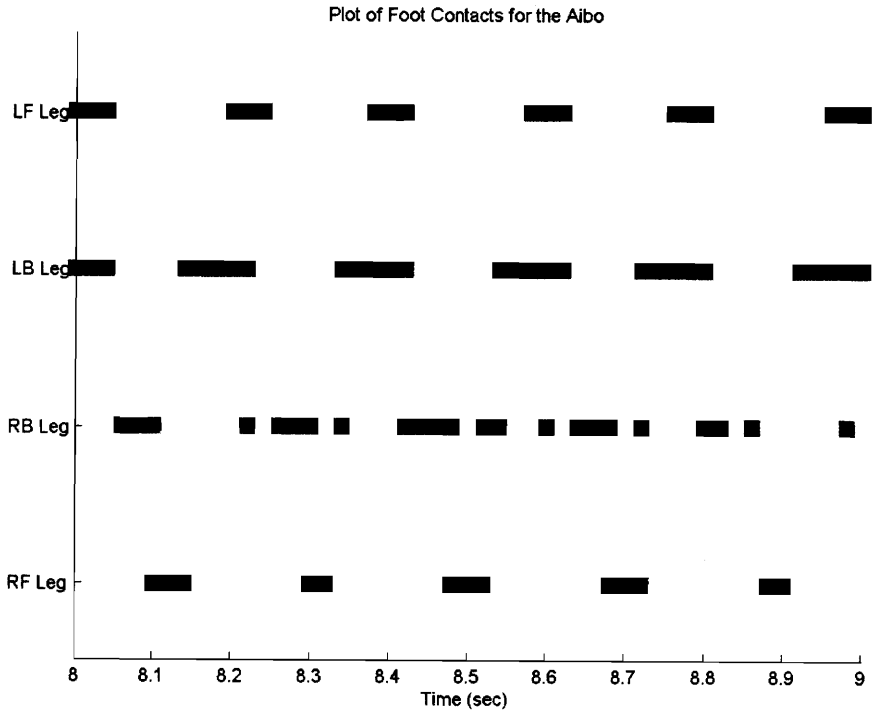

Figure 5.28: Foot Contacts for the Aibo During the GCRL Canter

The average velocity for the canter developed by the GA was 81.2 cm/s. The velocity profile of the Aibo while it is cantering is again somewhat odd due to the problems that occurred with the right back foot during the gait. Just as in the GA canter, that foot did not hit the ground with the correct phasing and when it was on the ground it bounced a lot. For an explanation of this behavior see Section 4.5.2.

Once more, the large bounces induced when the front feet hit the ground are the reason that the front feet are only on the ground for 30.9% of the stride while the rear feet are on the ground for 49.3% of the stride.

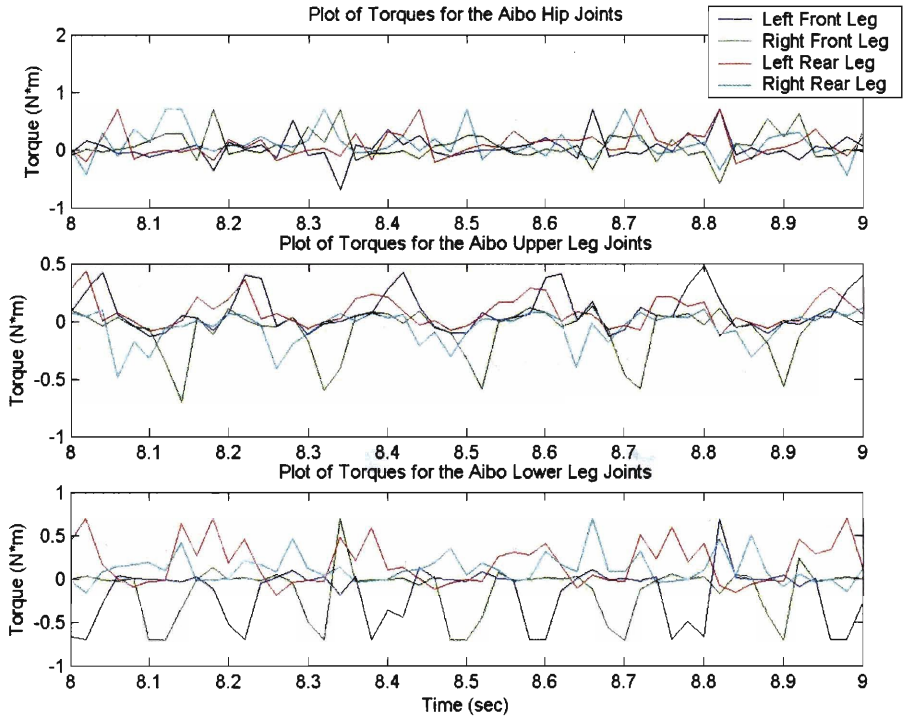

Figure 5.29: Torques for the Aibo During the GCRL Canter

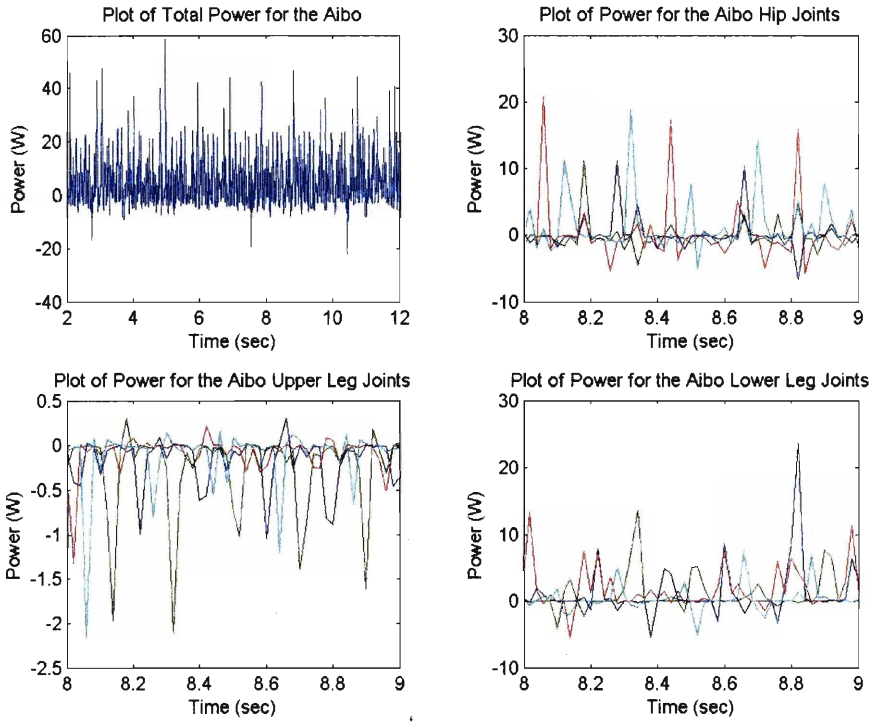

Figure 5.30: Power Used by the Aibo During the GCRL Canter

For the hip joints again most of the torque is applied during the step forward stage just as in the GA canter, but for the GCRL canter, the velocities required during this stage are slightly lower because although the step length is larger at 9.38 cm, the time to complete this stage is also larger resulting in a slower required speed overall. Thus the duration of the high torques is longer and the power levels needed are much larger. The torques for the knees during the thrust phase are very similar to those for all the other gaits, but the GCRL canter, like the GA canter has occasional spikes in the torque as it swings the legs forward due to the high speeds involved. The upper leg joint torques are again somewhat high as well because there are many points in the gait where only one leg is on the ground so a higher level of torque is required to keep the legs from splaying outward. This canter is faster than the GA canter and slightly more efficient, though it is still much less efficient than either the trot or the gallop.

5.5.3 Gallop

Front Step Height (cm)	2.0
Hind Step Height (cm)	3.1
Front Cutoff Angle (degrees)	22.0
Hind Cutoff Angle (degrees)	30.5
Front Cutoff Percentage (%)	36.0
Hind Cutoff Percentage (%)	29.0
Stride Period (sec)	0.2
Ground Fraction (%)	62.5
Leg 1 Phase	0.162
Leg 2 Phase	0.458
Leg 3 Phase	0.680
Front Y-Coordinate (cm)	1.7
Hind Y-Coordinate (cm)	1.7
Step Length (cm)	6.3
Stance Width (cm)	-0.7
Front Body Height (cm)	12.4
Hind Body Height (cm)	12.8

Figure 5.31: Parameters for the Gallop Learned by the GCRL

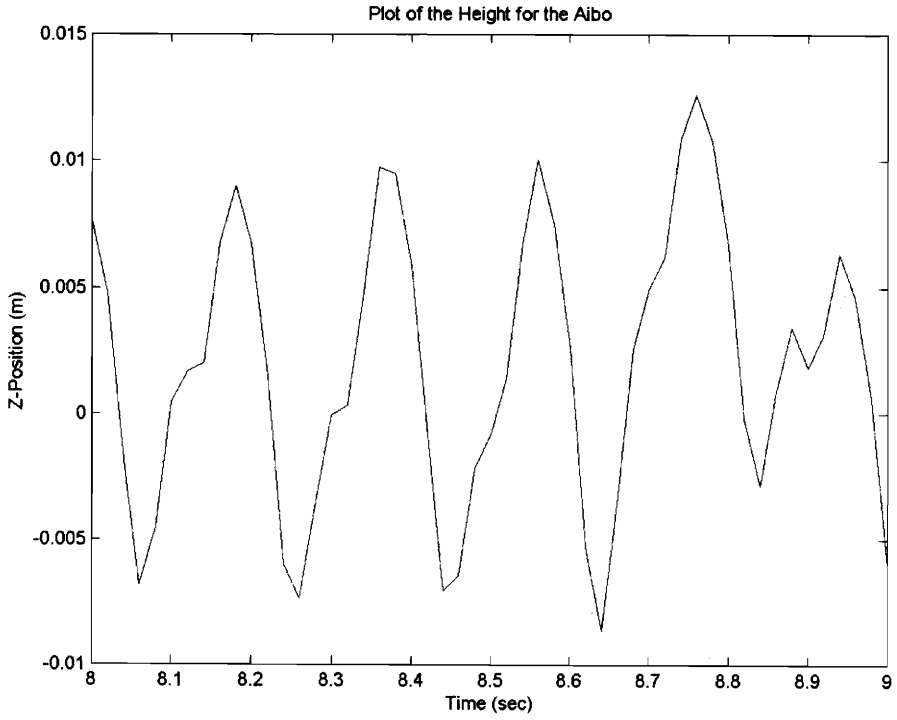

Figure 5.32: Height for the Aibo During the GCRL Gallop

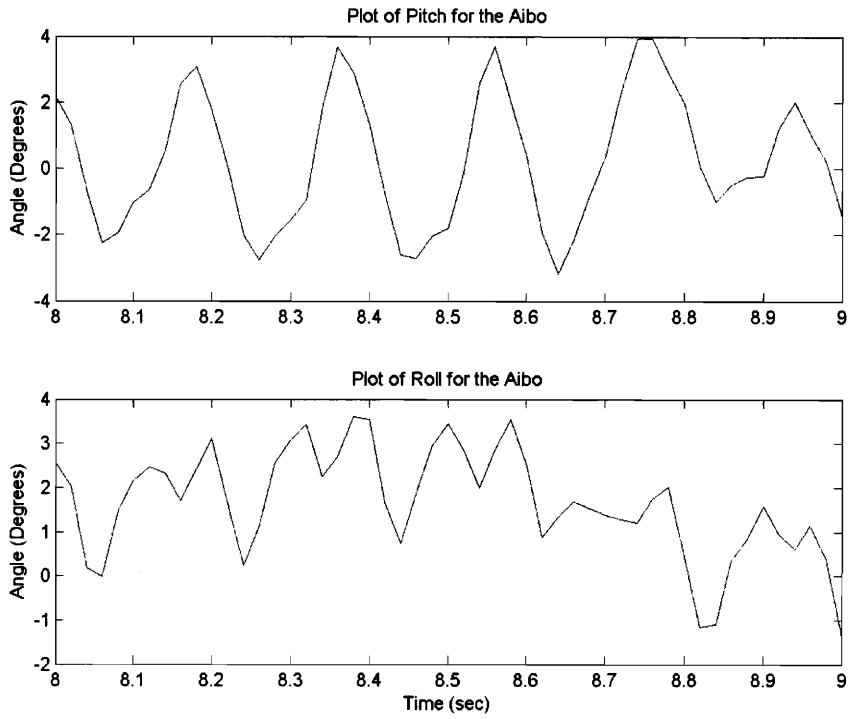

Figure 5.33: Pitch and Roll for the Aibo During the GCRL Gallop

For the GCRL transverse gallop, the height, pitch, and roll peaks all match up together fairly well just as with the GA Gallop. The data curves become somewhat irregular at about the 8.67 second mark due to a slight stumble by the right front foot, but the curves before that match the GA gallop data curves very well. The height, pitch, and roll variations are slightly less than they were for the GA gallop at 1.7 cm, 5.8 degrees, and 3.3 degrees respectively. Unfortunately, the height variation is still larger than it should be due to slow speed of this gallop.

Since the dynamics of the GCRL gallop are almost the same as those of the GA gallop, see Section 4.5.3 for a detailed explanation of how the footfall sequence affects the height, pitch, and roll of the Aibo. The only real difference in the curves for the two gallops is that the peaks for the roll data are lower for the GCRL gallop than they are for the GA gallop due to the fact that the robot is rolling and bouncing slightly less during the GCRL gallop.

Again, the transverse gallop has a much sharper curve to its trajectory than the trot or the canter, as can be seen from Figure 5.34. The reason for this is that there is

an average roll to right of 1.6 degrees for the gait and the that for both the front and the rear pairs of legs the left leg always hits first followed by the right, leading to a slight yaw that makes the robot's path curve. The Aibo is also traveling slower than in either the trot or the canter meaning it has less inertia to resist traveling in a curved path.

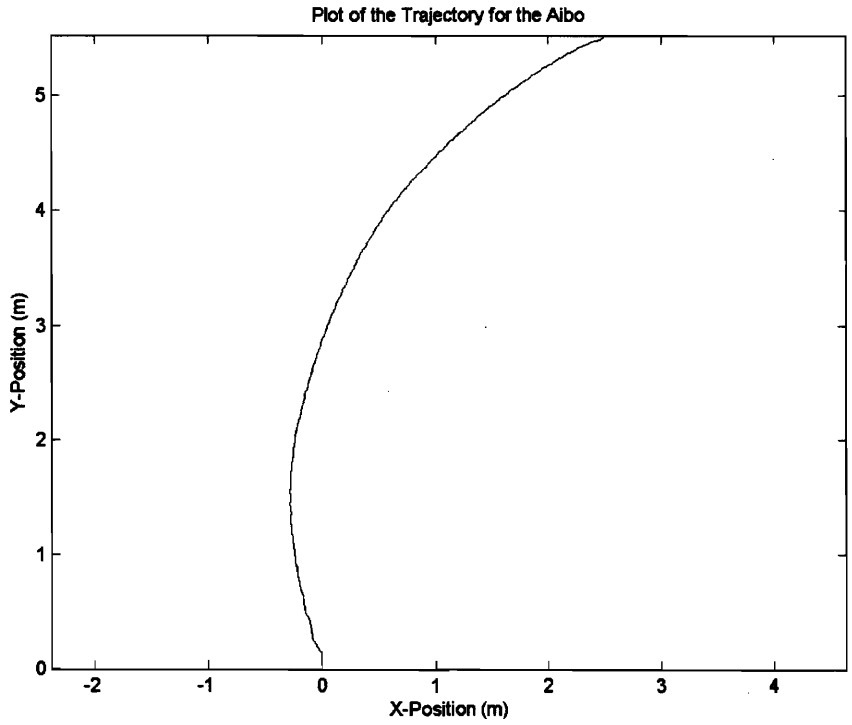

Figure 5.34: Trajectory for the Aibo During the GCRL Gallop

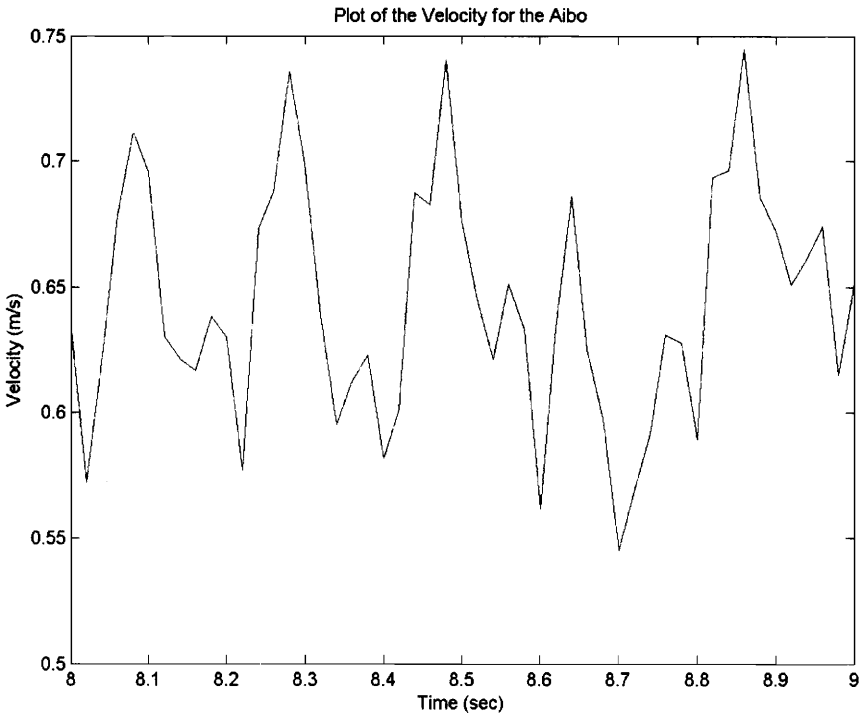

Figure 5.35: Velocity for the Aibo During the GCRL Gallop

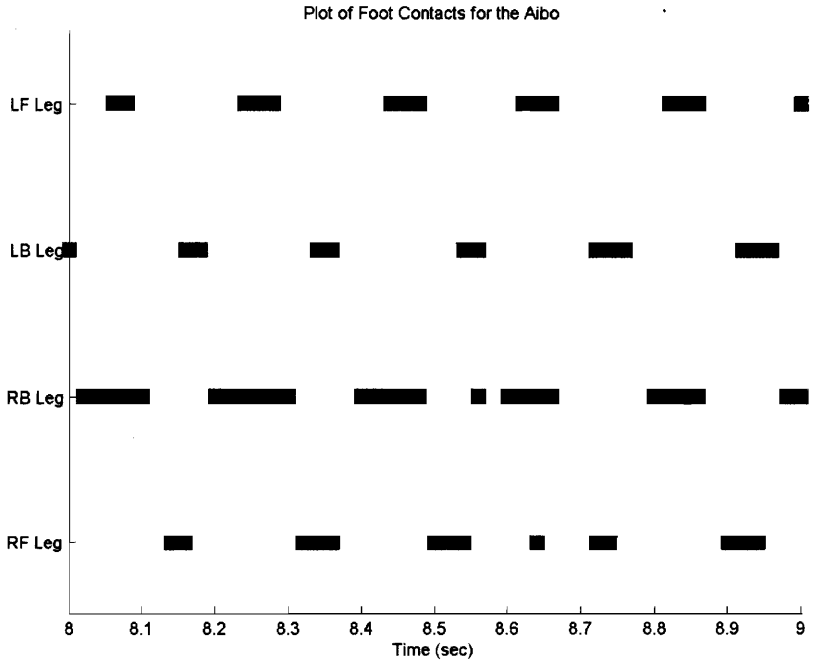

Figure 5.36: Foot Contacts for the Aibo During the GCRL Gallop

The average velocity for this gait was 61.6 cm/s, which is almost exactly the same speed as the GA gallop. Possible reasons for this slow speed are given in Section 4.5.3 as is an explanation of the gallop's velocity profile.

Once again, the right back foot is on the ground 55.3% of the time while the other feet are on the ground for only 30.5% of the time and the phase difference between the left front and right front legs is much larger than it should be. There were also short periods of flight for this gait, lasting 14.4% of the stride period on average, but as with the GA gallop, these flight periods actually occurred between the liftoff of the front left leg and the touch down of the right front leg rather than during the period between the liftoff of the front right leg and the touch down of the rear left leg as one would expect. For a slightly more in depth discussion of possible reasons for these behaviors see Section 4.5.3.

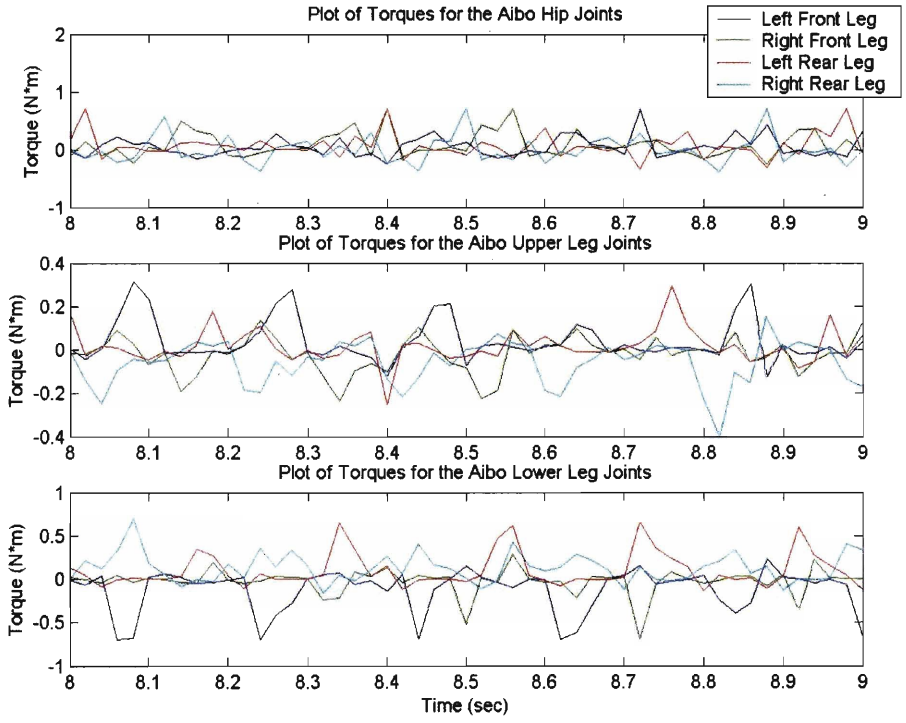

Figure 5.37: Torques for the Aibo During the GCRL Gallop

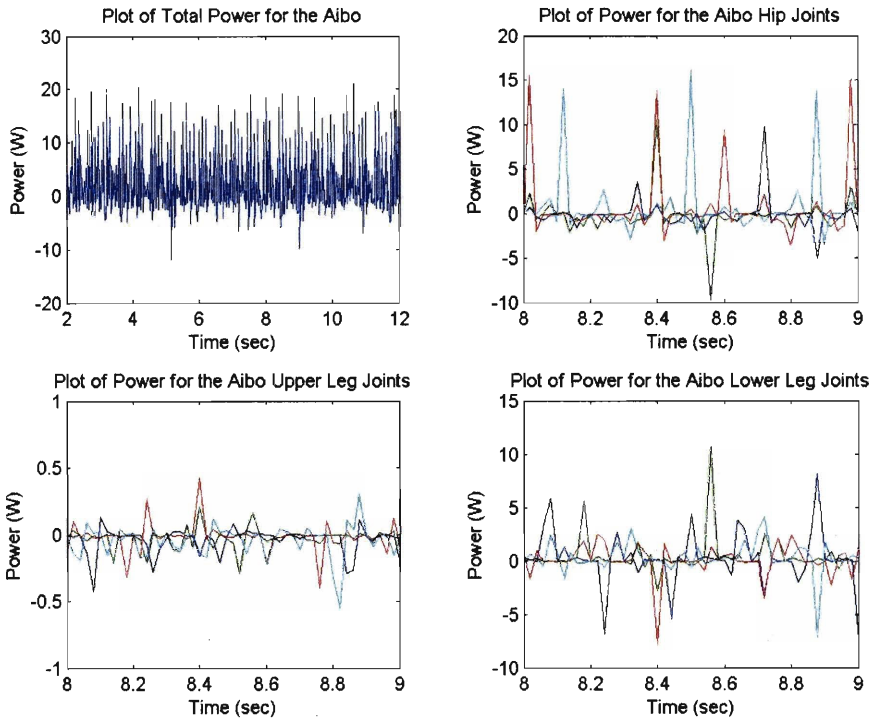

Figure 5.38: Power Used by the Aibo During the GCRL Gallop

As with all the other gaits, most of the power was used by the hips while trying to swing the legs forward during the step stage of the foot path, while the torques for the lower legs again peak during the thrust stage. The upper leg torque is rather small just as in the GA gallop because the legs are bent slightly inwards so there isn't as much torque trying to spread the legs out as there is with the other gaits. Overall, this gait was somewhat more efficient than the GA gallop because it used slightly less power during the thrust stage. This might be due to the fact that the legs stayed on the ground longer indicating that the robot wasn't bouncing quite as hard in the GCRL gait and thus the torque spikes from trying to resist these impacts weren't as high.

[This Page Intentionally Left Blank]

Chapter 6 Conclusion

6.1 Comparisons

6.1.1 Learning Algorithms

As expected, the genetic algorithm performed much better overall than the gradient-climbing reinforcement learning algorithm during the first round of testing. The GA makes use of every single individual in the population when it is creating the next generation, though their influence is proportional to their fitness, whereas the GCRL only uses the best individual from each generation in order to create the next one.

When starting from a completely random initial population, the chance that any single individual in that population is going to be near a region of the search space that contains a high fitness value is rather small. The GCRL just searches in the general area around its seed set in an attempt to find a better parameter set. If the best set from the initial random generation happens to be in an area that only has a relatively small peak fitness, the GCRL will only be able to find that local maxima.

In the case of the GA on the other hand, since it develops the next generation using multiple members from the previous generation, even if the best individual from the first generation isn't near a region with a high peak fitness, chances are that at least one of the individuals was. Thus, there is more of a chance for the GA to be able to zero in on a good area of the search space to explore than there is for the GCRL. Also, even if none of the parameter sets from the initial population are near a good area, the GA can perform mutation operations on its children, allowing it to explore other regions outside of those delineated by the initial population.

Therefore, in general, the GA is much better at searching broad swaths of the parameter space in order to locate regions with high peak fitness values than the

GCRL is, especially if the initial population is entirely random. This is born out by the results from the first phase of learning, which are shown in Table 6.1. Occasionally, the GCRL got lucky with its random starting population and managed to do rather well, but overall the GA did much better. For the trot, the GA scored an average of 100 better than the GCRL, for the canter it scored an average of 243.7 better, and for the gallop it scored and average of 49 better. The differences between the fitness scores of the two learning algorithms was related to the overall maximum speed attainable by a gait given the limitations of the Aibo and the gait controller. Thus, the canter had the largest difference while the gallop had the smallest. This happened because the GCRL had a much wider range of scores for each run than the GA did, with the GCRL scores ranging from very bad to very good while the GA scores were always relatively good. Therefore, in the case of a gait like the canter with a very high maximum possible score, the low GCRL scores pulled the average down significantly; whereas with a gait that had a low maximum possible score such as the gallop, the GCRL trials with poor scores did not pull the average down as much.

	Maximum Fitness	
	GA	GCRL
Trot 1	649	391
Trot 2	750	757
Trot 3	657	608
Average	**685.3**	**585.3**
Canter 1	849	791
Canter 2	762	579
Canter 3	788	325
Average	**799.7**	**565.0**
Gallop 1	664	460
Gallop 2	517	580
Gallop 3	542	536
Average	**574.3**	**525.3**

Table 6.1: Maximum Fitness Scores for Phase One of the Learning

For the second phase of the learning, however, the GCRL generally outperformed the GA, which was also expected. In the second phase, the learning algorithms used

the best gaits from phase one as seeds and, in this situation, the weaknesses and strengths of the two algorithms switched.

The GA had a starting population filled with high fitness individuals, but since it used multiple parameter sets to create the next generation and because the mutation factor kept altering the children so that the GA continually explored new areas of the search space, the GA was unable to focus in and closely search the high fitness region for its absolute maximum.

The GCRL, on the other hand, was able to do exactly that. It started with the highest fitness individual from phase one and used that as a base set. Then the GCRL searched in a tight region around the seed set to find a better individual, which it did in every case. This highly focused search on an area of the search space that was known to have high fitness scores allowed the GCRL to find better gaits more quickly than the GA. Undoubtedly, the GA would have found comparable gaits eventually and in the case of the gallop it actually did, but in general it was slower due to its nature of searching wide areas.

	Maximum Fitness	
	GA	GCRL
Trot	838	861
Canter	870	895
Gallop	677	675

Table 6.2: Maximum Fitness Scores for Phase Two of the Learning

Thus, the GA was confirmed to perform considerably better when starting the learning from scratch because it is better able to search large areas of the parameter space and is not as dependent on its initial population as the GCRL is. While the GCRL was shown to learn faster than the GA when starting with an initial population that had a high fitness because it focuses on small regions of the parameter space better than the GA. If the GA is run long enough however, it usually does eventually attain about the same maximum fitness scores.

6.1.2 Gaits

There are many different ways to compare the efficiency of different gaits, but the two that seem the most popular are specific resistance and travel efficiency. The specific resistance of a robot is a dimensionless number calculated by taking the average power divided by the product of the robot's weight and its average velocity as seen in Equation 6.1. The lower the specific resistance is, the more efficiently the robot is running [14].

$$\kappa = \frac{P_{avg}}{(ma)v_{avg}} \tag{6.1}$$

The travel efficiency is simply the total energy used by the robot divided by the distance it traveled. The lower this number is, the less energy the robot is using to travel a given distance and thus the more efficiently it is moving. The total energy used by the Aibo can be found by integrating the total power curve for a given run.

$$\eta = \frac{E}{d} = \frac{\int P \delta t}{d} \tag{6.2}$$

The average and integrated power values can be calculated several different ways. With many modern power supplies negative power can be regenerated making the robot considerably more efficient as long as there is enough capacitance to absorb excess energy and prevent over-voltages. If there is not enough capacitance to perform regeneration, a shunt regulator can be added in order to dissipate excess energy in the form of heat [1]. In this case, the negative portion of the power curve would become saturated at 0 W, effectively eliminating the negative powers from the curve. The final way of calculating the average and integrated power is to just use the absolute value of the power curve. To facilitate comparisons to other work, all three ways of calculating the power (regeneration, positive power, and absolute power) will be used. The results of these efficiency calculations for all of the final gaits developed

during phase two of the learning are shown in Table 6.3. For the specific resistance calculations the mass of the Aibo was taken to be 1.6 kilograms.

	GA Values			RL Values		
	Trot	Canter	Gallop	Trot	Canter	Gallop
Distance Traveled (m)	8.34	8.8	6.8	8.41	8.93	6.78
Avg Velocity (cm/s)	75.8	80.0	61.8	76.5	81.2	61.6
Avg Regen Power (W)	1.91	4.81	2.42	2.6	4.93	2.12
Avg Pos Power (W)	3.05	6.33	3.29	3.92	6.29	2.94
Avg Abs Power (W)	4.2	7.85	4.15	5.25	7.64	3.76
Regen Travel Eff. (J/m)	2.76	6.57	4.28	3.71	6.65	3.77
Pos Travel Eff. (J/m)	4.41	8.64	5.82	5.61	8.46	5.22
Abs Travel Eff. (J/m)	6.05	10.72	7.35	7.51	10.28	6.66
Regen Specific Res.	0.16	0.38	0.25	0.22	0.39	0.22
Pos Specific Res.	0.26	0.50	0.34	0.33	0.49	0.30
Abs Specific Res.	0.35	0.63	0.43	0.44	0.60	0.39

Table 6.3: Efficiency Calculations for the Phase Two Gaits

The most efficient gait found is the GA trot with an absolute specific resistance of 0.35, which is to be expected since the trots are the gaits with an average speed closest to the theoretical speed for an animal of about the Aibo's size starting the corresponding type of gait. This theoretical trotting speed was calculated to be 76.7 cm/s from Equation 2.1 using a Froude number of 0.5 and a stance height of 12 centimeters.

What is surprising is that the gallops are also fairly efficient, even though they are much slower than the expected galloping speed. In fact, these gallop efficiencies are fairly comparable to those found by other galloping robot studies though the gallops found in those studies were much faster. Marhefka [37] found a specific resistance of 0.40 (using the average absolute power) for a gallop with a top speed of 7.0 m/s for his simulation, though the robot used in that simulation was considerably larger than the Aibo. Krasny [32], who's robot ran at 4.15 m/s, found a specific resistance of 0.59 for his galloping robot simulation using the absolute power, 0.11 using positive power, and 0.03 using regeneration. The reason the positive and regeneration specific resistances are much lower than those of the GA and GCRL gallops even though the

absolute specific resistance is considerably higher is most likely due to the fact that the knees on his robot model all bent the same way and the phasing on his gallop was better.

The canters are far and away the least efficient of the three types of gaits, which is probably because of all the bouncing and large impact forces the legs are subjected to. These impacts caused large torque spikes, which increased the amount of power used, as did the extremely high speed of the step forward stage for the legs. However, since these large power increases were not accompanied by an equally large increase in the average speed, the efficiency for these canters is fairly low.

Overall, it appears that for the Aibo, trotting gaits are the most efficient gaits for traveling at high speeds. The Aibo does not have the necessary torque, design, and sensors for it to effectively perform faster gaits such as the canter or the gallop.

6.2 Lessons Learned

The Aibo does not appear to have the capabilities to truly use dynamic gaits faster than a trot. This is due to a variety of factors involving the basic design of the robot. One of the Aibo's biggest problems is its size; it is so small and its legs are so short that it would need to have extremely low stride times in order to either gallop or canter. If the robot were larger, with longer legs the required stride times would be much larger and as a bonus, the top speed that the robot could attain would be significantly higher. A larger robot would also allow for motors with higher torques and allow room for electronics that would enable the robot to regenerate power in order to make it more efficient. In addition, more room for electronics would mean that upgraded processors or data communications could be put on the robot so that the controller update rate was high enough that the legs could follow the designated foot paths better and maintain more accurate phase differentials.

Some design changes that would help the robot run more effectively would be to have the knees on both sets of legs bend the same way, preferably backwards as the

front legs on the Aibo currently do. As the Aibo is presently designed there were some issues with leg interference as the robot ran and these issues would become more noticeable as the gaits got faster and the stride length increased. Another design change that would help minimize the bouncing and torque spikes that occurred in both the canter and the gallop would be to add torsional springs in series with the motors on the knees in order to help absorb the ground impacts. This change would also increase the efficiency of the robot by allowing it to store and release the impact energy helping it achieve the flight periods necessary for faster gaits [41]. Having springs in series with the knees enables energy to be rapidly injected into those joints during the thrust stage and then gradually released as the leg moves backwards along the ground as well, which has been found to be quite effective for galloping in previous studies [32, 34].

The final change that would help the robot when using dynamic gaits would be to add two downward facing range sensors to the robot's body: one near the front left hip and the other near the back left hip. It is far more effective to preposition the legs at the desired forward position and use the height of the foot off the ground to trigger the early retraction stage than it is to just have the legs trace out their given foot path without any outside sensor input. For example, Krasny's [32] gallop would start by swinging the front legs forward to designated coordinates in a fixed amount of time. The legs would then stay there until the left front foot reached a specified height above the ground at which point it would begin the early retraction stage, followed by the thrust stage when the leg hit the ground. The right front leg would start its early retraction stage a certain amount of time after the left leg to maintain the proper phasing. Once the front legs were both on the ground, the rear legs would be in the air and they too would move forward to designated coordinates in a set amount of time and then stay there waiting for a trigger. The left back leg triggered when either the proper amount of time had passed according to its assigned leg phasing, or when the left back foot reached a certain height above the ground. The right back leg triggered a set amount of time after the left back leg. This system enabled the robot to adapt to slight variations in its roll and pitch as it ran without suffering the bouncing that occurred with the Aibo during the canter. It was also very effective at maintaining the

correct leg phasing for the gait and could be applied to both the canter and the trot as well as the gallop.

In terms of the gait learning process itself, some improvements could also have been made. In this thesis' research the robot started from a stand still and then began running. It might have been more effective to have the simulation start in the middle of the run (with the appropriate initial speed, height, and leg positions), so that one didn't have to worry about getting the robot to start moving and then accelerating up to speed. Once a good gait had been found, a different learning process could be performed to get the robot to transition from standing to the full running gait.

Chapter 7 Summary and Future Work

7.1 Summary

As more legged robots have begun to be developed for their obvious advantages in overall maneuverability and mobility over rough terrain and difficult obstacles, their shortcomings over flat terrain have become more apparent. Wheeled vehicles can cover flat terrain very efficiently and at a high rate of speed while legged platforms have been forced to plod along at nearly the same speeds they use to traverse rough terrain. This is primarily due to the fact that virtually all legged robots use a very stable, very slow walking gait to move, regardless of whether the ground is flat or rough.

The simplest way of solving this problem is to use the same method as legged animals: simply change the gait from a walk to a faster more dynamic gait in order to increase the robot's speed. It would be extremely useful if legged robots were capable of moving across flat ground at speed while still retaining their ability to cross extremely rough or broken ground. Currently, legged robots are not used nearly as much as wheeled ones in large part due to how extremely slow they are. The ability to utilize dynamic gaits to achieve speeds nearer to those of their wheeled counterparts would close this gap and make legged robots far more useful and versatile than they currently are.

For this thesis' research, a simulation of the Sony Aibo robot was used as a platform because it is one of the most capable and widely used quadruped robots. Unfortunately, programming dynamic gaits is extremely difficult, so evolutionary algorithms are used to learn the proper parameters to produce different types of gaits. The two algorithms that are used in this thesis are the genetic algorithm (GA) and a modified gradient-climbing reinforcement learning algorithm (GCRL).

The thesis begins by providing some background information on various types of gaits, focusing specifically on the trot, the canter, and the gallop. It then details some of the other research that has been done with dynamic gaits as applied to robotics.

Next, it describes the support programs that were written so that the machine learning algorithms could be applied to the problem of gait generation, focusing first on the simulation of the Aibo and then on the gait program that was written to take in a set of parameters and output the commands necessary for the simulated robot to move.

The two learning algorithms used in this research were chosen due to their very different methods of operation. The GA breeds and mutates multiple individuals from one generation in order to create the next, while the GCRL creates many variations on the single best individual from the previous generation in order to create the next generation. The GA tends to be better at doing a comprehensive search of a large area in the parameter space and the GCRL tends to be better at doing a very thorough search of a small region in the parameter space.

For each of the two algorithms there were two rounds of learning: in the first, gaits were generated using a random starting population, and in the second, the best gaits from several trials during the first round of learning were used as the starting population. The dynamics of the gaits generated from the second round of learning were then analyzed.

The results from the two algorithms for both rounds of learning were compared and analyzed as well. It was found that as expected the GA did much better during the initial phase of learning because it is excels at doing wide-ranging searches, while the GCRL did better during the second round of testing because it was able to search the area around the high-fitness gaits more comprehensively.

The efficiencies of the different types of gaits were also compared in order to determine the overall best type of gait for high-speed running with the Aibo. This turned out to be the trotting gait. Neither the canter nor the gallop could be performed by the Aibo very well due to a variety of problems with the robot's design. Finally, the thesis identifies these design problems and proposes possible solutions to rectify them on future quadruped robots.

7.2 Future Work

7.2.1 Improving the Controller

The controller utilized in this research is able to handle a wide variety of different gaits, but can only perform a single gait at a time. It would be quite useful to create a controller that was capable of modifying the robot's speed on the fly, transitioning between different gaits, and performing high speed turning and trajectory control.

Modifying the speed of the robot as it traveled would probably not be excessively difficult and could be accomplished by having the evolutionary algorithms learn the best gaits for a series of set speeds. Then to alter the robot's speed as it ran one would most likely just need to load the parameters for the different gaits in at a rate of a new gait every stride or two to transition between the gait for the current speed to the gait for the desired speed. For example, one could have the robot learn the optimum gaits for different speeds at 5 cm/s intervals. To change from running at 80 cm/s to running at 60 cm/s would then require loading the 75 cm/s gait, waiting until the speed had dropped to 75 cm/s, loading the 70 cm/s gait, waiting until the speed dropped to 70 cm/s second, etc. This approach should work because though the gaits would have slightly different speeds, their leg phasing would be about the same so they would be easy to transition between, though rapid speed changes would be more difficult.

Making a controller that could transition between different kinds of gaits, such as changing from a trot to a canter would be considerably harder. One possible simple approach might be to just have the robot make a relatively large bound and then change the leg phasing to the new gait while the robot is in the air. Depending on the speed that the legs can move and the size of the bound required this might work or it might not. Research into how biological quadrupeds transition between gaits would most likely be helpful when attempting to design this type of controller.

Having a real-time trajectory and balance controller that could be either placed on top of the gait controller architecture or integrated into it is another element that will be required to make dynamic running useful on actual robots. Without the ability to

turn at high speeds while the robot is galloping or to recover from an unexpected slip dynamic gaits will not be able to be used to perform real-world tasks.

One area of research that has received almost no attention is trying to develop a dynamic gait that can operate over relatively rough terrain. This is an extremely complex problem that would require all of the previously mentioned control elements in addition to a heavy integration of sensor data into the controller.

7.2.2 Transition to Actual Robots

It would be good to try out the various gaits found in this thesis' research on an actual Aibo to see how well the simulation matches with reality and whether any unforeseen problems occur with the gaits due to the randomness and imperfection of real-world conditions. This could also help determine how accurate the various torque and power requirements are and whether the calculated efficiencies hold true in reality.

A more interesting problem would be to try and build a robot using the recommendations from Section 6.2 that was truly designed for dynamic running and then developing gaits for that robot in simulation and evaluating them on the hardware. This would require a considerable amount of time and effort, but would probably prove very rewarding in the end considering the distinct lack of many actual robots that are capable of high speed dynamic gaits.

7.2.3 Incorporating Real Time Learning and Sensory Input

Both of the evolutionary algorithms used in this thesis' research were offline algorithms. Meaning that they do a full simulation run for various parameter sets, get the fitness scores at the end of the run, and finally, create new parameter sets based on

those fitness scores. The problem with these offline methods is that they can only learn gaits for conditions that have been simulated for them and once they are placed on the actual robot they cease to evolve.

Online learning methods, on the other hand, learn and adapt in real time as the robot runs. There are no separate trials for different parameter sets, the robot just learns about new situations as it experiences them. Using a temporal difference learning algorithm combined with a neural network is one popular method of online learning. It maps different states to outputs in real-time and when it comes across a state it hasn't seen before, it uses past experience to make a guess about the proper outputs and learns from the results of that guess. This means that it is always learning and it can adapt to new situations as they arise. One of the problems with online learning methods is that they require sensor data so that they can determine what state they are in and sensor data can often be noisy, which can inhibit the learning process. Another problem is determining how best to parameterize the state of the robot so that all of the relevant information is included without making the state space too large.

Adding in sensory input to the controller at some point will be a necessity if dynamic gaits are ever going to be used on robots in the real world. They will be required for online learning, balance control, trajectory control, travel over rough terrain, and to allow the controller to make decisions about transitioning between different speeds and types of gaits. Yet little or no research has been done on what kind of sensor data would be important for the controller to have access to and how to best integrate noisy sensors with a gait controller.

[This Page Intentionally Left Blank]

References

[1] Advanced Motion Controls, Inc., "Power supply selection document" [online], Camarillo, CA 93012, 2003, URL: http://www.a-m-c.com [cited 3 May 2006].

[2] M. Albrecht, T. Backhaus, S. Planthaber, H. Stöpler, D. Spenneberg, and F. Kirchner, "AIMEE: A Four-Legged Robot for RoboCup Rescue," *Proceedings of the 8th International Conference on Climbing and Walking Robots (CLAWAR)*, pp. 1003-1010, London, U.K., 2005.

[3] K. Arikawa and S. Hirose, "Development of Quadruped Walking Robot TITAN VIII," *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pp. 208–214, 1996.

[4] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford: Oxford University Press, 1996.

[5] J. Baxter and P. L. Bartlett, "Infinite-Horizon Policy-Gradient Estimation." *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

[6] S. Boccuzzo and D. Steiner, "Artificial and Natural Walking Machines: Rapid Locomotion," *Artificial Intelligence Seminar 2002* [online], URL: http://www.ifi.unizh.ch/ailab/teaching/ANWM_Seminar/StudentMaterials/rapidloc.pdf [cited 3 May 2006].

[7] S. Chernova and M. Veloso, "An Evolutionary Approach to Gait Learning for Four-Legged Robots" [online], September 2004, URL: http://www.cs.cmu.edu/~coral/papers/iros04-chernova.pdf [cited 5 May 2006].

[8] R. M. Chigliazza, R. Altendorfer, P. Holmes, and D. E. Koditschek, "Passively Stable Conservative Locomotion." submitted to *SIAM Journal of Applied Dynamical Systems*.

[9] K. S. Espenschied, R. D. Quinn, R. D. Beer, and H. J. Chiel, "Biologically Based Distributed Control and Local Reflexes Improve Rough Terrain Locomotion in a Hexapod Robot," *Robotics and Autonomous Systems*, vol. 18, pp. 59–64, 1996.

[10] K. S. Espenschied, R. D. Quinn, H. J. Chiel, and R. D. Beer, "Leg Coordination Mechanisms in the Stick Insect Applied to Hexapod Robot Locomotion," *Adaptive Behavior*, no. 4, pp. 455–468, 1993.

[11] D. Floreano and F. Mondada, "Evolutionary Neurocontrollers for Autonomous Mobile Robots," *Neural Networks*, vol. 11, pp. 1461–1478, 1998.

[12] R. Full, *Biomotion: Mystery of Motion. Why Do Animals Change Gait?* [online], 2000. http://polypedal.berkeley.edu/ib32/Lectures/gaitfolder/index.htm [cited 3 January 2006].

[13] J. Furushu, S. Akihito, S. Masamichi, and K. Eichi, "Realization of a Bounce Gait in a Quadruped Robot with Articular-Joint-Type Legs," *Proceedings of the 1995 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 697–702, Nagoya, Japan, 1995.

[14] G. Gabrielli and T. H. von Karman, "What Price Speed?," *Mechanical Engineering*, vol. 72, no. 10, pp. 775–781, 1950.

[15] P. P. Gambaryan, *How Mammals Run: Anatomical Adaptations*, JohnWiley & Sons, New York, 1974.

[16] C. Georgiades, A. German, A. Hogue, H. Liu, C. Prahacs, A. Ripsman, R. Sim, L. A. Torres, P. Zhang, M. Buehler, G. Dudek, M. Jenkin, and E. Milios, "AQUA: An Aquatic Walking Robot," *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3525–3531, Sendai, Japan, 2004.

[17] N. C. Heglund and C. R. Taylor, "Speed, Stride Frequency, and Energy Cost per Stride: How Do They Change with Body Size and Gait?," *Journal of Experimental Biology*, vol. 138, pp. 301–318, 1988.

[18] M. Hildebrand, "The Adaptive Significance of Tetrapod Gait Selection," *American Zoologist*, vol. 20, pp. 97-103, 1977.

[19] M. Hildebrand, "Analysis of asymmetrical gaits," *Journal of Mammalogy*, vol. 58, pp. 131-156, 1977.

[20] M. Hildebrand, "Analysis of Tetrapod Gaits: General Considerations and Symmetrical Gaits," In *Neural Control of Locomotion*, pp. 203-236, R. M. Herman, S. Grillner, P. S. G. Stein, and D. G. Stuart, editors, Plenum Press, New York, 1976.

[21] J. Hilljegerdes, D. Spenneberg, and F. Kirchner, "The Construction of the Four Legged Prototype Robot ARAMIES," *Proceedings of the 8th International Conference on Climbing and Walking Robots (CLAWAR)*, pp. 335-342, London, U.K., 2005.

[22] Hirose and Yoneda Robotics Lab, Tokyo Institute of Technology, URL: http://mozu.mes.titech.ac.jp/ [cited 23 March 2006].

[23] R. Hodoshima, T. Doi, Y. Fukuda, S. Hirose, T. Okamoto, and J. Mori, "Development of TITAN XI: A Quadruped Walking Robot to Work on Slopes. Design of System and Mechanism," *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 792-797, Sendai, Japan, 2004.

[24] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata, "Autonomous Evolution of Gaits with the Sony Quadruped Robot," *Proceedings of the Genetic and Evolutionary Computation Conference*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 2, pp. 1297–1304, Orlando, Florida, USA: Morgan Kaufmann, 1999.

[25] R. Huber, "Neuroethology: Flexibility and Adaptability in Neural Systems" [online], URL: http://caspar.bgsu.edu/~courses/NeuroEthology/Lectures/zlecture30/Lect30_NeuroEtho.htm [cited 5 May 2006].

[26] J. R. Hutchinson, D. Famini, R. Lair, and R. Kram, "Biomechanics: Are Fast-Moving Elephants Really Running?," *Nature,* vol. 422, pp. 493-494, April 2003.

[27] M. Jamshidi, L. dos Santos Coelho, R. A. Krohling, and P. J. Fleming, *Robust Control Systems with Genetic Algorithms*, Boca Raton, FL, CRC Press, 2003.

[28] M. S. Kim and W. Uther, "Automatic Gait Optimization for Quadruped Robots," *Australasian Conference on Robotics and Automation*, Brisbane, December 2003.

[29] H. Kimura, S. Akiyama, and K. Sakurama, "Realization of Dynamic Walking and Running of the Quadruped Using Neural Oscillator," *Autonomous Robots*, vol. 7, pp. 247–258, 1999.

[30] N. Kohl and P. Stone, "Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion," November 2003.

[31] D. P. Krasny, "An Analysis of High-Speed Running Gaits in a Quadruped Robot Using an Evolutionary Optimization Strategy," Master's thesis, The Ohio State University, Columbus, Ohio, 2002.

[32] D. P. Krasny, "Evolving Dynamic Maneuvers in a Quadruped Robot," Ph.D. dissertation, The Ohio State University, Columbus, Ohio, 2005.

[33] D. P. Krasny and D. E. Orin, "Achieving Periodic Leg Trajectories to Evolve a Quadruped Gallop," *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3842-3848, Taipei, Taiwan, September 2003.

[34] D. P. Krasny and D. E. Orin, "Generating High-Speed Dynamic Running Gaits in a Quadruped Robot Using an Evolutionary Search," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 4, pp. 1685-1696, August 2004.

[35] E. Krotkov and R. Simmons, "Performance of a Six-Legged Planetary Rover: Power, Positioning, and Autonomous Walking," *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pp. 169-174, Nice, France, 1992.

[36] W. A. Lewinger, M. S. Branicky, and R. D. Quinn, "Insect-Inspired, Actively Compliant Hexapod Capable of Object Manipulation," *Proceedings of the 8th International Conference on Climbing and Walking Robots (CLAWAR)*, pp. 65-72, London, U.K., 2005.

[37] D. W. Marhefka, "Fuzzy Control and Dynamic Simulation of a Quadruped Galloping Machine," Ph.D. dissertation, The Ohio State University, Columbus, Ohio, 2000.

[38] D. W. Marhefka, D. E. Orin, J. P. Schmiedeler, and K. J. Waldron, "Intelligent Control of Quadruped Gallops," *IEEE/ASME Transactions on Mechatronics*, vol. 8, no. 4, pp. 446-456, December 2003.

[39] R. B. McGhee, "Finite State Control of Quadruped Locomotion," *Proceedings of the Second International Symposium on External Control of Human Extremities*, Dubrovnik, Yugoslavia, 1966.

[40] R. B. McGhee, D. E. Orin, D. R. Pugh, and M. R. Patterson, "A Hierarchically Structured System for Computer Control of a Hexapod Walking Machine," *Theory and Practice of Robots and Manipulators, Proceedings of RoManSy-84 Symposium*, A. Morecki, Ed., pp. 375-381, London: Hermes Publishing Co., 1985.

[41] T. A. McMahon, "The Role of Compliance in Mammalian Running Gaits," *Journal of Experimental Biology*, vol. 115, pp. 263–282, 1985.

[42] A. E. Minetti, L. P. Ardigò, E. Reinach, and F. Saibene, "The Relationship Between Mechanical Work and Energy Expenditure of Locomotion in Horses," *The Journal of Experimental Biology*, vol. 202, pp. 2329–2338, 1999.

[43] R. S. Mosher, "Test and Evaluation of a Versatile Walking Truck," *Proceedings of the Off-Road Mobility Research Symposium, International Society for Terrain Vehicle Systems*, pp. 359–379, Washington, D.C., 1968.

[44] P. Nanua and K. J. Waldron, "Instability and Chaos in Quadruped Gallop," *Journal of Mechanical Design*, vol. 116, pp. 1096-1101, 1994.

[45] G. M. Nelson, R. D. Quinn, R. J. Bachman, and W. C. Flannigan, "Design and Simulation of a Cockroach-like Hexapod Robot," *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pp. 1106–1111, Albuquerque, NM, April 1997.

[46] N. Nicholson, *Gaits in General for Dressage: Math & Variations on a Theme of Walk, Trot, Canter* [online], URL: http://nicholnl.wcp.muohio.edu/DingosBreakfastClub/BioMech/BioMechGaits.html [cited 17 April 2006].

[47] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, MIT Press, 2000.

[48] K. Nonami, "Development of Mine Detection Robot COMET-III," *JSME International Journal*, Series C, vol. 46, no. 3, pp. 881–890, 2003.

[49] D. Papadopoulos and M. Buehler, "Stable Running in a Quadruped Robot with Compliant Legs," *Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 444–449, San Francisco, CA, 2000.

[50] S. Peng, C. P. Lam, and G. R. Cole, "A Biologically Inspired Four Legged Walking Robot," *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2024–2030, Taipei, Taiwan, 2003.

[51] I. Poulakakis, J. A. Smith, and M. Buehler, "On the Dynamics of Bounding and Extensions Towards the Half-Bound and the Gallop Gaits," *Proceedings of the 2nd International Symposium on Adaptive Motion of Animals and Machines*, March 2003.

[52] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992.

[53] D. Pugh, E. Ribble, V. Vohnout, T. Bihari, T. Walliser, M. Patterson, and K. Waldron, "Technical Description of the Adaptive Suspension Vehicle," *The International Journal of Robotics Research*, vol. 9, pp. 24–42, April 1990.

[54] M. H. Raibert, "Trotting, Pacing, and Bounding by a Quadruped Robot," *Journal of Biomechanics*, vol. 23, pp. 79–98, 1990.

[55] J. P. Schmiedeler, "The Mechanics of and Robotic Design for Quadrupedal Galloping," Ph.D. dissertation, The Ohio State University, Columbus, Ohio, 2001.

[56] J. Schofield, "Dante Survives the Inferno," *Design News*, pp. 68–74, September 1994.

[57] A. Seyfarth, H. Geyer, M. Guenther, and R. Blickhan, "A Movement Criterion for Running," *Journal of Biomechanics*, vol. 35, pp. 649-655, 2002.

[58] J. A. Smith and I. Poulakakis, "Rotary Gallop in the Untethered Quadrupedal Robot Scout II," *Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2004*, pp. 2556–2561, Sendai, Japan, 2004.

[59] D. Spenneberg, K. McCullough, and F. Kirchner, "Stability of Walking in a Multilegged Robot Suffering Leg Loss," *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2159-2164, 2004.

[60] S. M. Song and K. J. Waldron, *Machines That Walk: The Adaptive Suspension Vehicle*, Cambridge, MA: MIT Press, 1989.

[61] Sony Corporation, "OPEN-R SDK: Model Information for ERS-7" [online], 2003, URL: http://www.ai.rug.nl/vakinformatie/pas/download/AIBO/ModelInformation_7_E.pdf [cited 8 May 2006].

[62] L. Steels, "Emergent Functionality in Robotics Agents Through On-line Evolution," *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 8–14, R. A. Brooks and P. Maes, Eds., 1994.

[63] V. B. Sukhanov, *General System of Symmetrical Locomotion of Terrestrial Vertebrates and Some Features of Movement of Lower Tetrapods*, Amerind Publishing, New Delhi, 1974.

[64] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," *Advances in Neural Information Processing Systems*, vol. 12, pp. 1057–1063, The MIT Press, 2000.

[65] J. Urzelai and D. Floreano, "Evolutionary Robots with Fast Adaptive Behavior in New Environments." *Proceedings of the Third International Conference on Evolvable Systems (ICES)*, pp. 241–251, Edinburgh, Scotland, UK, 2000.

[66] L. Ziegler, *Identifying the Symmetrical Gaits of Horses* [online], 2001, URL: http://www.leeziegler.com/sgaits1.html [cited 12 May 2006].