INTELLIGENT CONTROLS FOR A SEMI-ACTIVE HYDRAULIC PROSTHETIC KNEE

TIMOTHY WILMOT

Bachelor of Science in Electrical Engineering

Cleveland State University

May, 2010

Bachelor of Arts in Physics

Cleveland State University

May, 2010

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE IN ELECTRICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

August, 2011

This thesis has been approved for the

Department of **ELECTRICAL AND COMPUTER ENGINEERING**

and the College of Graduate Studies by

_____

Thesis Committee Chairperson, Dr. Dan Simon

_____

Department/Date

_____

Dr. Lili Dong

_____

Department/Date

_____

Dr. Fuqin Xiong

_____

Department/Date

ACKNOWLEDGMENTS

This research was partially supported by the State of Ohio, Department of Development and Third Frontier Commission, which provided funding through the Cleveland Clinic Foundation in support of the project Rapid Rehabilitation and Return to Function for Amputee Soldiers. This work was also partially supported by the National Science Foundation under grant number 0826124. Brian Davis from the Austen BioInnovation Institute, William Smith and Sergey Samorezov from the Cleveland Clinic Foundation, and Ton van den Bogert from Orchard Kinetics, significantly contributed to the groundwork for this thesis.

I would like to thank my advisor, Dr. Dan Simon, for all of his help and guidance and the many hours he labored with me on this project. The other thesis committee members, Dr. Fuqing Xiong and Dr. Lili Dong, made several valuable suggestions that improved the quality of this thesis. I would also like to thank my fellow students Rick Rarick and Steve Szatmary for all their help and input. Rick Rarick developed much of the code used for finding the open loop control using biogeography based optimization. Steve Szatmary is responsible for the gradient descent development. Their work forms a large part of the foundation of this thesis. Finally I would like to thank my wife, Sarah, my parents, and the rest of my family for all their support and patience.

INTELLIGENT CONTROLS FOR A SEMI-ACTIVE HYDRAULIC PROSTHETIC KNEE

TIMOTHY WILMOT

ABSTRACT

We discuss open loop control development and simulation results for a semi-active above-knee prosthesis. The control signal consists of two hydraulic valve settings. These valves control a rotary actuator that provides torque to the prosthetic knee. We develop open loop control using biogeography-based optimization (BBO), which is a recently developed evolutionary algorithm, and gradient descent. We use gradient descent to show that the control generated by BBO is locally optimal. This research contributes to the field of evolutionary algorithms by demonstrating that BBO is successful at finding optimal solutions to complex, real-world, nonlinear, time varying control problems. The

research contributes to the field of prosthetics by showing that it is possible to find effective open loop control signals for a newly proposed semi-active hydraulic knee prosthesis. The control algorithm provides knee angle tracking with an RMS error of 7.9 degrees, and thigh angle tracking with an RMS error of 4.7 degrees. Robustness tests show that the BBO control solution is affected very little by disturbances added during the simulation. However, the open loop control is very sensitive to the initial conditions. So a closed loop control is needed to mitigate the effects of varying initial conditions. We implement a proportional, integral, derivative (PID) controller for the prosthesis and show that it is not a sufficient form of closed loop control. Instead, we implement artificial neural networks (ANNs) as the mechanism for closed loop control. We show that ANNs can greatly improve performance when noise and disturbance cause high tracking errors, thus reducing the risk of stumbles and falls. We also show that ANNs are able to improve average performance by as much as 8% over open loop control. We also discuss embedded system implementation with a microcontroller and associated hardware and software.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Problem Overview

We propose a new design for a hydraulic prosthetic knee for above knee amputees, and also derive open and closed loop control signals for the prosthesis. The prosthesis harvests energy and provides controlled release of energy during the gait cycle with a spring loaded high pressure hydraulic chamber, a low pressure hydraulic chamber, and a rotary actuator. We show that solving the control problem by analytical means is not feasible, but that biogeography-based optimization (BBO) can find near optimal open loop control solutions.

Artificial neural networks (ANNs) are used for closed loop control due to their good generalization properties and ability to learn desired behaviors. We train the ANNs with BBO to find the optimal network weights to determine how signals propagate through the network. This demonstrates the effectiveness of BBO for finding solutions to difficult optimization problems. The semi-active nature of the prosthesis allows the

device to use less power than its fully active prosthetic counterparts. We also discuss implementation of the control with a microcontroller and the associated hardware.

1.2 Literature Review

Prostheses have long been known to produce degenerative side effects such as arthritis and spinal pain [1] because of the unnatural and high torques that the user's hip produces when compensating for the prosthesis' inadequacy. Therefore, we place a high priority not only on the appearance of normal gait through tracking reference angles and coordinates, but also on the hip torques that the amputee has to produce to interface with the prosthesis.

Microprocessor controlled knees have succeeded in several different prostheses. Most notably, the Otto Bock C-Leg has become the benchmark of prosthetic knees. The performance of the C-Leg depends on the controls embedded in its microcontroller. Otto Bock's leg reacts well to a variety of situations and has proven to decrease detrimental side effects relative to more conventional prostheses [4]. Microprocessor control has proven to be the best option for high performance prostheses through a series of performance evaluation tests comparing the C-leg to non-microprocessor controlled prostheses [4]. However, even the most modern and technically sophisticated knee prostheses still do not fully restore normal gait and do not prevent all detrimental side effects [5].

Our open loop prosthetic control approach focuses on biogeography based optimization (BBO), which is a recently developed evolutionary algorithm (EA). BBO gives better performance than traditional EAs for a wide variety of benchmarks and real-

world optimization problems [10]. After BBO generates a control solution, we use gradient descent for fine tuning to further optimize the control, or to determine if the BBO solution is already locally optimal. Solving for an optimal open loop control by strictly analytical means is intractable for the complex, non-differentiable prosthetic control problem. We use BBO to search for an open loop control by evaluating and minimizing a cost function through testing candidate control solutions on simulations and/or hardware.

Our closed loop control approach focuses on using artificial neural networks (ANN) to add a delta control to the open loop control to compensate for disturbances and measurement noise. The ANN uses weighted connections between the artificial neurons to determine how signals will propagate through the network. We use BBO to train the network to take error data inputs and output delta controls that will compensate for disturbances and noise. As with open loop control, BBO evaluates and minimizes a cost function for candidate weight vectors in the ANN.

There are a number of reasons that researchers have found EAs to be attractive for complex, nonlinear, and time varying problems like prosthetic limb control. For example, genetic algorithms (GAs) might be appropriate tools for finding solutions to nonlinear, second order, two point boundary value problems [11] because GAs are simple and do not require any advanced mathematical tools. EAs can find nonlinear controls for generic trajectory optimization problems [12]. GAs and simulated annealing have found optimal trajectories for spacecraft trajectory optimization problems [13]. GA-based optimization for missile flight midcourse guidance is another example of their usefulness for control [14]. They key to all of these studies is the conversion of the control optimization

problem to a parameter optimization problem. This is done by parameterizing the control signals, and then using a parameter optimization algorithm, such as an EA, to find the parameters that result in the best controls. This method was used to optimize muscle excitation signals for large-scale musculoskeletal systems [15]. EAs are often effective tools for parameter optimization, so the conversion of control problems to parameter optimization problems makes them appropriate problems for EAs.

We convert the prosthetic open loop control problem into a parameter optimization problem by representing the control signals as Fourier series. This idea was first used for the optimization of structural systems [16] with linear dynamics and a quadratic performance index. That reference assumed that the optimal profile of each configuration variable was continuous on the interval [0, T], where T is the terminal time, with a convergent Fourier series. In practice only a finite number of Fourier terms are used to represent the control signals and this idea converts the control optimization problem to a parameter optimization problem. This approach is a computationally efficient approach for optimal control, and is able to handle boundary conditions and high order problems. The GA / Fourier series approach to optimal control was also applied to robotic manipulator control [17].

This previous research motivates us to use the Fourier series approach for the prosthetic control problem. The recent success of BBO further motivates us to use BBO for the optimization of the Fourier series coefficients that represent the control signals.

We can also formulate prosthetic closed loop control as a parameter optimization problem. The individual weights in the ANNs are parameters that we can optimize with BBO. Artificial neural networks are generally used for pattern recognition and to model

complex, nonlinear relationships between inputs and outputs. Their structure is inspired by biological neurons and they have been used in bioengineering applications like prosthetic control. The network contains weighted connections between the artificial neurons that determine how the inputs propagate through the network. These networks can learn behaviors by adjusting these weights. If we train the networks with BBO they do not need to have any information about the dynamic system in order to learn. BBO can find the optimal weight vectors for the ANN to maximize performance for a given number of training data sets. They can also generalize and are able to handle a large number of input cases not in the training data and still make correct decisions. These factors make them ideal for prosthetic control.

ANNs are widely used for control by processing myoelectric signals as inputs. Cerebellar Model Arithmetic Computer (CMAC) is an effort for neural network control of cybernetic limb prostheses [18]. It is a perceptron like ANN that mimics the cerebellum in complex motor tasks. CMAC is used for control of the elbow prosthesis and takes the myoelectric signals and feedback to form the input vector to the network. What is particularly interesting about CMAC is why it is used: “Because of the dynamics of prosthetic limb control, a Perceptron-like network which ‘… generalizes only over a small neighborhood of input-space [sensory input vectors], and which has good dichotomizing properties for [trajectory] points well separated …’ [19] would serve as a suitable limb controller.” ANNs have also been used to control a hand prosthetic with myoelectric inputs [20]. The ANN is trained to mimic the behavior of the prosthetic hand (by comparing the hand model to the prosthetic hand movements). Then the hand model

motions are compared to the biological hand and the errors are propagated back to the controller ANN and used as the target signals for the controller training.

No prosthesis exists that can truly provide “natural walking,” and even the acclaimed C-leg is unable to accomplish this task [21]. Researchers believe that control of a powered above knee prosthetic using ANNs might be able to achieve more natural walking [21]. This approach uses a feedback error learning network that is used to identify inverse dynamics of the prosthetic joint movements with simple sinusoidal inputs. The inverse of the system can then be used to determine what inputs should be used given certain outputs. ANNs are a good choice of control technique for systems like the prosthetic knee where dynamics are very hard to solve.

1.3 Thesis Contribution

This thesis contributes to the current research through the novel use of the methods described in the preceding section on this particular prosthesis. The prosthesis incorporates an original hydraulic valve and energy management scheme. BBO has never been applied to any prosthetic control problem. We further enhance this application of BBO with gradient descent analysis. ANNs have been used for prosthetic control, however they have never provided control in the way they do in this thesis. The evaluation of the PID controller has not been done before on this prosthesis. We also provide a hardware and software layout for the prosthesis in this thesis.

1.4 Thesis Organization

This thesis is organized as follows. Chapter 2 discusses the prosthetic dynamics, the prosthetic control problem formulation, and prosthetic system modeling in MATLAB®. Chapter 3 discusses the open loop control problem formulation, its solution using BBO, and simulation results. The results show that BBO can achieve knee angle tracking with an RMS error of 7.9 degrees, and thigh angle tracking with an RMS error of 4.7 degrees. Furthermore, analysis of the gradient descent Hamiltonian shows that the BBO control solution is near optimal. Chapter 4 discusses the PID closed loop controller implementation and shows the results. Chapter 5 discusses the closed loop control problem formulation, its solution using ANNs, and simulation results. Chapter 6 discusses microcontroller implementation. Chapter 7 contains conclusions and suggestions for future work.

CHAPTER 2

PROBLEM FORMULATION

2.1 System Modeling

We begin the problem formulation for prosthetic knee control with the derivation of the governing dynamic equations. There are two distinct phases of the human gait cycle that are represented by two different dynamic system models. The stance phase of a leg is defined as the period of time when the foot is in contact with the ground. It begins when the heel first makes contact with the ground, and ends when the foot lifts up off the ground. Swing phase follows stance phase, and is defined as the period of time when the foot is not in contact with the ground. During stance phase we can represent the dynamic system that describes the behavior of the leg as an inverted double pendulum. During swing phase we can represent the dynamic system that describes the behavior of the leg as a regular double pendulum. Figure 1 shows the stance and swing phase of the human gait during one stride. The stance phase comprises about the first 60% of the stride, and the swing phase comprises about the last 40%.

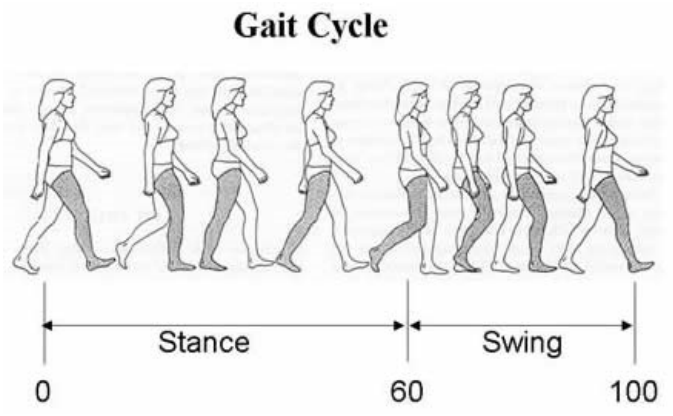

Figure 1: The human gait cycle.
The stance phase of the shaded leg begins when the heel first makes contact with the ground, and ends when the foot leaves the ground. The swing phase of the shaded leg begins when the foot leaves the ground, and ends when the heel first strikes the ground. This figure is adopted from [22].

We derived dynamic equations for the two systems in Figure 1 using AutoLev™ software [23]. The equations are unwieldy and so we list them in the appendix. The equations are derived in [24]. Figure 2 shows the diagram of the limb along with the definition of the angles and forces.

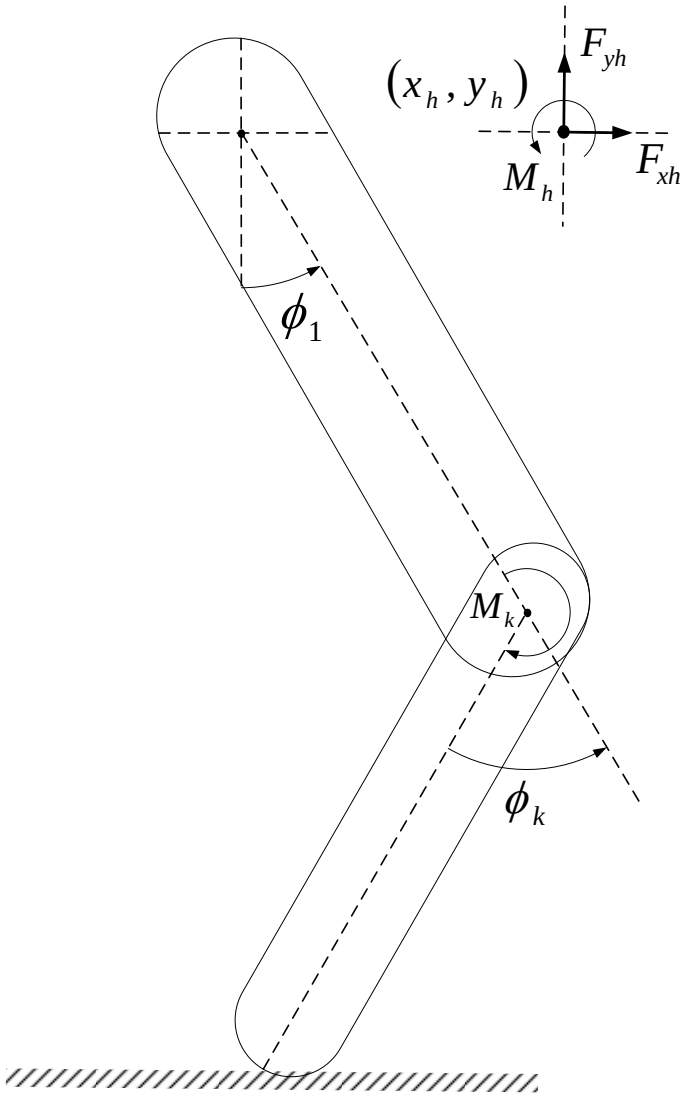

Figure 2: The prosthetic limb diagram.
Angles are positive in the counter clockwise direction and are negative as shown here.

The general form of the dynamic equations is given as:

$$\ddot{\phi}_1 = f_1\left(M_k, M_h, F_{yh}, F_{xh}, \phi_1, \phi_k, \dot{\phi}_1, \dot{\phi}_k\right) \quad \text{stance phase hip angle dynamics} \quad (1)$$

$$\ddot{\phi}_k = f_2\left(M_k, M_h, F_{yh}, F_{xh}, \phi_1, \phi_k, \dot{\phi}_1, \dot{\phi}_k\right) \quad \text{stance phase knee angle dynamics} \quad (2)$$

$$\ddot{\phi}_k = f_3\left(M_k, \ddot{y_h}, \ddot{x_h}, \phi_1, \phi_k, \ddot{\phi}_1\right) \quad \text{swing phase knee angle dynamics} \quad (3)$$

Note that during swing phase, the thigh angle ϕ_1 is entirely controlled by the user, and so we do not have a dynamic equation for $\ddot{\phi}_1$ during swing phase. M_h, F_{xh} and F_{yh} are moments and forces applied at the hip by the user. M_k is the moment applied by the prosthesis at the knee. Table I shows the definitions of the variables in Equations 1–3.

Table I: Dynamic equation variables.

ϕ_1	Thigh angle	F_{yh}	Hip force *y*-coordinate
ϕ_k	Knee angle	F_{xh}	Hip force *x*-coordinate
M_h	Hip moment (torque)	y_h	Hip position *y*-coordinate
M_k	Knee moment (torque)	x_h	Hip position *x*-coordinate

Next we discuss the modeling of the rotary hydraulic actuator. The actuator provides a mechanism for controlled storage and release of energy during the gait cycle. This storage and release enables the hydraulic actuator to deliver torque and damping to the knee without external power; the only power required by the knee is for opening and closing hydraulic valves. This significantly reduces the amount of power needed for operation when compared to a fully active, powered knee. Figure 3 shows the hydraulic rotary actuator. The high pressure accumulator (HPA) is loaded with a spring. The low pressure accumulator (LPA) is loaded with a bladder to maintain constant pressure. The solid shaded part of the rotary device is the stator, and the dashed-filled part is the rotor. The four dot-filled parts are the hydraulic chambers. Control is provided by two valves that enable fluid flow into and out of the high and low pressure accumulators, and u_1 and

u_2 are the control signals. The high pressure accumulator is spring loaded and this spring provides the energy storage and release capabilities.

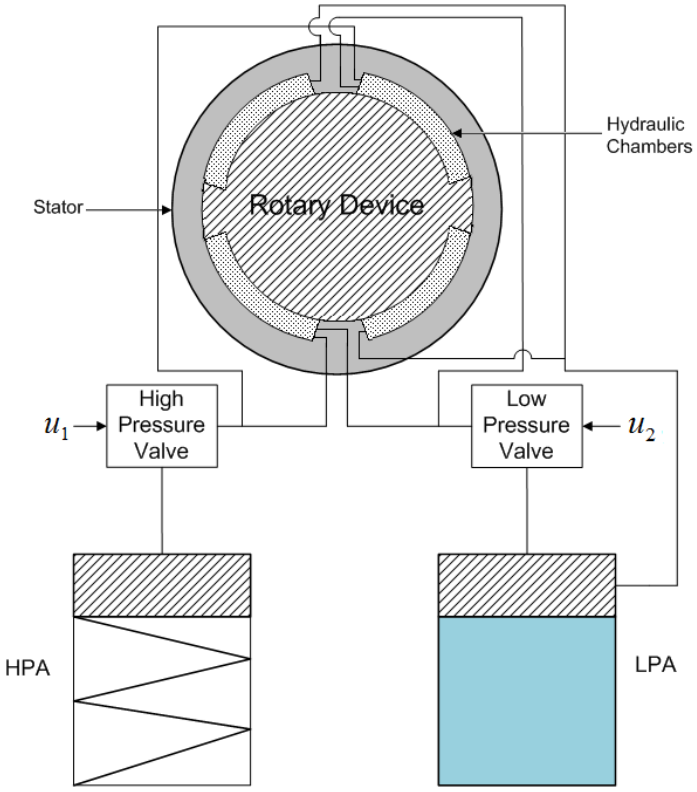

Figure 3: Rotary hydraulic actuator.

Table II shows the rotary actuator parameter definitions. The equations that describe the motion of the rotary actuator are derived in [24] and given as:

$$u_1^2 C_1^2 (ks - M_k G - B_1 v_1) - v_1 |v_1| = 0 \tag{4}$$

$$u_2^2 C_2^2 (P_0 - M_k G - B_2 v_2) - v_2 |v_2| = 0 \tag{5}$$

$$\dot{\phi}_k - G(v_1 + v_2) = 0 \tag{6}$$

$$\dot{s} + v_1 = 0 \tag{7}$$

Table II: Rotary equation parameters.
The valve controls are normalized between 0 (fully closed) and 1 (fully open).

B_1	Constant viscous drag through valve 1
B_2	Constant viscous drag through valve 2
C_1	Maximum cross-sectional area of valve 1
C_2	Maximum cross-sectional area of valve 2
G	Moment-pressure proportional constant
k	High pressure accumulator spring elasticity
P_0	Pressure in the low pressure accumulator
s	High pressure fluid volume
u_1	Valve 1 control normalized to [0, 1]
u_2	Valve 2 control normalized to [0, 1]
v_1	Upward fluid flow through valve 1
v_2	Upward fluid flow through valve 2

2.2 Control Concepts

We collected reference data from an able-bodied human subject in the Cleveland Clinic Foundation gait lab. Cameras in the lab track thigh and knee angles, and a force plate collects ground contact data while the subject walks at a normal but slow pace. The test subject has a mass of 78 kilograms and a height of 1.83 meters. Gait lab software also calculates the hip and knee torques that the able-bodied human generates during his slow walk. We use the able-bodied knee and thigh angles as reference trajectories for our prosthetic controller. We are particularly interested in the able-bodied hip torque. We want an amputee to walk with hip torque that is close to the reference trajectory to minimize the negative degenerative side effects due to long-term use of the prosthesis.

To control the prosthesis, we first look for an open loop control without considering any disturbances, uncertainties, or noise. Later we can use this open loop control in conjunction with feedback control to provide a more robust control solution. Figure 4 and Figure 5 show the control block diagrams for open loop control and Figure 6 and Figure 7 show the control block diagrams for closed loop control. During stance phase we want to track the reference knee angle. Operation of the prosthesis during stance phase exerts forces and torques on the hip. The user needs to compensate for the forces and torques that the prosthesis exerts on the hip by using his/her existing hip muscles and conscious or unconscious effort. We model the user's compensation with a simple proportional feedback controller that adds hip torque to the reference hip torque based on the thigh angle tracking error that occurs during prosthetic operation. The equation is shown as:

$$M_h = M_{h\,ref} + P_f\left(\phi_{1\,ref} - \phi_1\right), \qquad \text{where } P_f = 100 \tag{8}$$

We obtained P_f by manually tuning the parameter to provide good thigh angle tracking while maintaining hip torques that were still reasonably close to reference values.

During swing phase we assume that the thigh angle and its derivatives are supplied by the user, and are exactly equal to the reference values. We make this assumption because during swing phase the operation of the prosthesis will have little impact on the hip torque. Because of this assumption, during swing phase we only want to track the knee angle. Note that the parameters in Figure 4 through Figure 7 are defined in Table I, the limb dynamics are shown in Equations 1–3, and the rotary dynamics are given in Equations 4–7.

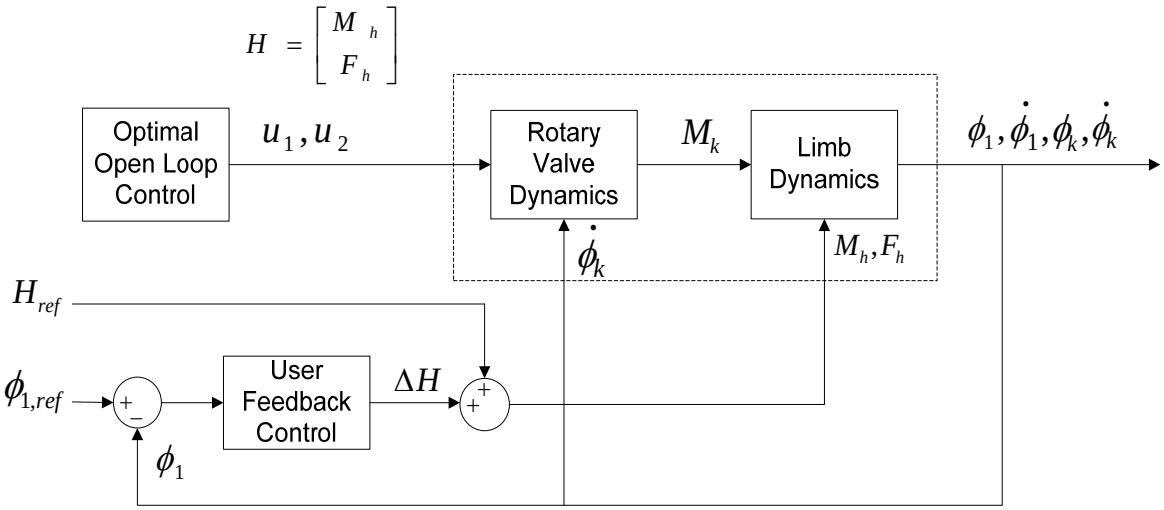

Figure 4: Optimal open-loop control during stance phase.

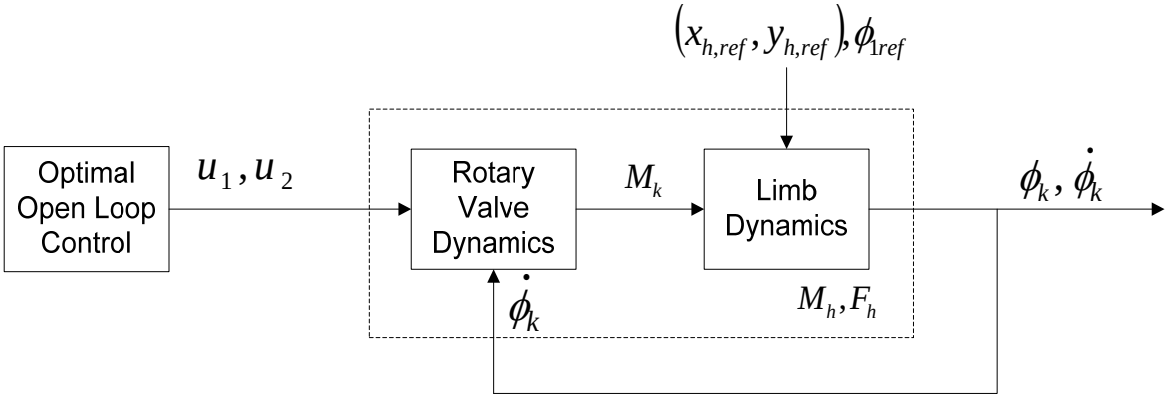

Figure 5: Optimal open-loop control during swing phase.

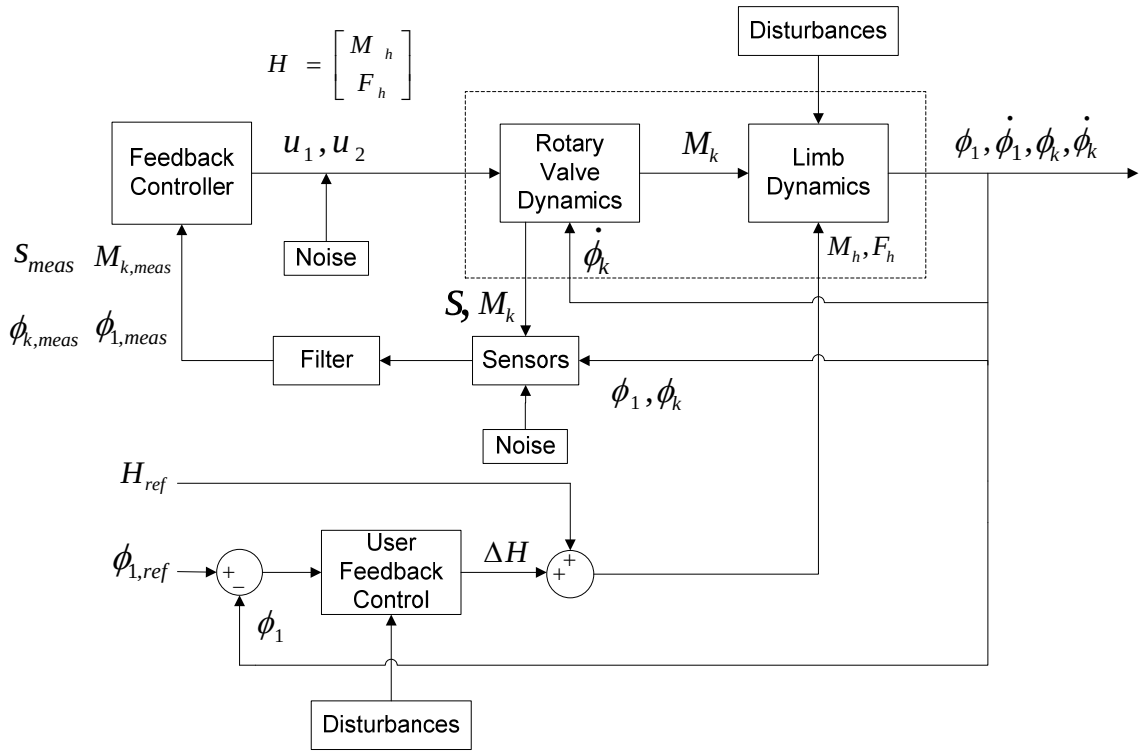

Figure 6: Closed loop control during stance phase.

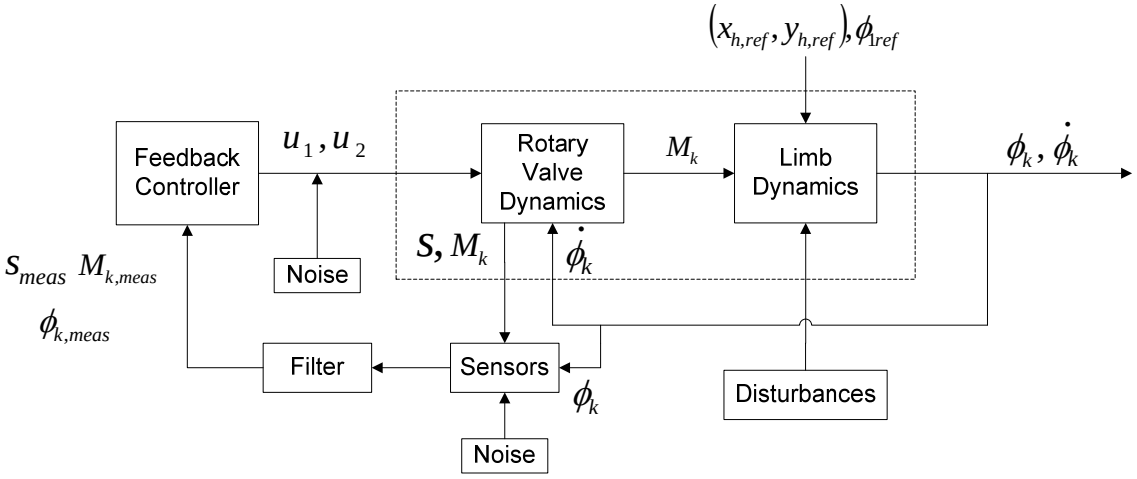

Figure 7: Closed loop control during swing phase.

CHAPTER 3

EVOLUTIONARY OPEN LOOP CONTROL

As a starting point for prosthetic control we first consider the open loop control that delivers the best tracking performance without any disturbances or uncertainties. We control the prosthesis in discrete time with a control frequency of 100 Hz. The open loop control consists of the sequence of signals, u_1 and u_2, to the two hydraulic flow valves. The control signals vary between 0 and 1, corresponding to fully closed and fully open respectively. We want to find the sequence of controls that will give the best overall performance.

Our search techniques rely on BBO because analytical solutions are intractable since the prosthetic system is complex and non-differentiable (Equations 4–7). Since we do not have a power source that provides torque to the knee other than the spring in the high pressure accumulator, we must store and release energy selectively so as not to deplete the stored energy or lose energy expenditure capability at points that might cause the prosthesis to collapse, cause the knee angle to exceed zero, or cause angle tracking to be poor.

We can easily treat the rotary equations for the prosthetic knee, Equations 4–7, as a black box for which we can numerically obtain input/output characteristics. Given s, $\dot{\phi}_k$, u_1 and u_2 we can solve Equations 4–7 for knee torque M_k. We developed a MATLAB program that will show the range of outputs (M_k) for a given s and $\dot{\phi}_k$. Then we develop a surface where the control signals u_1 and u_2 are on the interval 0 to 1 (0 is fully closed and 1 is fully open) and incremented by 0.1. The control signals cannot both be below 0.2 at the same time. Not allowing the control signals to be below a certain threshold at the same time is due to numerical issues in the rotary hydraulic equations. If we close both valves at the same time then the knee joint would lock resulting in an unknown torque. We generate a surface of torque based on these calculations to provide visual insight. Figure 8 shows several plots of torque for different s and $\dot{\phi}_k$ values.

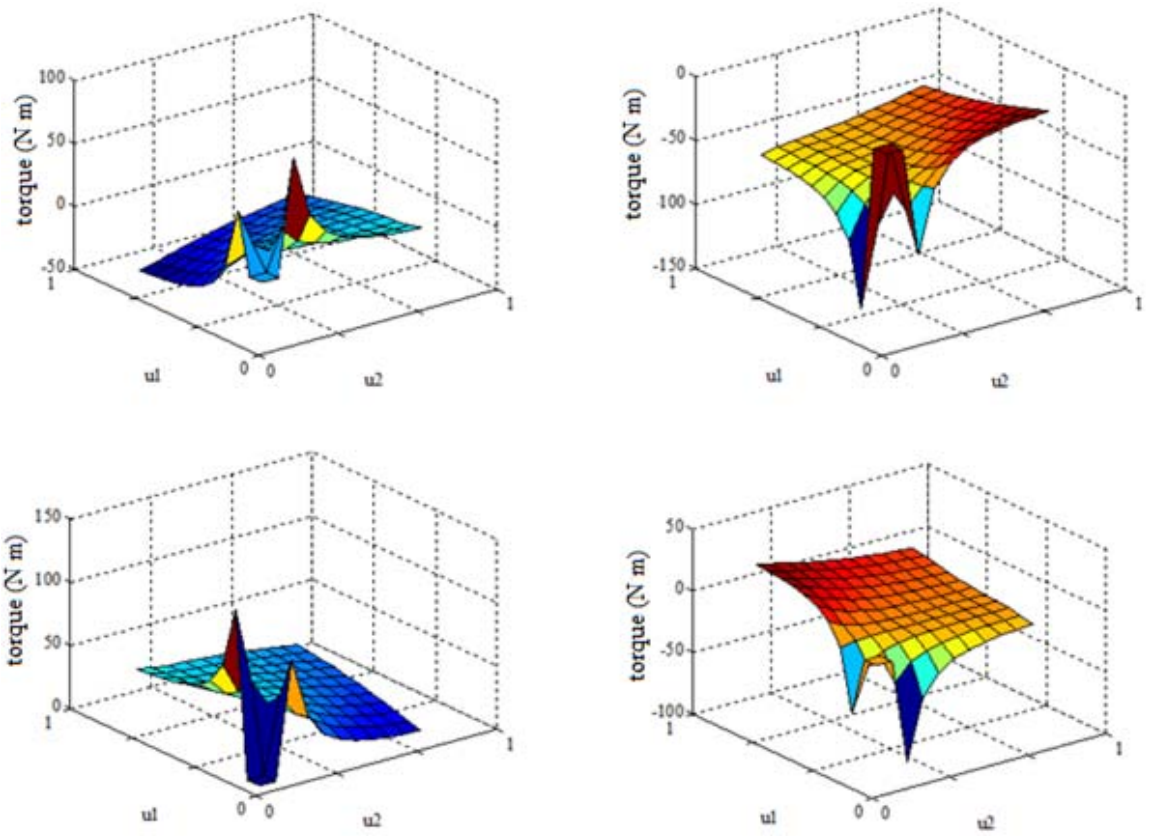

Figure 8: Plots of knee torque for different s and $\dot{\phi}_k$ values.
Velocity = –3 rad/sec and Volume = –2 cm^3 (top left).
Velocity = 3 rad/sec and Volume = –2 cm^3 (top right).
Velocity = –3 rad/sec and Volume = 2 cm^3 (bottom left).
Velocity = 3 rad/sec and Volume = 2 cm^3 (bottom right).

The relationship between u_1, u_2 and desired torque is a complex, nonlinear function of knee velocity and high pressure accumulator volume. We can see from Figure 8 that an inverse function does not exist. That is, given s, ϕ_1, and torque, there is not a unique solution for u_1 and u_2. Also, determining the torque that gives the best angle tracking at one instant might change the system states in such a way that stored energy is depleted later in the gait, which would result in a drastic deterioration of angle tracking.

Evolutionary algorithms often excel at this type of complex, multidimensional, nonlinear optimization problem. Therefore, we choose BBO, a recently developed EA, to optimize the prosthetic controls. Section 3.1 gives an overview of BBO and how it can be used to find optimal controls. Section 3.2 provides simulation results. Section 3.3 discusses optimization results using gradient descent, which theoretically confirms that the BBO controls are locally optimal. Section 3.4 presents robustness tests.

3.1 Biogeography-Based Optimization

BBO is an evolutionary algorithm that has solved optimization problems more effectively than many other evolutionary algorithms [25]. BBO has also solved real-world application problems such as ECG signal classification [25], power system optimization [26], groundwater detection [27], and satellite image classification [28]. BBO is based on the science and study of species migration from one habitat to another. Habitats have different levels of suitability for different species. This is called the habitat suitability index (HSI) of a particular habitat. Habitats with a high HSI tend to have a large number of species, and habitats with a low HSI tend to have a low number of species. Species will immigrate to, and emigrate from a habitat based on how many species are already there. A habitat with a large number of species (high HSI) will tend to have a low immigration rate and a high emigration rate. Conversely, a habitat with a low number of species (low HSI) will tend to have a high immigration rate and low emigration rate.

Figure 9 shows sample migration curves for BBO. Nature will optimize the number of species living in each habitat to achieve equilibrium. Figure 9 shows two candidate solutions to the same problem, S_1 and S_2. S_1 is a relatively bad solution and S_2 is a relatively good solution. Bad candidate solutions are likely to receive features from other candidate solutions, and unlikely to share features with other candidate solutions. Good candidate solutions are likely to share features with other candidate solutions, and unlikely to receive features from other candidate solutions.

Now picture each habitat as a candidate solution to an optimization problem, and picture each species as a distinguishing feature (independent variable) of that candidate solution. Following BBO, each candidate solution shares its features with the surrounding candidate solutions. As migration proceeds for many generations, the habitats become more suitable for their species, which corresponds to candidate solutions providing better solutions to an optimization problem. Also implemented in BBO are common evolutionary algorithm concepts like elitism and mutation, which we discuss in more detail later in this section. Figure 10 shows the outline of the BBO algorithm.

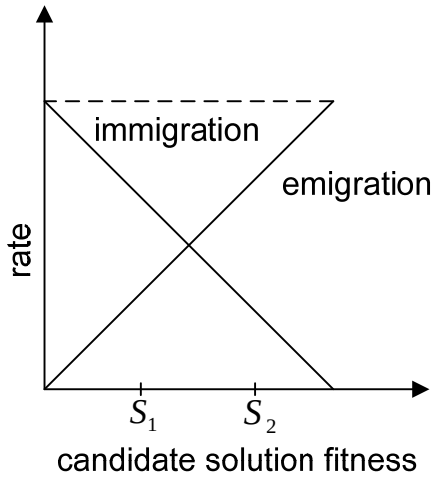

Figure 9: BBO migration curves.

For each candidate solution $y_k, k \in \{1, \dots, N\}$, define emigration probability $\mu_k \propto$ fitness of $y_k, \mu_k \in [0,1]$
For each candidate solution $y_k$define immigration probability $\lambda_k = 1 - \mu_k$
$z \leftarrow y$
For each candidate solution z_k
    For each solution feature s
        Use λ_k to probabilistically decide whether to immigrate to z_k
        If immigrating then
            Use $\{\mu_i\}$ to probabilistically select the parent solution y_j
            $z_k(s) \leftarrow y_j(s)$
        End if
    Next solution feature
    Probabilistically mutate z_k
Next candidate solution
$y \leftarrow z$

Figure 10: Outline of one BBO generation.
***N* is the population size (number of candidate solutions each generation). *z* is a temporary population of solutions so that migration can be completed before the population is replaced for the next generation.**

In order to use BBO to solve the prosthetic knee control problem, we need to decide two things. First, we need to decide what to use as features of a candidate control solution. Second, we need to decide what cost function to use. Our candidate control solutions consist of the two valve control signals for the entire period of the gait cycle.

Assuming a T = 1.26 second gait period, as obtained in our gait lab from able-bodied gait data, and assuming a 100 Hz control signal, this requires 126 values for each control signal. In order to reduce the size of the search space, we represent each control signal as a Fourier series. The Fourier series can point-wisely approximate any periodic, integrable function to any degree of accuracy away from any points of discontinuity [29]. The formula for one of the control signals with a like formula for the second control signal is shown as:

$$u_1(t) = \frac{a_0}{2} + \sum_{n=1}^{12} [a_n \cos(2\pi t/T) + b_n \sin(2\pi n t/T)] \quad (9)$$

The control signals saturate at 0 (fully closed) and 1 (fully open). We have compared the Fourier series form of the control to other forms, such as piecewise linear, piecewise constant, and cubic splines. Our studies show that the Fourier series representation performs the best. As seen in Equation 9, we use 25 coefficients in the Fourier series of each control. Our experiments show that this number of coefficients provides enough resolution to thoroughly search the space of control signals, while not unduly increasing the size of the search space. We limit the search space to $0 \leq a_0 \leq 2$, and $-1 \leq a_n \leq 1, -1 \leq b_n \leq 1$ for $n > 0$. We know that the control signal must be between 0 and 1 and we want to limit the search space so that a good control can be found in a reasonable amount of time. We found these ranges of values to be an appropriate balance through trial and error.

Every 0.01 seconds we evaluate the Fourier series for each control and use those values as the control for the next 0.01 seconds. This simulates the operation of the zero-order hold microcontroller, which updates the control signals at 100 Hz.

To improve control performance, we also optimize the initial conditions of the gait phase. These conditions include the initial knee angle velocity and the initial thigh angle velocity. We allow these values to vary during the search between plus or minus five times their reference values. We also optimize two physical parameters of the rotary actuator: the spring constant and the initial volume of the high pressure cylinder. We allow the spring constant to vary during the search between zero and two times its nominal design value and we allow the initial fluid volume to vary between −10 and +10 cubic centimeters. Table III summarizes the independent variables that are optimized by BBO. Note that these are the candidate solution features that we discussed at the beginning of this section. There are thus a total of 56 independent variables in each candidate control solution.

Table III: Independent variables optimized by BBO for open-loop control.

25 Fourier coefficients for control signal u_1
25 Fourier coefficients for control signal u_2
Initial thigh angle (stance phase) $\phi_1(0)$
Initial thigh angle derivative (stance phase) $\dot{\phi}_1(0)$
Initial knee angle (stance phase) $\phi_k(0)$
Initial knee angle derivative $\dot{\phi_k}(0)$
Initial high pressure accumulator volume $s(0)$
Spring constant k

Next we assign a cost value to each candidate solution. Cost and fitness are generally related to each other inversely. That is, as a candidate solution improves, its cost decreases while its fitness increases. Our cost function includes angle tracking errors, the amount by which the knee angle exceeds zero, and the high pressure volume difference between the beginning and end of the gait. Note that the cost function adds dimensionally incompatible quantities, scaling is implied [30] and units are not applicable. We want the high pressure volume to be periodic for effective operation over multiple gait cycles. The cost function is given as:

$$J = w_1(s(0) - s(T))^2 + \int_{t=0}^{T} \left[w_2\left(\phi_1(t) - \phi_{1ref}(t)\right)^2 + w_3\left(\phi_k(t) - \phi_{kref}(t)\right)^2 + w_4 U\left(\phi_k(t)\right)\phi_k(t) \right] dt \qquad (10)$$

Note that $U(\cdot)$ is the unit step function and T is the time at the end of the gait cycle. We choose the unit step function because it is equal to one when its argument is greater than zero and we want to penalize all knee values that are greater than zero. We choose the weights (w_i) by trial and error. We run a series of BBO optimization programs with different weight values, and visually inspect the resulting outputs to judge the tracking performance, the negativity constraint on the knee angle, and the periodicity of the HPA volume. We find that $w_1 = 0.1,\ w_2 = 1,\ w_3 = 1,$ and $w_4 = 0.1$ are appropriate values for the weights. We do not assess a cost to the hip torque tracking error. However, we assign a sufficiently small proportional gain (P = 100) to the feedback controller to keep the hip torque reasonably close to the reference torque.

Table IV shows the parameters that we used for BBO. The BBO software allows us to seed the initial population with the best solutions (prosthetic controls, initial conditions and high pressure accumulator spring constant) from previous BBO runs, and here the initial population was seeded from six previous runs. Thus, the number of

generations shown in Table IV does not by itself represent a measure of the speed of convergence.

Table IV: BBO parameters for open-loop control.

Mutation rate	0.01
Number of elites	2
Population size	100
Number of generations	100 per run

Mutation is a process that probabilistically mutates features of a candidate solution to increase diversity in the population [10]. As we show in Table IV, in every generation, each candidate solution feature has a 1% probability of mutation. If the solution feature is selected for mutation, then it is replaced with a random number uniformly distributed between the minimum and maximum of its search domain.

Elitism involves saving some of the best solutions of the current generation to insert into the population of the next generation. This ensures that BBO will never lose the best solutions from one generation to the next. The lowest cost value reported at each generation will never increase from one generation to the next. As we show in Table IV, we saved the two best candidate solutions at each generation for insertion into the next generation's population.

We choose our population size and number of generations based on computation time and the effect of diminishing returns. Experience shows that for the prosthetic control optimization problem, a BBO run of 100 generations with 100 individuals can

find a good solution while not wasting valuable computation time on unneeded generations or an unnecessarily large population.

3.2 Open Loop Control Results

Figure 11 shows the best cost at every generation of the BBO algorithm, with Figure 12 showing the last 100 generations. We reinitialized the population at certain intervals to widen the search space and avoid becoming trapped in a local minimum. When we reinitialize the population we also keep some of the best results from the previous population to avoid losing a good solution. Figure 13 shows the average cost at each generation. Figure 14 shows the best Fourier coefficients of the control signal. Table V shows the best initial conditions and *k* that BBO found. Figure 15 shows the best knee angle and thigh angle tracking that BBO achieved and Figure 16 shows the angle tracking errors. Figure 17 shows the high pressure accumulator volume, the valve control signals, and the resulting knee and hip torques. Notice that the high pressure volume is nearly periodic, which is important because gait is periodic. The hip torque is close to the reference hip torque, which is also important. This ensures that the prosthesis user will not need to exert unreasonable hip torques, and this means that long-term degenerative effects of prosthesis use will be minimized. Although we do not care as much about tracking the knee torque reference since the knee torque is generated by the rotary actuator, Figure 17 shows that the knee torque is reasonable (that is, similar to human knee torque). Figure 18 shows the *x* and *y* coordinates of the hip and knee. Figure 19 shows the stick figure simulation of the reference data along with the open loop control results.

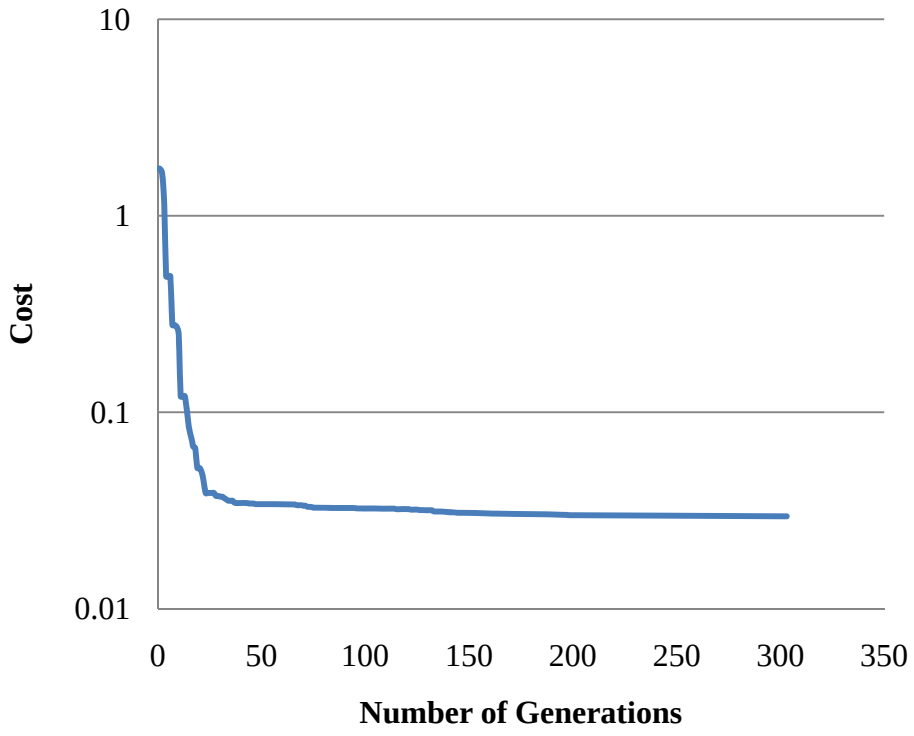

Figure 11: BBO minimum cost vs. number of generations.
Final cost = 0.029507.

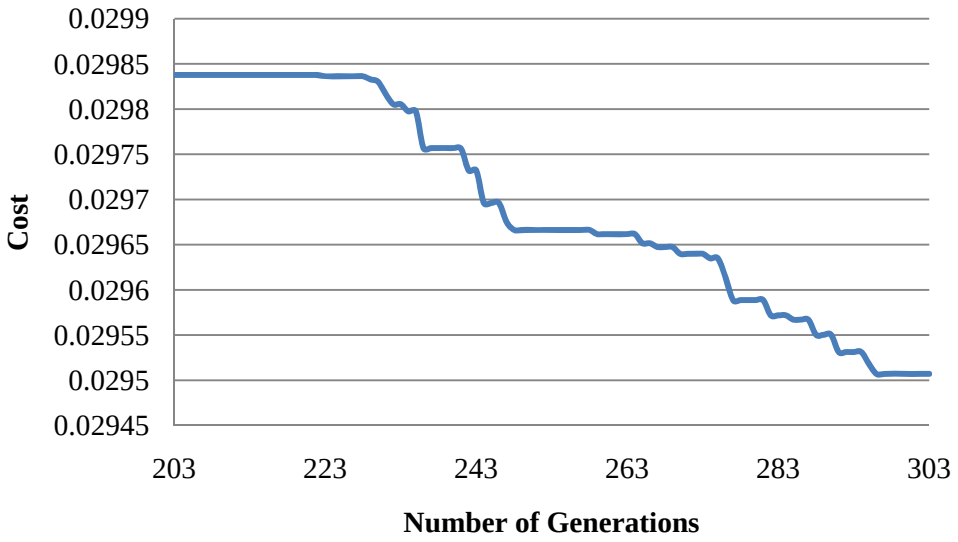

Figure 12: Close-up of the last 100 BBO generations.
Notice that the cost function is still decreasing at the end of the simulation, albeit at a very slow rate.

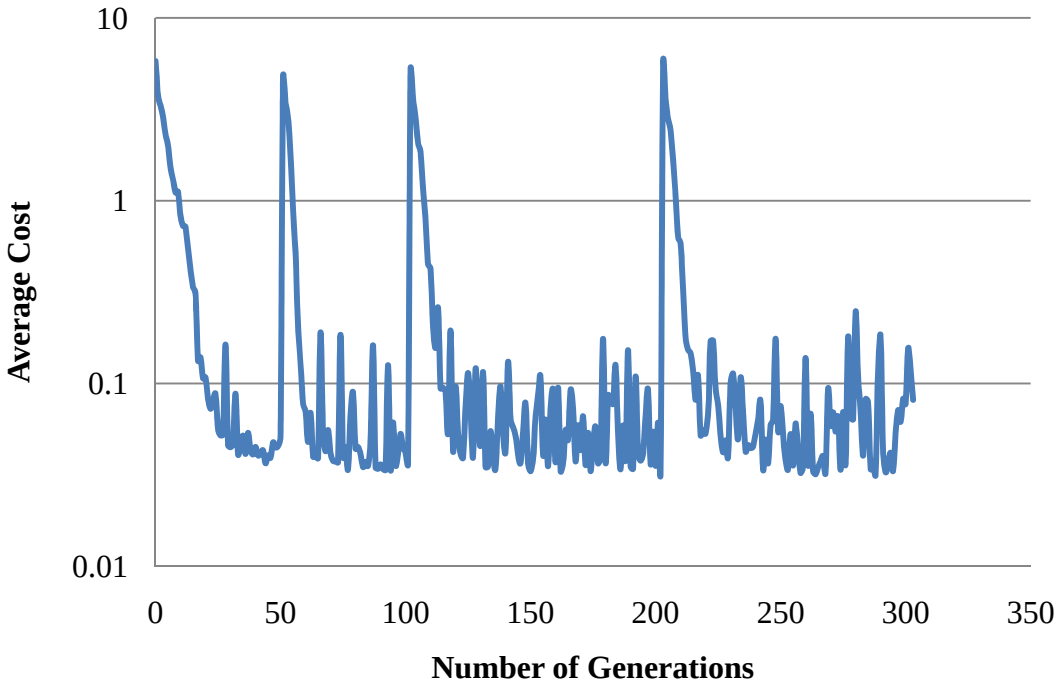

Figure 13: BBO average cost vs. number of generations.
The spikes in the average cost are where we reinitialized the population to search for a better global minimum.

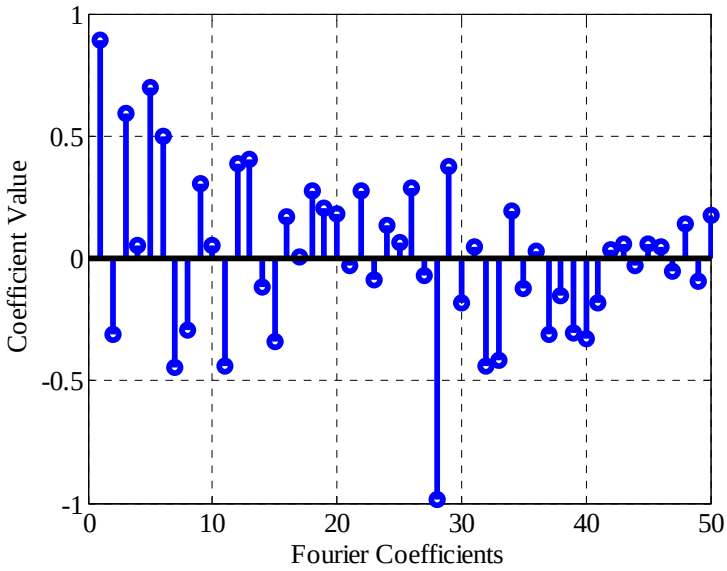

Figure 14: Best BBO Fourier coefficients after 300 generations.

Table V: Initial conditions and *k* optimized by BBO.

Parameter	$\phi_1(0)$	$\dot{\phi}_1(0)$	$\phi_k(0)$	$\dot{\phi}_k(0)$	$s(0)$	k
BBO	0.4735	-0.3323	-0.0792	-2.1453	-0.6426	5.621

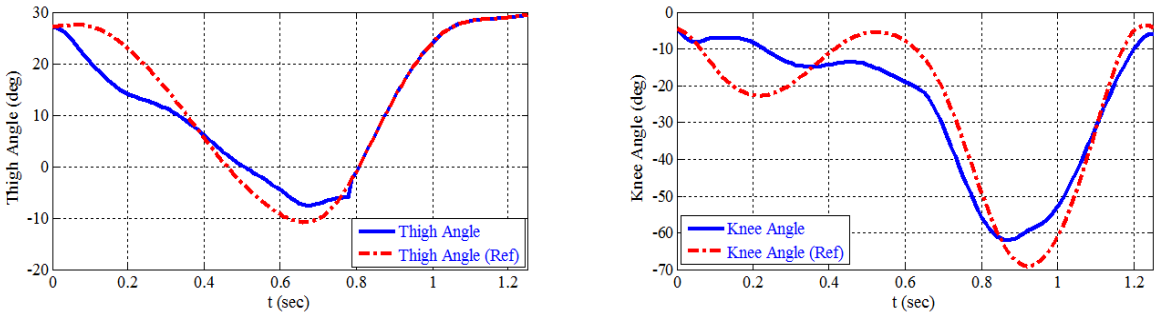

Figure 15: Thigh angle and knee angle tracking of the best candidate solution in the BBO population after 300 generations.

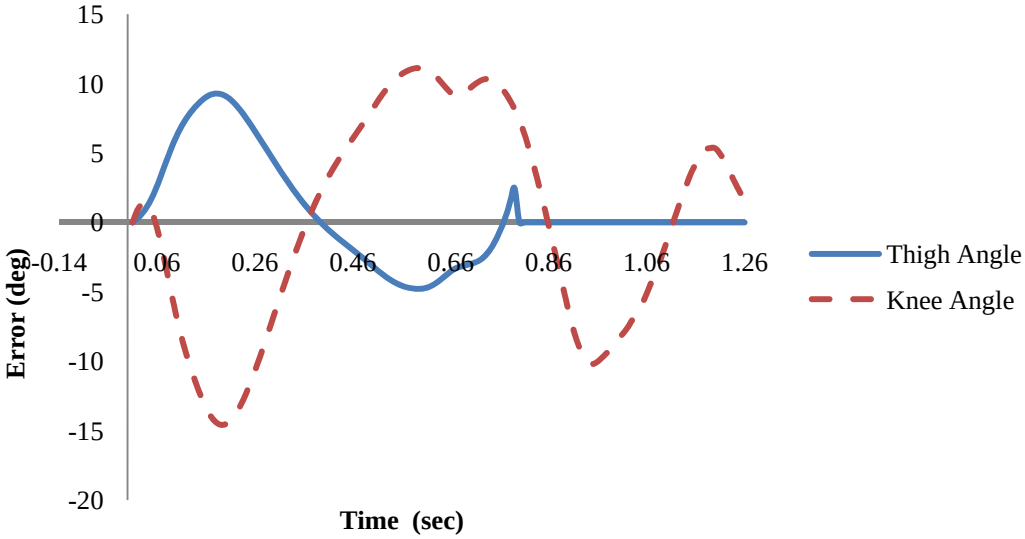

Figure 16: Thigh and knee angle tracking errors from Figure 15.

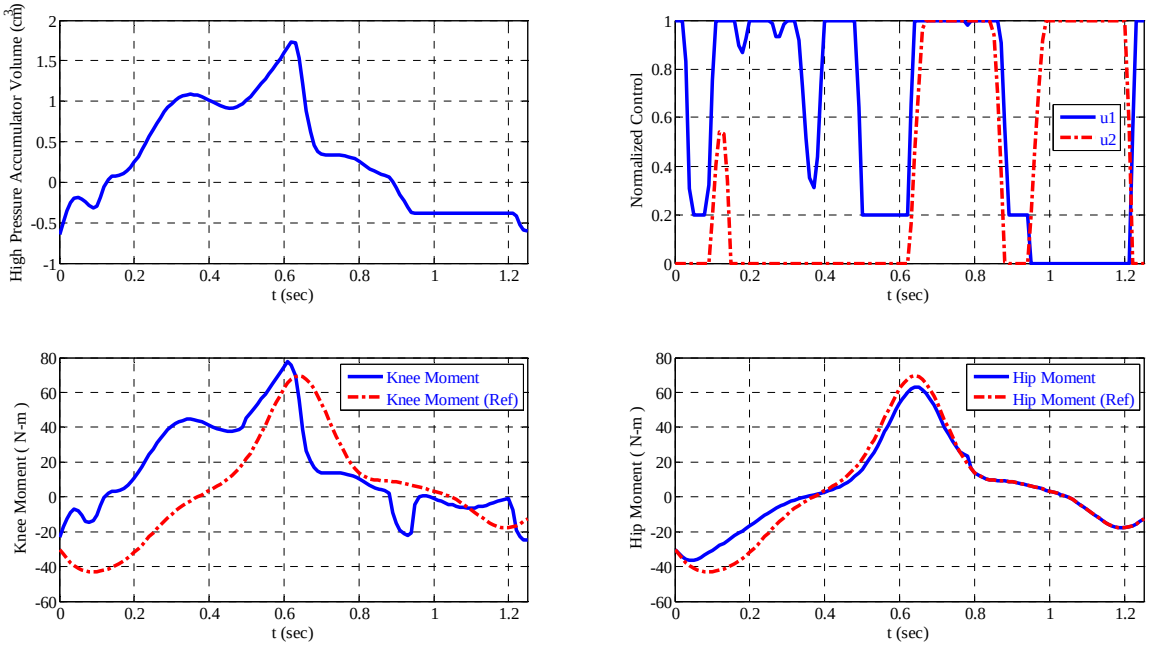

Figure 17: BBO results for open-loop control.

From Figure 17 we show the high pressure volume (top left), control signals (top right), knee torque and reference knee torque (bottom left), and hip torque and reference hip torque (bottom right). These signals show control performance of the best candidate solution in the BBO population after 300 generations.

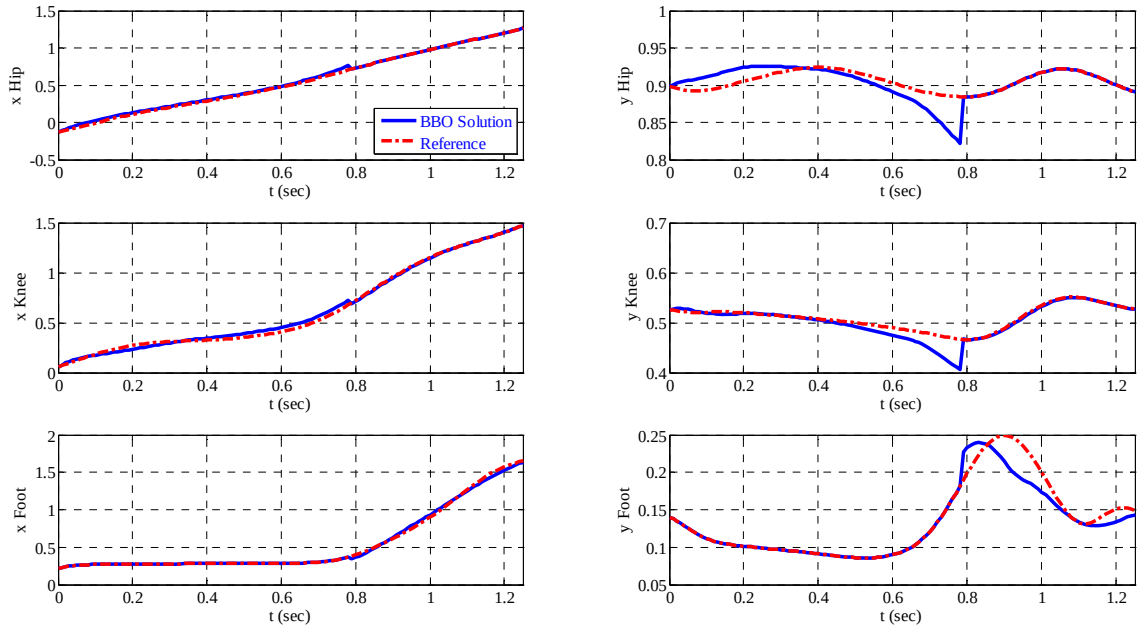

Figure 18: *x* and *y* coordinates for the hip, knee, and foot of the open-loop BBO controller.

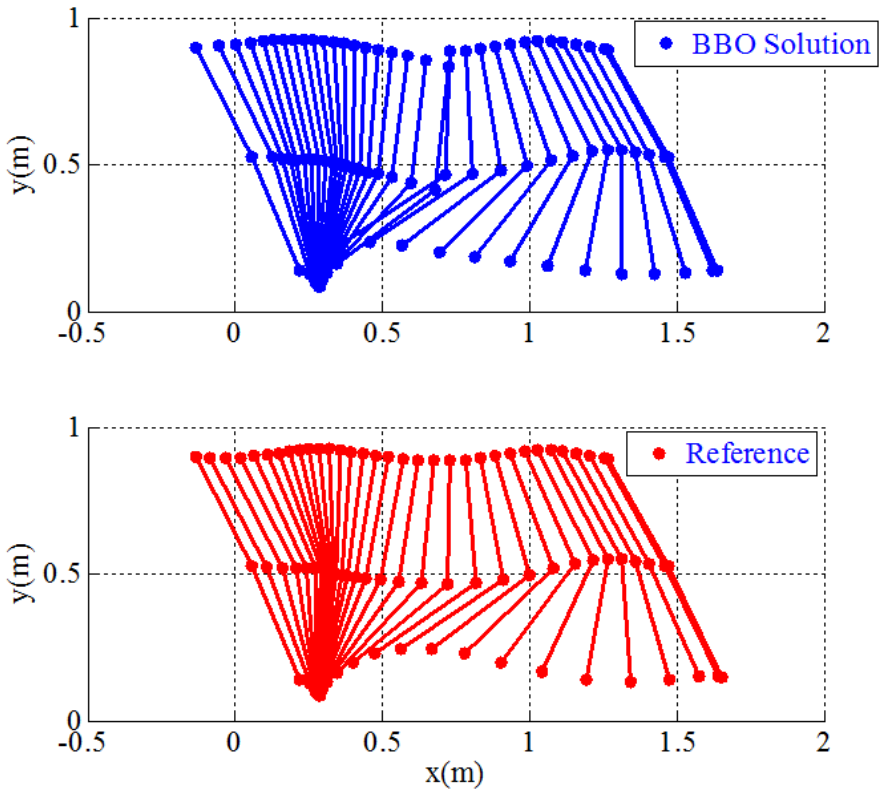

Figure 19: Stick figure simulation of the open-loop prosthesis operation compared to the reference trajectory.

We can see that BBO finds a control solution that tracks angles and requires reasonable user hip torque. However, we see that during stance phase the angle tracking is not as good as one might hope (Figure 15).This is primarily due to the fact that our model does not contain the dynamics of the foot. The foot provides key stability and support dynamics in the stance phase, and after it is incorporated in our model, we expect that BBO will be able to find a better control solution. We can particularly see that the hip *y* coordinate is falling off towards the end of stance phase (Figure 18). This effect would be mitigated with the addition of the foot dynamics since the foot lifts the body, especially during the end of stance phase.

We may also look at some data to see how important it is to follow the particular trajectory that is shown with the reference angles. The Cleveland Clinic collected several different types of gait data. We can look at this data to see how much the trajectory angles varied to get a better idea about how good our results are. Figure 20 shows the trajectories of slow walk, normal walk, fast walk, and slow run. The angles are all normalized to show angle vs. fraction of the way through the gait so that we can easily compare them. Also note that slow run is not a full gait cycle. This is because the gait lab data collection area is not big enough to capture the whole gait at higher speeds. We can still get a good idea about the variation we might expect by showing the slow run data. We only look at knee angle data because that is the angle we are directly controlling and we are only looking for an idea about how much angles might vary.

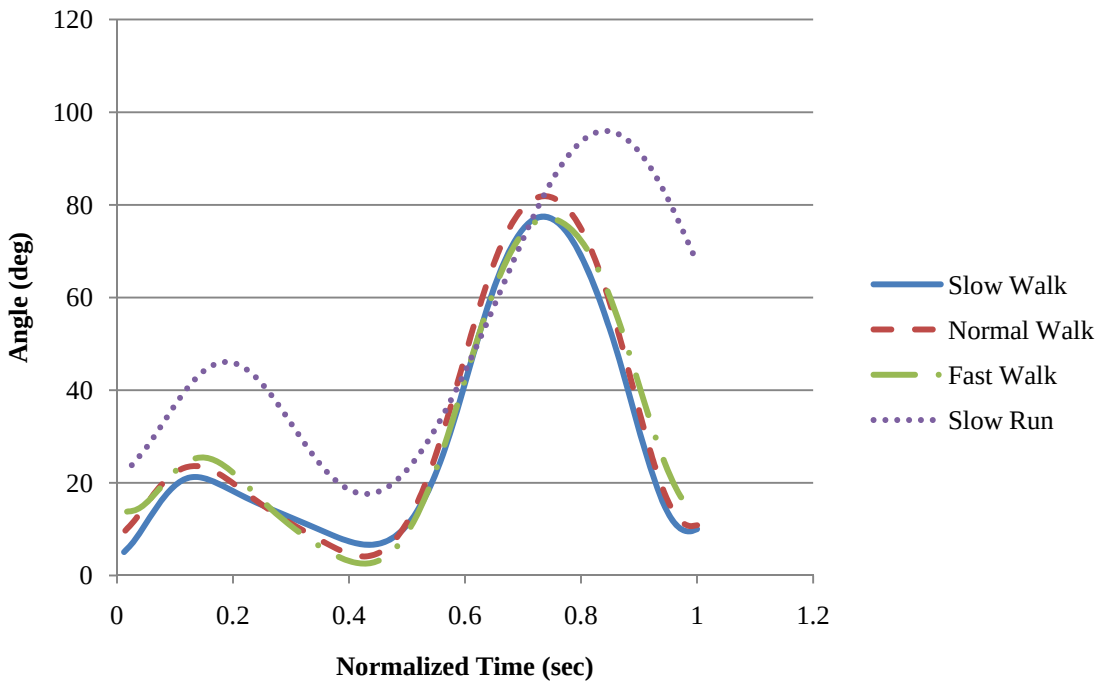

Figure 20: Knee angle variations between different gaits.

We can make a comparison to BBO open loop simulated angles by looking at the RMS difference between the different walks (excluding slow run). We see in Figure 21 that the RMS difference in the walks is comparable to the BBO RMS error shown in Figure 16. This indicates that the BBO solution is fairly good.

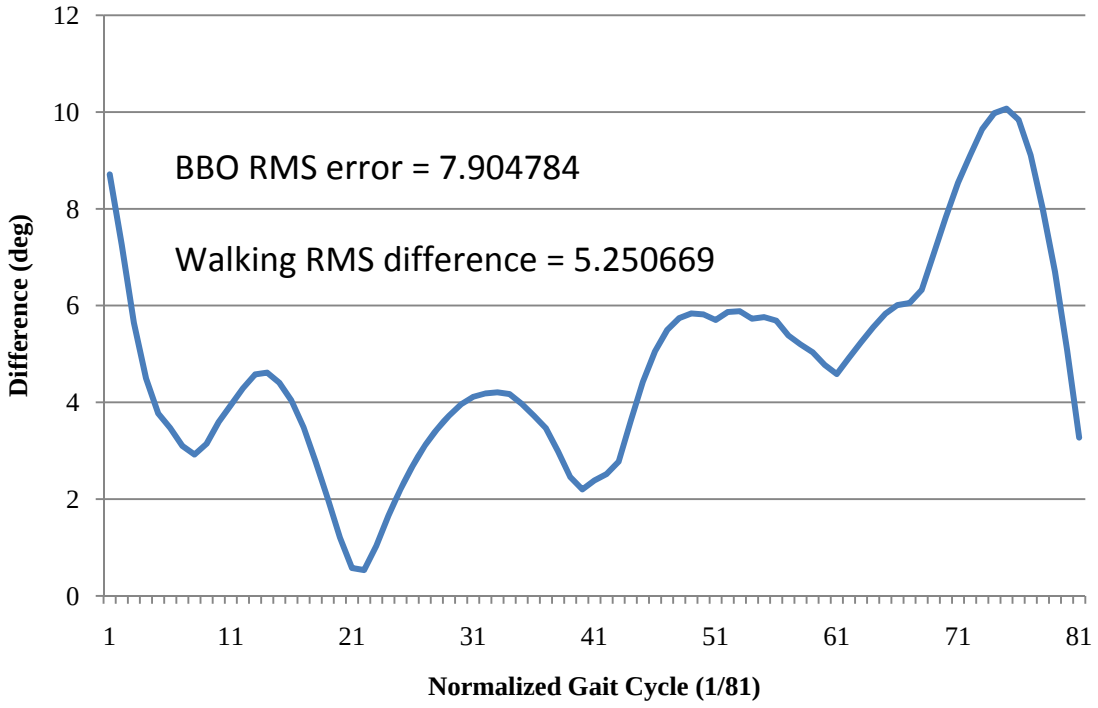

Figure 21: RMS difference for the three different types of walks.

3.3 Gradient Descent and Local Optimality

This section discusses gradient descent optimization (GDO) of the prosthetic controls, the prosthetic system initial conditions, and the rotary actuator high pressure accumulator spring constant. The improvement of the GDO generated controls relative to the BBO generated controls is very slight. However, the norm of derivative of the Hamiltonian with respect to the control is close to zero. This indicates that the BBO generated controls are locally optimal.

3.3.1 Gradient Descent Problem Setup

We give the BBO cost function in Equation 10. GDO uses the same cost function but adds a penalty term to the integrand:

$$p = \begin{cases} w_5\left(\sqrt{u_1^2(t) + u_2^2(t)} - \zeta\right)^4, & \text{if } \sqrt{u_1^2 + u_2^2} < \zeta \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where w_5 is a weight constant and $0 < \zeta < 1$ is a threshold constant. The purpose of the penalty term in GDO is to discourage GDO from bringing both controls too close to zero at the same time, where the system of equations governing the rotary actuator (Equations 4–7) have a singularity. This penalty term is not necessary in BBO since the control modification due to migration or mutation is constrained to satisfy such conditions. The weight parameters in Equation 11 are $w_5 = 10^6$, and $\zeta = 0.2$. All other cost function weights from Equation 10 are the same in GDO as they are in BBO.

The GDO algorithm consists of an outer loop and an inner loop. The inner loop optimizes the prosthetic controls for fixed initial conditions and fixed high pressure accumulator spring constant using standard optimal control theory [30]. The outer loop

optimizes the initial conditions and high pressure accumulator spring constant using static gradient descent. That is, the outer loop optimizes the control with respect to six parameters: $\phi_1(0), \dot{\phi}_1(0), \phi_k(0), \dot{\phi}_k(0), s(0), \text{and } k$. We calculate all derivatives in both the inner loop and outer loop numerically. We can separate the GDO parameters into two groups: one for the inner loop and one for the outer loop. Figure 22 shows how these two loops operate and interact.

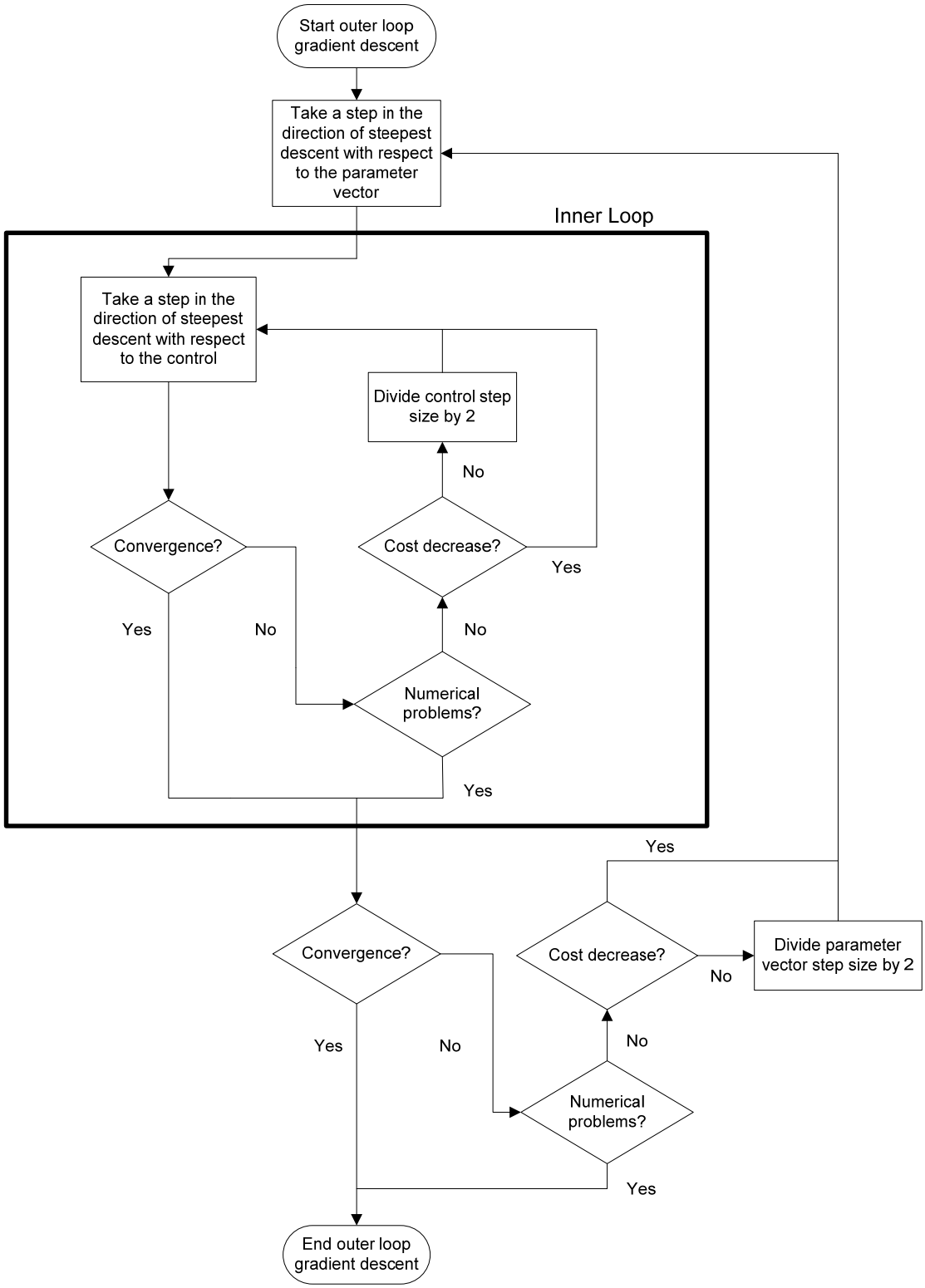

Figure 22: Gradient descent inner and outer loop interaction.
The outer loop optimizes performance with respect to the last six parameters shown in Table III. The inner loop optimizes performance with respect to the control signals.

The parameter vector consists of the initial conditions and the spring constant (that is, the last six parameters in Table VI). We use thresholds in the outer loop for the value of the norm of the gradient, the norm of the step size times the gradient, the cost decrease, and the gradient step size. We use thresholds in the inner loop for the norm of the derivative of the Hamiltonian with respect to the control, the number of iterations, and the gradient step size. When any of these parameters exceed their threshold, we exit the respective loop.

3.3.2 Gradient Descent Results

We initialized GDO with the best BBO generated controls and initial conditions. Figure 23 shows the performance improvement due to GDO during the final execution of the inner loop. GDO optimizes the control signals, the initial conditions, and the rotary actuator high pressure accumulator spring constant. The final cost of the GDO generated controls, with the optimized initial conditions and spring constant, is 0.029399; a very modest 0.37% improvement from the BBO generated cost of 0.029507. Table VI shows how GDO changes the initial conditions and spring constant. Figure 24 shows the minute changes from the original BBO control that GDO produces.

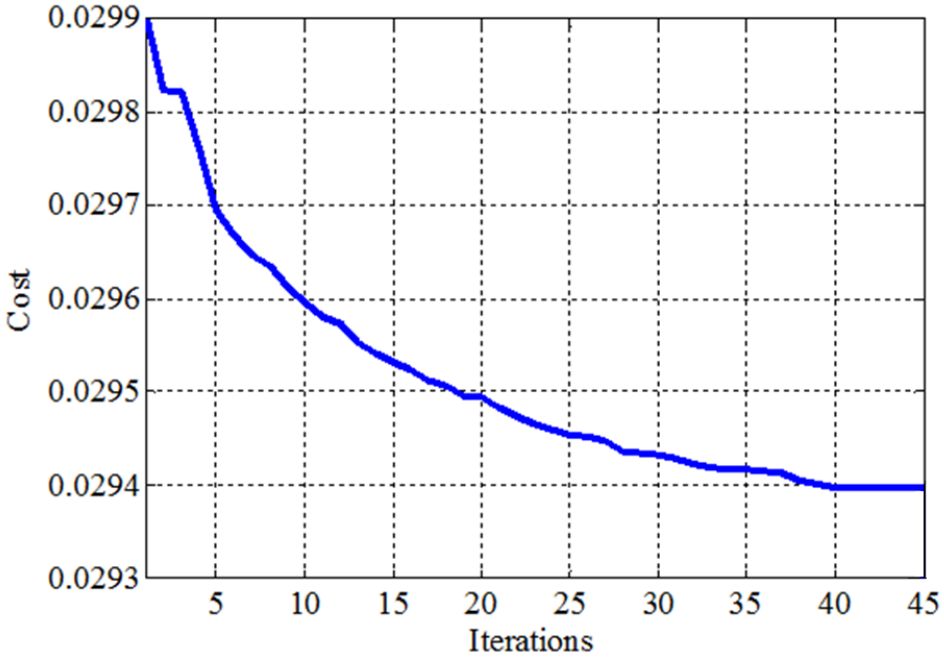

Figure 23: Convergence of gradient descent in the inner loop of Figure 22.

Table VI: Comparison of BBO vs. gradient descent optimization of initial conditions.

Parameter	$\phi_1(0)$	$\dot{\phi}_1(0)$	$\phi_k(0)$	$\dot{\phi}_k(0)$	$s(0)$	k
BBO	0.4735	−0.3323	−0.0792	−2.1453	−0.6426	5.621
GDO	0.4766	−0.3306	−0.0904	−2.1515	−0.6295	5.619

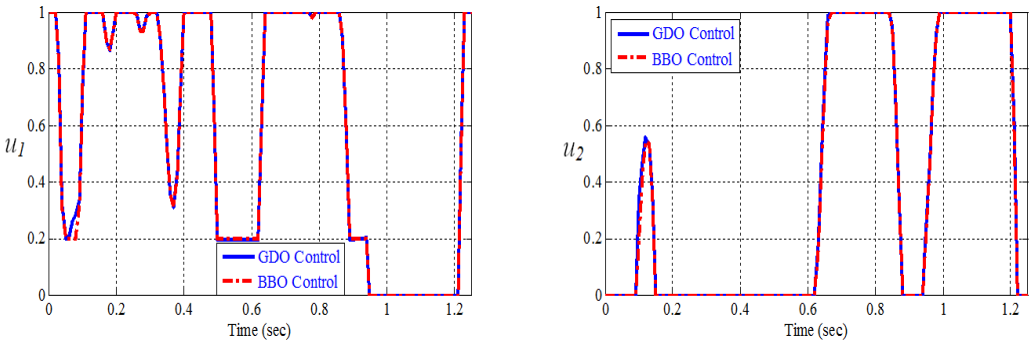

Figure 24: Control change from BBO to GDO.

Now consider the partial derivative of the Hamiltonian of the system with respect to the controls. Optimal control theory tells us that when this derivative is zero, then the control is locally optimal [30]. The two-norm of the partial derivative of the Hamiltonian of the prosthetic system dynamics with respect to the controls for the BBO generated solution is 0.061, while the value for the GDO generated solution is 0.022. This indicates that both the GDO generated controls and the BBO generated controls are close to a local optimum.

3.4 Robustness Tests

Now we discuss the results of robustness tests on the BBO generated open loop control. It is important to see how well the control performs when the initial conditions are not optimal or when the user's hip torque feedback is different than the value that we assumed when we found the optimal control using BBO.

Our first robustness test takes the open loop control and simulates the prosthesis operation when different initial conditions are used. We perform this test by adding random noise to the initial conditions. The random noise for each initial condition is taken from a normal distribution with a standard deviation of 1/10 of the optimal value. We performed this test with ten different initial conditions.

For our next test we add noise to the control signal, the feedback proportionality constant P_f used during stance phase (see Equation 8), and the thigh angle generated by the user during swing phase ($\phi_{1\,ref}$ in Figure 5). Each of these noise histories are time varying. For the feedback proportionality constant and the thigh angle data, we band limit the noise to 30 rad/sec. This simulates approximately how fast the human user might

deviate from reference values. The standard deviation of the thigh angle random noise is 0.1 radians. The standard deviation of the hip torque feedback noise is 60, and the feedback is constrained to the range [30, 200]. This ensures that the feedback is within reasonable limits. Figure 25 shows how the cost function is affected by the different types of noise. The red "No Noise" line is the BBO open loop control cost with no noise or initial condition variations imposed on it.

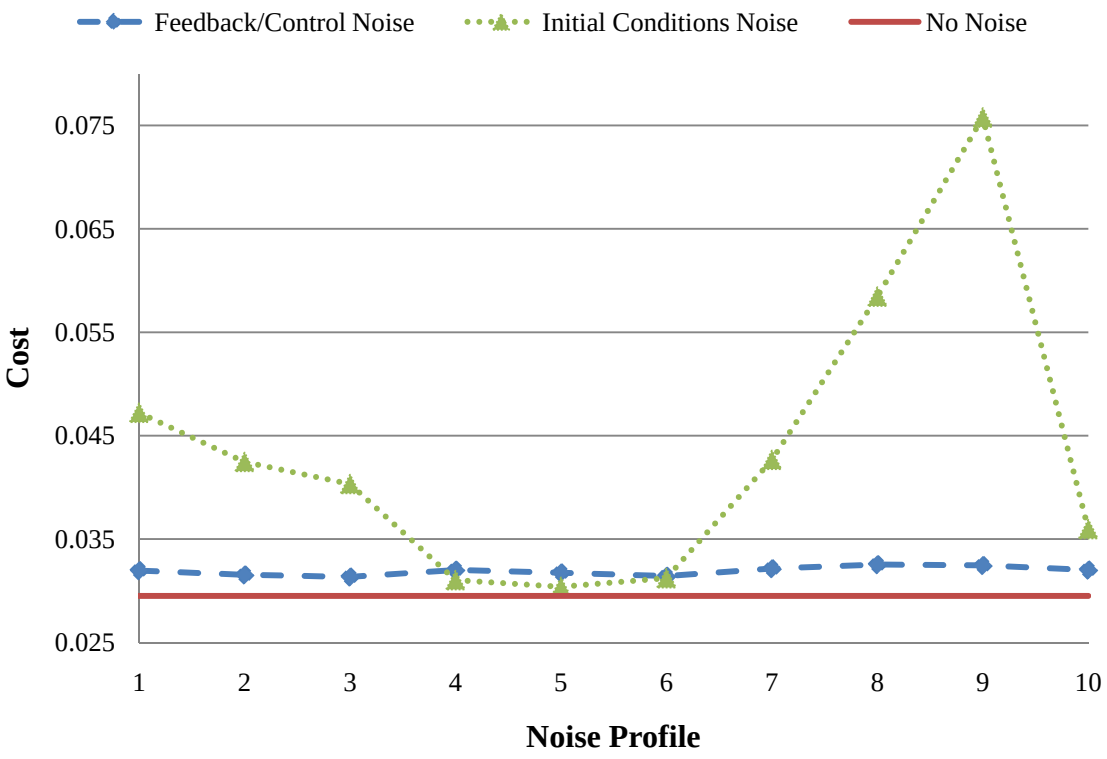

Figure 25: Cost for different types of noise and 10 different random noise profiles.

We can see in Figure 25 the feedback and control noise does not significantly affect the cost function. Therefore we conclude that the feedback and control noise does not significantly affect the performance. The BBO generated open loop solution is robust against feedback and control noise. The initial conditions, however, are more sensitive, and changing them could significantly affect performance. Changes in initial conditions require the implementation of closed loop control to generate a more robust control than can be provided by open loop control. We discuss noise in more detail in Section 5.5.1 when we discuss ANN closed loop control.

CHAPTER 4

PROPORTIONAL, INTEGRAL, DERIVATIVE CONTROL

In this chapter we discuss the proportional, integral, derivative (PID) controller and how we use it to control the prosthesis. In Section 4.1 we discuss PID controller and the interface to the system. In Section 4.2 we show the tracking results we obtained under normal conditions. In Section 4.3 we show the tracking results we obtained under varying initial conditions and we discuss the results. We will show that the PID controller is not a sufficient form of control for the prosthesis.

4.1 Introduction

The PID controller is the industry standard closed loop controller and typically exhibits robust performance. We evaluate PID control on the prosthesis to investigate whether this is an effective form of closed loop control. The primary issues that are of concern with this type of control stem from the rotary hydraulic actuator. There is a

limited source of energy in the actuator that is harvested and released. The maximum output torque depends on the available energy in the actuator at that particular time. Also, using the maximum power available at one time instant may deplete the available energy too much, resulting in a lack of needed power later in the gait cycle.

We test these theories in this chapter to see how well a simple PID controller is able to perform. Our results show that the PID controller is not a sufficient form of control. Section 4.2 shows that angle tracking is not as good as BBO. Section 4.3 shows that varying initial conditions have a worse effect when using the PID controller than with the BBO open loop control.

Recall that Chapter 3 showed that with the current mechanical design, the prosthesis is not able to track the reference angles exactly. Therefore, instead of using the reference angles for calculating errors for the PID controller, we use the optimal BBO trajectory angles to calculate errors. This way the PID controller attempts to follow the optimal BBO trajectory. We also use the optimal initial conditions from the BBO trajectory. The PID controller calculates a particular torque based on the errors. Then we calculate the torque available from the rotary hydraulic actuator. We accomplish this by fixing the system state at the current values and varying the control signals to see what torques result from the rotary equations. We set the resolution of the control signals to 0.05, and the signals are not permitted to fall below 0.1 at the same time. This is because when the control signals are small the rotary equations tend to have numerical problems. Figure 26 shows the PID control diagram.

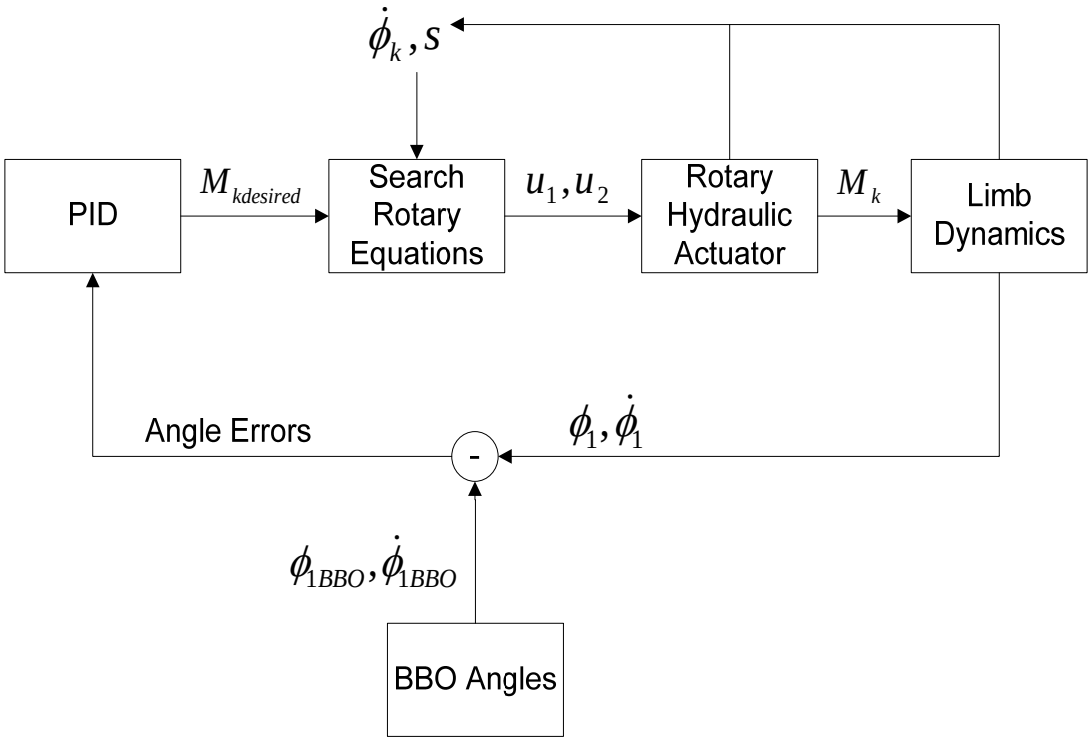

Figure 26: PID control diagram.

4.2 PID Results under Nominal Conditions

In order to achieve good tracking, we change the value of the high pressure valve size to twice the normal value. This allows a higher maximum flow rate from the high pressure reservoir and higher resulting torques. We tune the PID controller by adjusting the PID parameters and looking for good angle tracking while also supplying torques that are reasonably close to those demanded by the PID controller. We find that P = 500, D = 20, and I = 0 give the best results. Figure 27 shows the results. The knee angle is shown on the top left, the knee moment is shown on the top right, the control signal is shown on the bottom left, and the thigh angle is shown on the bottom right. The final cost is 0.054.

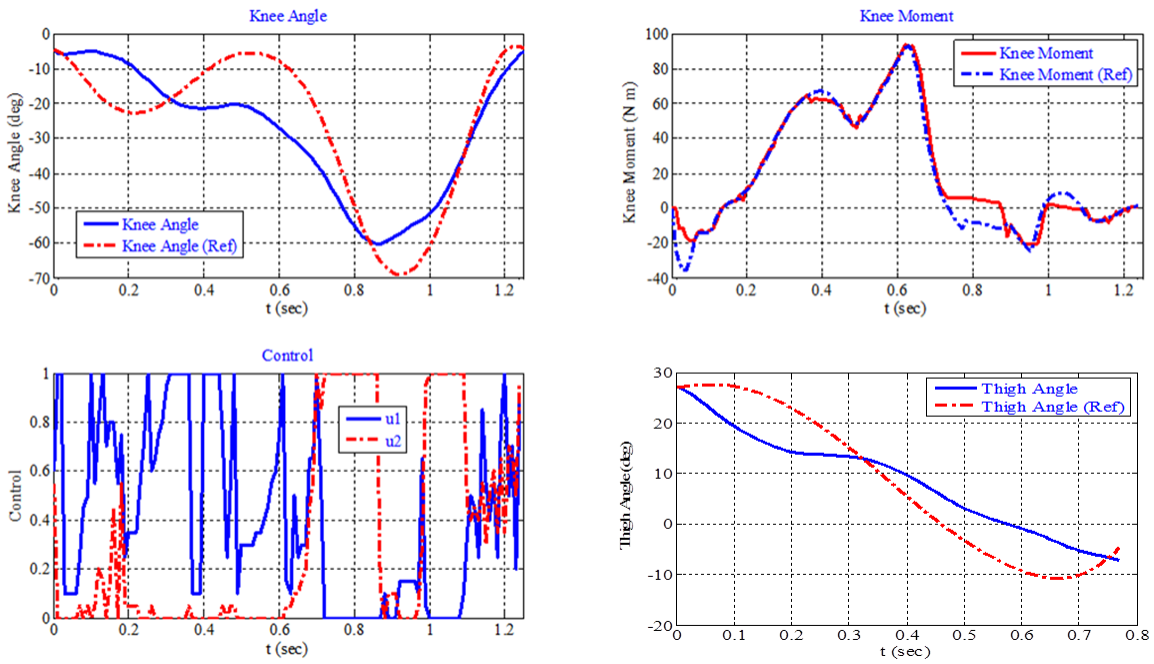

Figure 27: PID results under nominal operating conditions.

The PID controller is able to produce angle tracking comparable to the BBO generated open loop control. One point of possible concern, however, is the wildly varying control signal. At each time point the control is calculated based on which values of u_1 and u_2 will produce the closest torque to the torque that the PID controller calculates. We see that these torques line up fairly well in Figure 27. Consequently, the control signal looks undesirable from a mechanical and power consumption standpoint.

4.3 PID Results under Varying Initial Conditions

Next we would like to see how well the PID controller performs when the initial conditions are varied as they were in Chapter 3. We compare the PID results with the BBO open loop results. This gives us a good idea if the PID controller is a good design choice.

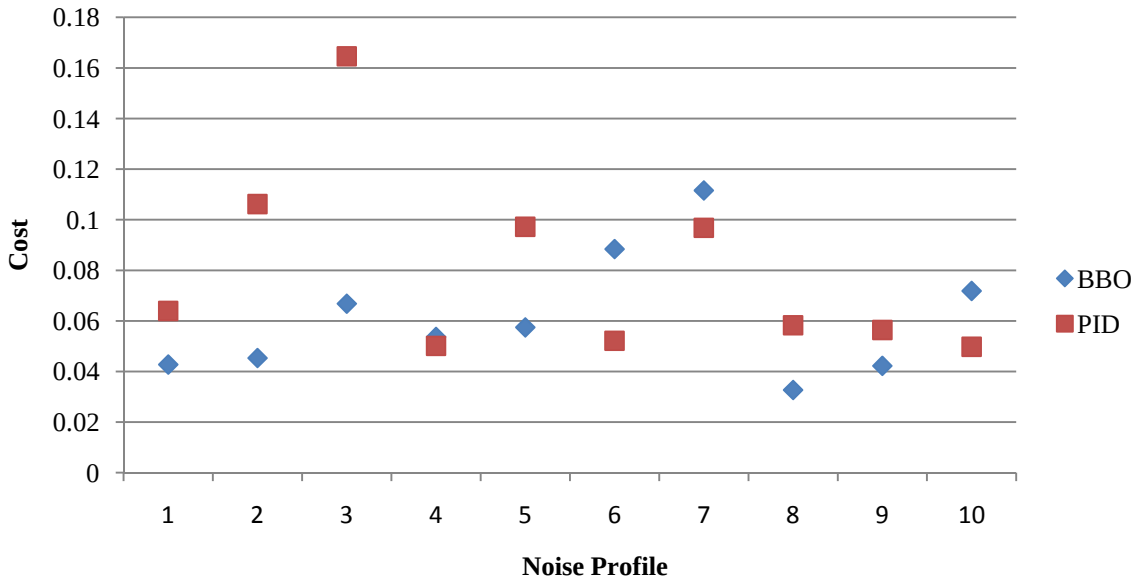

Figure 28: PID vs. BBO with initial conditions noise.

Figure 28 shows that the PID controller tends to perform slightly worse than the BBO open loop control when dealing with changes in the initial conditions. While the PID controller performs slightly better in a few instances, the average cost is higher with the PID controller than with open loop control. Also, the highest cost in Figure 28 is from the PID controller. One reason that the PID controller might not be able to perform as well is that we do not take into account the thigh angle error. Prosthetic knee operation has a complicated effect on the thigh angle that cannot be taken into account in a simple PID controller. The PID controller in its current form is not a good choice for a closed loop controller.

CHAPTER 5

ARTIFICIAL NEURAL NETWORK CLOSED LOOP CONTROL

Once we derive an optimal open loop control we need to apply closed loop control to correct for noisy measurements and disturbances. In this case, something like a simple PID controller will not be sufficient to provide the necessary correctional torque. The first problem is that we indirectly provide torque by manipulating the two hydraulic valves connected to the low and high pressure accumulators.

We have developed rotary equations to determine the torque that the rotary actuator will provide for a particular u_1, u_2, s, and $\dot{\phi}_k$. This means that we would need to have an inverse function in order to use a PID controller or a lookup table that tabulates the torque output for a particular set of inputs. We show that this inverse function does not exist in Chapter 3. It might be feasible to construct a fuzzy inference system to determine what the valve positions need to be given the state of the system. However, even if an inverse existed, or a fuzzy inference system could be constructed, we will still

have problems trying to simply apply a PID determined torque. To determine the best torque to provide the PID simply looks at correcting errors. The best torque to correct for a particular error at one time step might deplete the energy of the hydraulic actuator too soon in the gait causing the prosthesis to collapse or cause angle tracking to deteriorate significantly later in the gait cycle.

We saw in Chapter 4 that the PID controller will not be sufficient. We need a controller that can intelligently manipulate the hydraulic valves to provide the torques for different situations that result in optimal angle tracking for the entire gait. Artificial neural networks are capable of providing the solution to the closed loop control problem. Through training they can learn behaviors and generalize to handle a large number of similar cases. ANNs are attractive because of universal approximation theorems [31] and because they mimic the way that humans control natural knees. ANNs are also attractive because they do not need to explicitly incorporate any information about the system.

We show that the ANN controller is capable of mitigating the worst effects of the varying initial conditions and that other types of disturbances and noise have little effect on the angle tracking. The ANN controller does slightly better on average than with open loop control and significantly reduces high and dangerous errors that could cause stumbles and falls.

5.1 ANN Structure

Our ANN structure has several data inputs and one bias input. Adding a bias input to the network may improve performance [31]. The ANN passes the inputs through one hidden layer, and an output layer. The ANN has two outputs (delta u_1 and delta u_2). We

want the ANN to gather information about the states of the system and generate a delta control that performs a corrective action. We used sigmoidal activation functions at the hidden layer and output layer. The sigmoidal function maps outputs to values between zero and one. We want our output to be between plus and minus one so the sigmoidal outputs are mapped to this range of values. This is the largest amount we could change our control and we need the ability to change the control in both directions. Figure 29 shows the structure of an ANN with three inputs (including the bias input), three hidden neurons, and two outputs. The ANN implemented in the prosthetic knee will have more hidden neurons and, in some cases, more inputs, however the structure becomes too complex to represent.

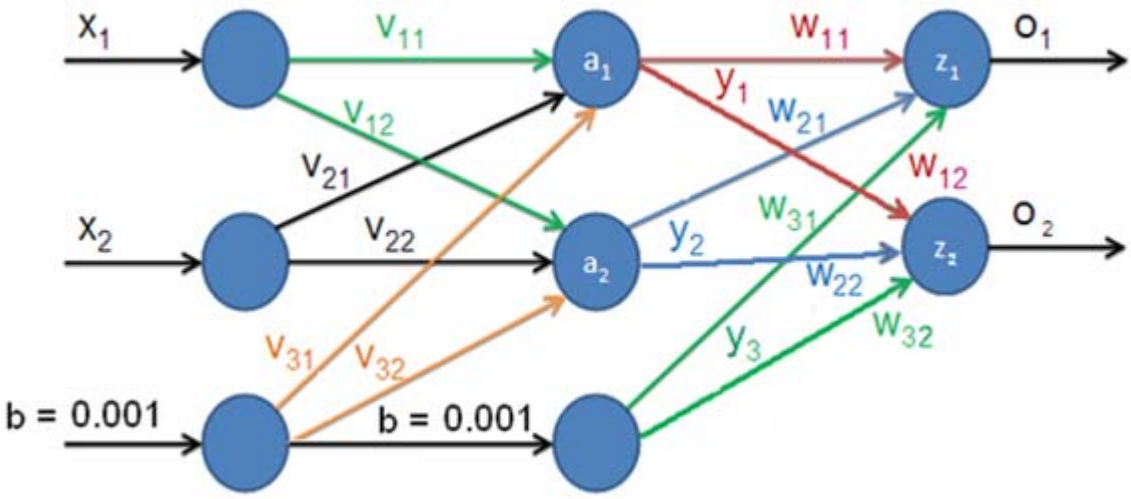

Figure 29: ANN structure.
***x* is the input vector, *b* is the bias input, *v* is the input weights to the hidden layer, *y* is the output of the hidden layer, *w* is the weights to the output neurons, and *o* is the output.**

The sigmoidal activation function is given as:

$$f(x) = (1 + e^{-x})^{-1} \tag{12}$$

We changed the bias input in our ANN from the standard of 1 to 0.001. This is because when we have zero or very small error/delta error, we want our ANN output to be close to zero. We can adjust the weights of the ANN to produce certain behaviors for certain inputs (this is called training the ANN). The typical training mechanism is called back propagation. For this mechanism to work we need to know the desired output for a given input so that we can use the partial derivative of the error with respect to the weights and propagate the error backwards through the weights. Then we could use gradient descent to minimize the error for a particular training input. We do not know what the desired output should be for a particular input so we cannot use back propagation. We instead use BBO to train the weights. BBO evaluates the fitness of an individual in the population (set of weights) by using the ANN for closed loop control and evaluating the cost function. BBO then finds the ANN that minimizes the cost function.

5.2 ANN Interface

Figure 30 shows how the ANN takes information about the system state and generates a delta control to add to the open loop control. This provides corrective actions for errors. The closed loop ANN control generates a delta control for each open loop control input. Figure 31 shows how BBO trains the ANN. BBO calculates a cost by simulating the system for each candidate ANN in the population. BBO then trains the weights for the next generation of candidate ANNs and repeats the process for the set number of generations.

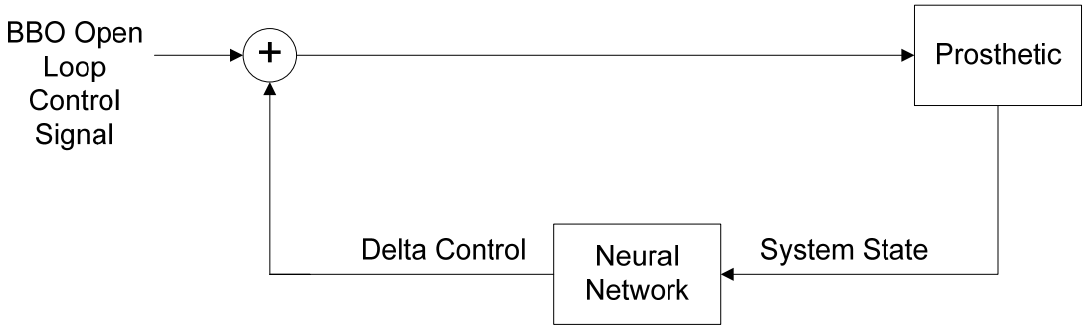

Figure 30: ANN interface with the prosthetic knee system.

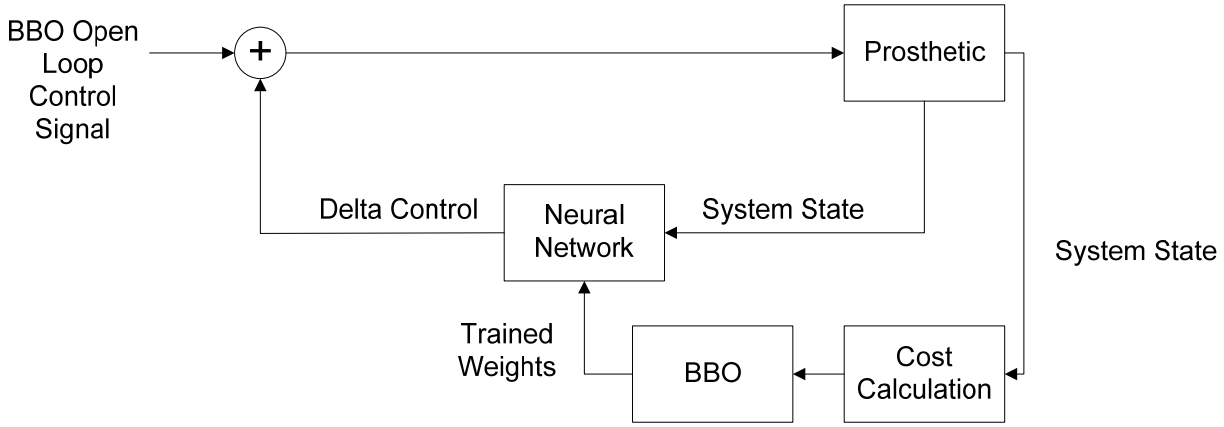

Figure 31: BBO weight training of the candidate ANN.

5.3 ANN Parameters

We want to investigate the inputs and hidden neuron structure that give optimal performance. This optimal performance consists of not only effectively reducing the error of the training data, but also having a good level of robustness to handle other situations other than the training data. Also, we want the network to converge on this solution in a reasonable amount of time. In order to answer questions about what particular network to use, we came up with a series of tests.

To test the effectiveness of adding hidden neurons to the network, we run a series of simulations. Each simulation is a BBO training session that has a population size of 50 and a generation limit of 50 and allowed weight ranges between ±0.5 (weight range values will be discussed in greater detail later in Section 4.3). Starting with four hidden neurons, we increase the number by two until the final cost stabilizes. This is a very rudimentary approach because every time neurons are added, the search space grows significantly, yet we are searching for the same amount of time in different sized search spaces. A network with four hidden neurons is simply a subset of a network with six hidden neurons. We know that compared to a four neuron network, a better solution or equal solution exists with the six neuron network.

However, we are also interested in how long it takes to find a good solution, and in keeping the network as simple as possible. When we add more hidden neurons we increase the search space. This increase in possible solutions is a benefit because it means that it is more likely that a better solution exists than with a smaller search space. However, this also means that it will take much longer to search this space and there are also many more bad solutions along with the good ones. It might be more effective to search a smaller space more efficiently for a good solution than a very large space. Also, when an ANN grows in size it tends to lose the ability to generalize and it starts to memorize a solution that is specific to the training data rather than a solution that will work generally for a larger number of situations. The bottom line is that adding neurons adds complexity to our system and it is desirable to find the simplest solution. We want to find the best solution with as few hidden neurons as possible. The test we perform is for a network trained only for stance phase. The inputs are knee angle error and thigh

angle error. Table VII summarizes the test parameters. Figure 32 shows the results of the test. The inputs are angle errors. Notice the cost stabilizes around 10 hidden neurons. Note that the cost is much higher with these trials than with the best BBO cost. This is because these tests were conducted during earlier stages of BBO development.

Table VII: Parameters used for neural network simulations.

Input Set	BBO Population size	Number of BBO Generations	Weight Range	Number of Hidden Neurons
Hip angle error, knee angle error	50	50	+0.5 to -0.5	4, 6, 8, 10, 12

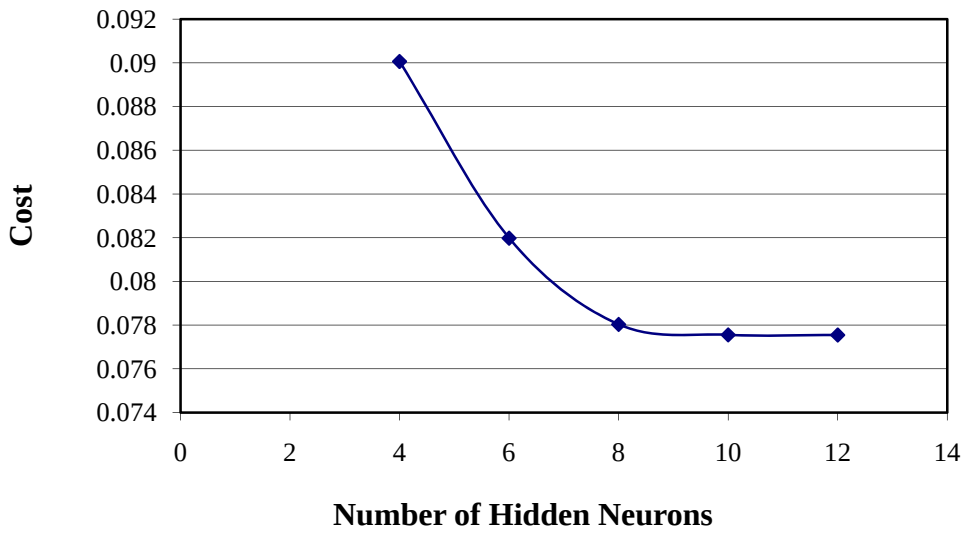

Figure 32: Cost after 50 generations vs. number of hidden neurons.

By the time we reached 10 hidden neurons the cost seemed to have stabilized. This number is a good compromise between an ANN that has a high possibility of

containing a good solution and an ANN that is not too complex. This number of hidden neurons seems to agree with current research. In [32] they use an ANN to map electromyography inputs to joint angles and torques. The ANN that they use has one hidden layer, ten hidden neurons, and one output. This is a similar ANN to the one that we are using.

Another question that presents itself while performing these tests is what the proper range of weights is for the network. Adding to the range increases the likelihood of a better solution, but also increases the search space and the time it may require to find a good solution. To help answer this question we can look at the weights that BBO chooses as the best cost for our system. If the weights are mostly tending towards the limits, then we need to increase the range. If they are mostly far from the limits, then we need to reduce the range. Figure 33 is a typical example of the weights that BBO chooses after a run. This particular run is with angle errors for the inputs, and eight hidden neurons.

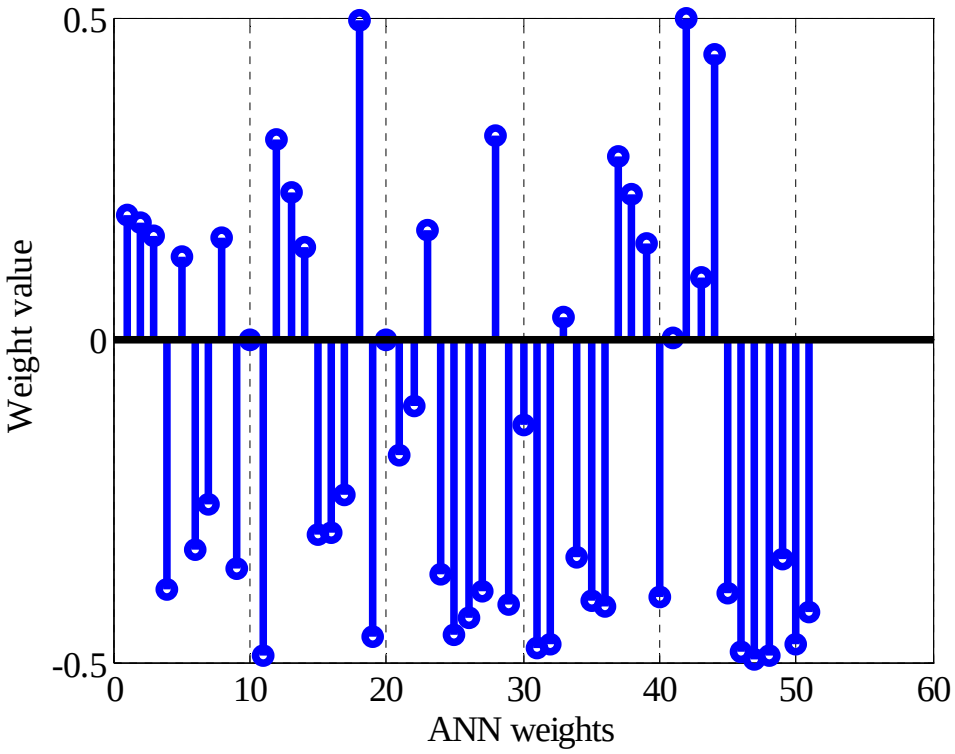

Figure 33: Stem plot of ANN weights for eight hidden neurons.

We can see that the weights hold a wide range of values and do not necessarily tend towards the upper or lower bound. This indicates that a good solution can be found with weight ranges of ±0.5.

5.4 ANN Tests

We devised several tests to determine the effectiveness of using artificial neural networks for closed loop control. We saw in Chapter 3 that BBO is able to find a control that is locally optimal; therefore we do not try to improve the open loop control through adding closed loop control. Instead we add different disturbances and noise to the system and attempt to mitigate the detrimental effects with closed loop control.

5.4.1 Noise and Disturbance

We add several different disturbances and noises into the system to see how the output is affected. We model all of the noise and disturbances by adding a normally distributed random number with an appropriate standard deviation. We can select different profiles of random noise added by selecting a particular seed value. This way we can repeat random noise profiles for training purposes and use a particular seed value to objectively compare the performance of the closed loop control to the open loop control.

The first disturbance that we can add to the system is random noise to the control signal. Some noise is to be expected with any measurement value and output value in a real system. We model the control system noise by adding a random number to the control signal at each time the control is used to calculate the system response. This means that the random noise will not be added to the signal at 100 Hz but at every time the differential equation solver in MATLAB calls the routine that calculates the system. The random noise is normally distributed, as is all of the noise, with a standard deviation of 0.01, which is normalized to the range [0, 1].

We also add disturbance to the hip feedback proportionality constant used to correct thigh angle errors during the stance phase of the gait. This simulates the fact that the user will not react exactly as we have calculated when deriving the open loop control. We simulate the user's possible differences from the open loop assumption by adding a random number to the feedback term. The random number has a standard deviation of 60. This means that there is possibility for large variations in the feedback term. We add upper and lower bounds on the feedback term to ensure that the feedback term is still

reasonable. The upper bound is 200, which is twice the standard of 100 but still well within human capability. The lower bound is 30 which is much less than the standard. We band limit this random number to 30 radians/second by passing white noise through a low pass filter. We use band limited noise because we know that the user can only react at relatively low frequencies. We add the noise to the feedback term each time the system is calculated.

We also know that the user will not control the thigh angle exactly like we have simulated it. When calculating open loop control we assume that the thigh angle is exactly equal to the reference thigh angle calculated from one human subject in the gait lab. We know that even the reference angle will have a standard deviation among healthy non-amputees. An amputee's thigh angle during swing phase will also be affected by the addition of the prosthesis as it will add unnatural torques that are different from the torques that a healthy knee might exert on the subject. Therefore we can also add noise to the reference thigh angle during swing phase. The noise has a standard deviation of 0.1 radians. We band limit the noise frequency to 30 radians/second. We also add this same type of noise to the derivative and second derivative of the knee angle.

Another set of parameters we can add noise to are the optimal initial conditions found though BBO. These initial conditions are in no way guaranteed to be exactly the same as in the simulation. With training, the amputee might be able to somewhat accurately reproduce the human controlled initial conditions for the gait but there will be some differences. We can add a random variable to each one of the initial condition parameters to help simulate reality. The random numbers we add to the initial conditions have a standard deviation of 10% of the optimal value. It will be shown that the control is

very sensitive to the initial conditions. This indicates that much training will be needed with the current design to ensure that the user can get within 10% of the optimal values.

Finally, we can add noise to the measured values that the feedback controller uses to calculate the closed loop control. All measurements will have some associated noise. We can deal with this noise to an extent by adding filters. The standard deviation of the measurement noise is 0.01 radians. This is simply to observe the effect of noisy measurements on the performance of the closed loop controller. We show all the different noise parameters in Table VIII.

Table VIII: Noise parameters for ANN training and testing.

Noise Type	**Standard Deviation**	**Frequency**
Control Signal	0.01 (control signal between 0 and 1)	No Filter
Hip Feedback Parameter	60 Newton*Meters / Radian	30 Rad/Sec
Swing Phase Reference Angles	0.1 Radians	30 Rad/Sec
Initial Conditions	10% of BBO Value	Not Applicable
Measurement	0.01 Radians	No Filter

5.4.2 ANN Training

One of the first challenges is to decide how we should train the ANN. Due to long training times, we increased the simulation time step from 0.01 seconds in Chapter 3 to 0.02 seconds. This significantly reduces simulation and training times but also slightly increases the cost. However, for a proof of concept the slight increase of all costs involved are inconsequential. The increase in cost is due to the loss in control resolution.

Initially we thought that one ANN for the entire gait cycle would be good enough. It would keep things simple and hopefully provide the robustness we are looking for. However, the network did not perform exceedingly well and new formulations in logic brought us to a different conclusion. The closed loop controller should consist of two different ANNs. One ANN is trained for stance phase and one ANN is trained for swing phase. This makes sense because stance phase and swing phase are two entirely different systems. A single ANN trained for the whole gait cycle would not be able to make the correct decisions for both systems. Instead it would make decisions that are compromising for both stance and swing phase in an attempt to decrease the average cost.

The inputs we use for training the ANNs are based on the experimental ANN in Section 5.3. The inputs for stance phase are knee angle error, thigh angle error, knee angle velocity error, and the small bias input. Swing phase will not have the thigh angle error because we assume that the operation of the prosthesis has negligible effects on the thigh angle during swing phase. Therefore the inputs for swing phase are knee angle error, knee angle velocity error, and the small bias input. We add the knee angle velocity error as the only change from the experimental network in Section 5.3. We did this to add important information to the network while not adding too much complexity.

Logic suggests that two separate ANNs would be more effective for closed loop control. Stance phase and swing phase are two completely different dynamic systems, so we should train two separate ANNs. One will provide control for stance phase and one will provide control for swing phase. This also requires that we train the networks separately. First we will train stance phase. Then we can train swing phase by running the stance phase ANN simulation and using the final conditions of the stance phase as the

initial conditions of the swing phase. We can use any number of the different types of noise described in Section 5.4.1 to impose on the open loop control and then train the ANN to minimize the cost of imposing the noise with BBO. In order to ensure that the ANN will be able to handle many different noise cases, we run the simulation a number of times for each candidate ANN in the BBO population. Each time we use a different seed value for the random number generator. After the candidate ANN has been evaluated the specified number of times, we can assign a cost to the ANN that is some function of the separate evaluations. This way the ANN gives a better performance on average for a large number of noise cases.

Training two separate ANNs poses some difficulties. We have to decide what cost function will provide the best overall tracking, not just the best tracking during stance or swing phase. When training a stance phase ANN we need to make sure that the network will not only attempt to minimize angle tracking error, but also the final conditions of stance phase must be favorable initial conditions for swing phase. We tried many different cost functions for stance phase to find the best way to optimize these two parameters. One approach was to use the BBO optimal trajectory as the reference angle data for error calculation and cost calculation. We could use a cost function that utilized a weighted cost for the angle tracking data and weighted endpoint cost so as to help ensure that swing phase will start with favorable initial conditions. We experimented with many different weights for the different cost parameters with some success, however we settled on a different technique for ensuring good angle tracking as well as good final conditions. To train the stance phase ANN we evaluated the cost of the ANN by evaluating the cost of the whole gait cycle using the reference data from the gait lab to calculate errors and

costs. We use closed loop control for stance phase and open loop control for swing phase. This way we avoid trying to determine what the best weights should be for the endpoint values. If closed loop control provides good initial conditions for swing phase, than the open loop BBO generated control will give a better swing phase response. The cost function we used for stance phase is shown as:

$$J_{st} = \int_{t=0}^{T} \left[w_1 \left(\phi_1(t) - \phi_{1ref}(t) \right)^2 + w_2 \left(\phi_k(t) - \phi_{kref}(t) \right)^2 + w_3 U\left(\phi_k(t) \right) \phi_k(t) \right] dt \tag{13}$$

With this cost function we choose the weights to be $w_1 = w_2 = 1$, and $w_3 = 2$. The swing phase ANN cost function is the same as the stance phase cost function except that it is only integrated over swing phase operation. We are only concerned with single strides at this time and we are not researching the effects of the swing phase final conditions. The swing phase cost function can be shown as:

$$J_{sw} = \int_{t=T_{st}}^{T} \left[w_1 \left(\phi_1(t) - \phi_{1ref}(t) \right)^2 + w_2 \left(\phi_k(t) - \phi_{kref}(t) \right)^2 + w_3 U\left(\phi_k(t) \right) \phi_k(t) \right] dt \tag{14}$$

Note that T_{st} is the stance phase final time. With this cost function we choose the weights to be $w_1 = 0$ $w_2 = 1$, and $w_3 = 2$. Remember that we do not need to assign a cost to the thigh angle in swing phase operation. These cost functions will assign a cost to the candidate ANN for a particular noise profile. We train the ANN with more than one noise profile so we will have an array of costs ($J = J_{ST}$ or J_{SW} depending on which ANN we are training) that will be used to assign the final cost. To assign the final cost to the stance or swing phase ANN we use the following cost function:

$$J_{ANN} = Max(J) + w * Average(J) \tag{15}$$

The motivation for this cost function is twofold. First we are interested in ensuring safe operation of the prosthesis. This means we want consistent results that are not wildly affected by the noise. We especially want to reduce high costs that might cause

the user to stumble or have especially detrimental side effects with prolonged use. Second, we want better overall performance over open loop control when dealing with noise. The weight $w = 0.5$, which has shown to give an effective balance between minimizing the highest and average cost.

5.5 ANN Results

To prove the concept that ANN closed loop control can provide robustness from noise we train the ANN using the initial conditions noise. This noise has the most dramatic effect on the open loop control as we have seen in Figure 25. We train each ANN (stance phase and swing phase) with 10 different noise profiles. Then we evaluate the performance of the ANN on 20 different noise profiles. This shows how well the ANN is able to perform on the training noise profiles as well as how it is able to generalize over noise profiles not trained with. Note that we are using a different open loop control than is found in Chapter 5. This is because the infrastructure of the ANN training code is based on an older BBO open loop result. We are also simulating the code with a 0.02 second time step instead of our normal 0.01 second time step. We do this to decrease the training time because training with many noise profiles is very time consuming. As a result the costs shown will be slightly higher than they would otherwise be. Figure 34 shows the cost for BBO only (open loop), and ANN (closed loop) control for 20 different noise profiles. The first 10 noise profiles are the profiles that the ANN was trained with and the next 10 are profiles the ANN was not specifically trained for.

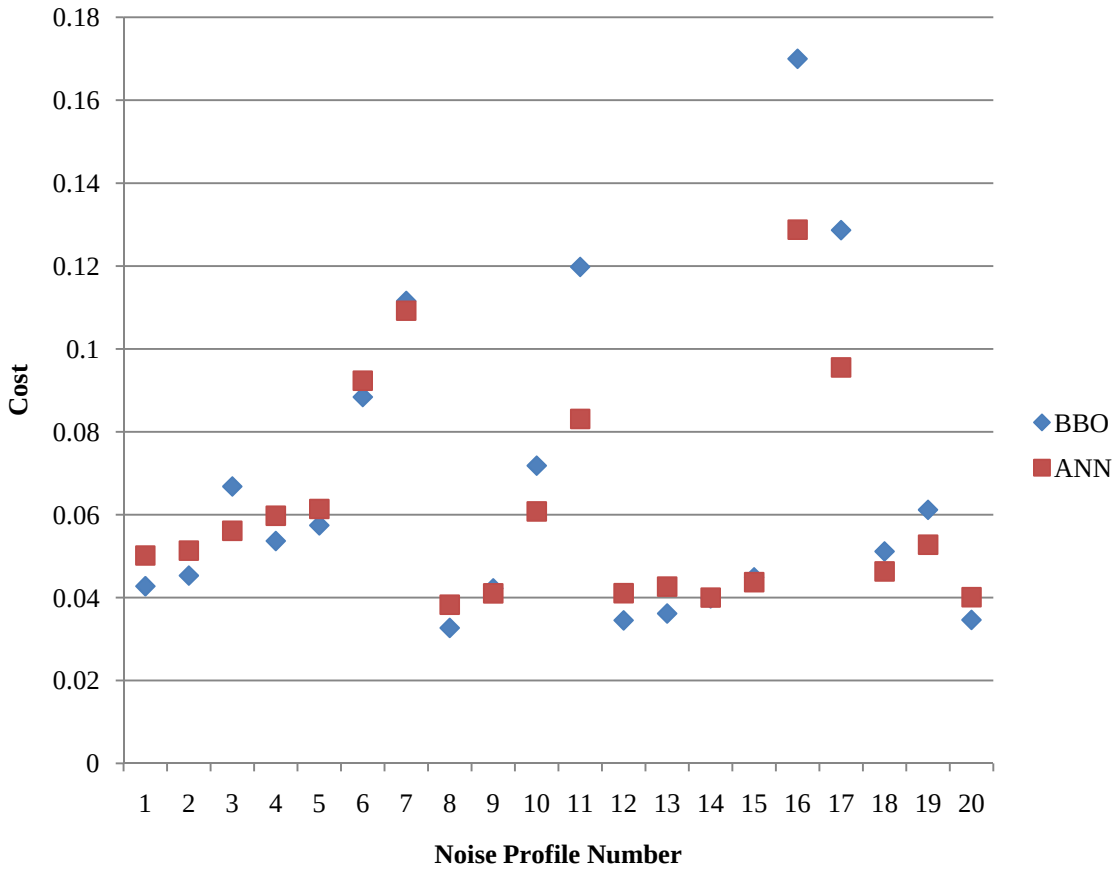

Figure 34: Cost vs. noise profile number (varying initial conditions is the only noise imposed). The first 10 noise profiles are training cases, and the last 10 are test cases.

Notice that although the closed loop cost is not always lower than the open loop cost, the noise that causes the highest costs benefits greatly from the ANN control. The overall average cost is also lower with the ANN control than with BBO alone, as we show in Table IX. This is what we want to see from our closed loop control: lower costs on average and consistent lowering of especially high and potentially dangerous costs.

Next we check to see if the ANN is able to generalize its error correcting capabilities even further by adding control signal noise and user feedback noise to the varying initial conditions and running the same 20 noise profiles. These results are shown in Figure 35. We see that the ANN is able to generalize well into other sources of error

and even generates a larger percent average cost lowering (shown in Table IX) than it did with the initial conditions noise alone.

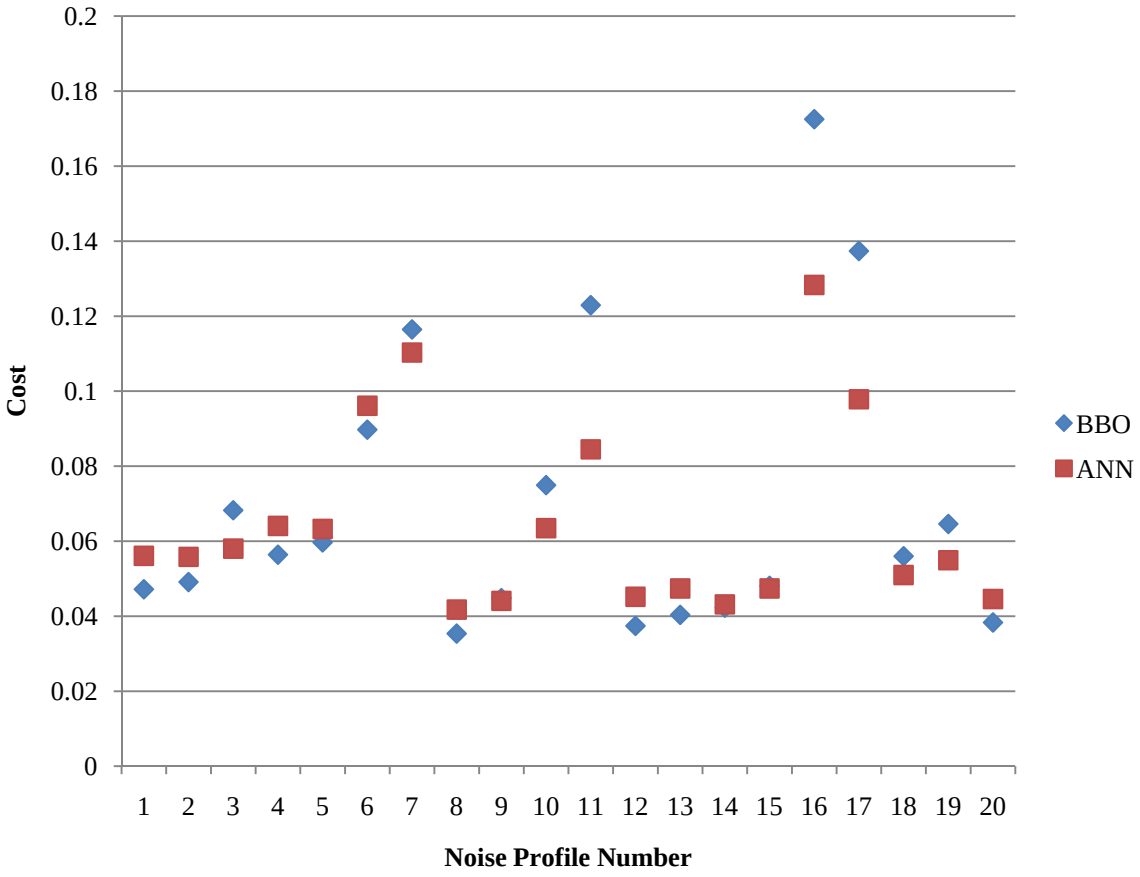

Figure 35: Cost vs. noise profile number (initial conditions noise, control signal noise, user feedback noise). The first 10 noise profiles are training cases, and the last 10 are test cases.

Finally, we can observe the effect on performance of adding measurement noise to the ANN. Figure 36 shows the performance. Small amounts of measurement noise have minimal side effects to the ANN's performance. With these tests we can see that ANNs are able to provide robustness against noise and disturbances over open loop control alone.

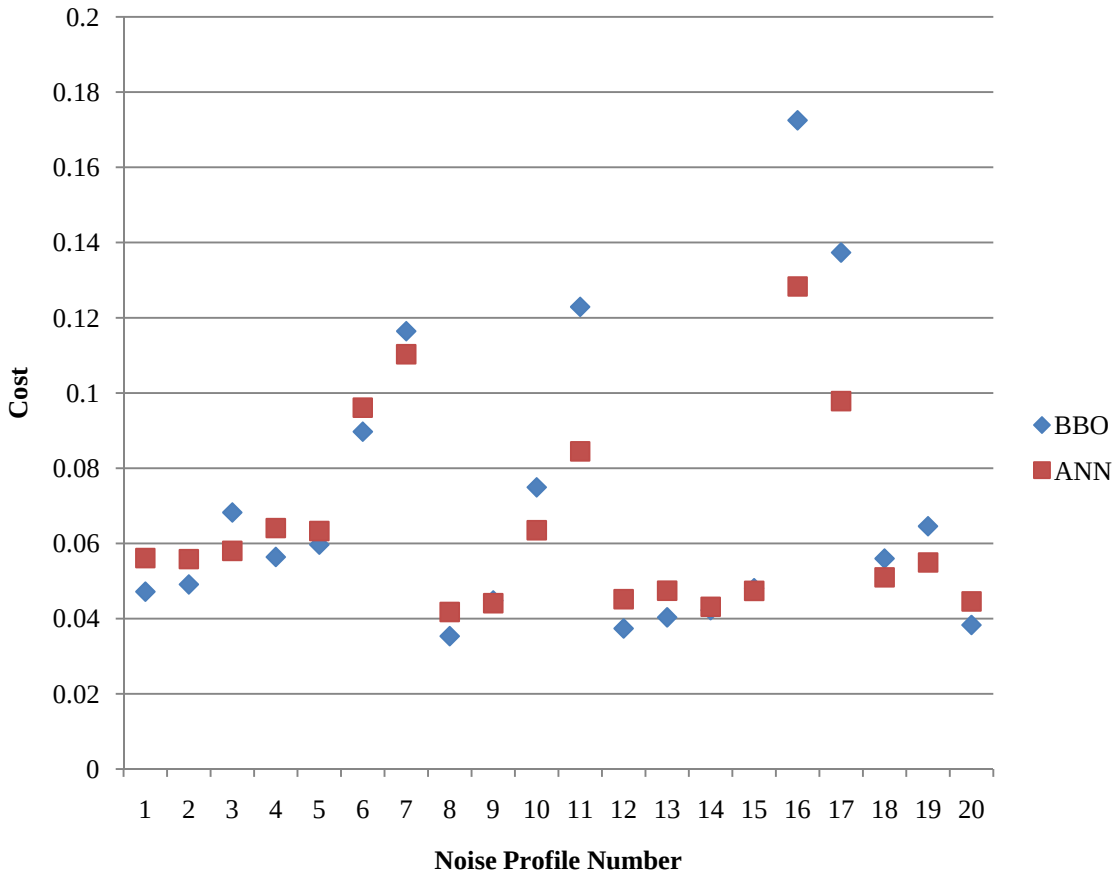

Figure 36: Cost vs. noise profile number (initial conditions noise, control signal noise, user feedback noise, measurement noise). The first 10 noise profiles are training cases, and the last 10 are test cases.

Table IX: Open-loop BBO vs. closed-loop ANN average cost.

Noise Type	BBO Cost	ANN Cost	% Difference
Initial Conditions	0.066671	0.061187	8.23
Initial Conditions, Feedback, Control Signal	0.070077	0.064058	8.59
Initial Conditions, Feedback, Control Signal, Measurement	0.070077	0.063881	8.84

We are especially interested in the ankle *y*-coordinate of the prosthesis when evaluating a closed loop controller's performance. Stumble prevention is the most important task of the closed loop controller, and stubbing the toe of the prosthetic during

the swing phase of operation is a likely way to cause a stumble. We show the reference ankle *y*-coordinate for swing and stance phase in Figure 37. We show the ankle *y*-coordinate generated by the same 20 initial condition noise profiles we have been using for BBO and ANN control in Figure 38 and Figure 39 respectively. We show that the ANN closed loop controller is less likely to cause a stumble by the prosthetic stubbing its toe. We come to this conclusion because the closer the ankle is to the ground during swing phase, the closer the foot will be to the ground. The foot has a greater chance of striking the ground prematurely if the ankle is too close to the ground. Note that during stance phase we use the reference ankle coordinate. Notice in Figure 38 and Figure 39 how the *y*-coordinate falls below the reference during swing phase. Another factor determining whether the foot will hit the ground accidentally or not depends on what type of foot will be implemented in the prosthetic and its dimensions. All we can say at this point is that the ANN controller is less likely to cause these sorts of accidents.

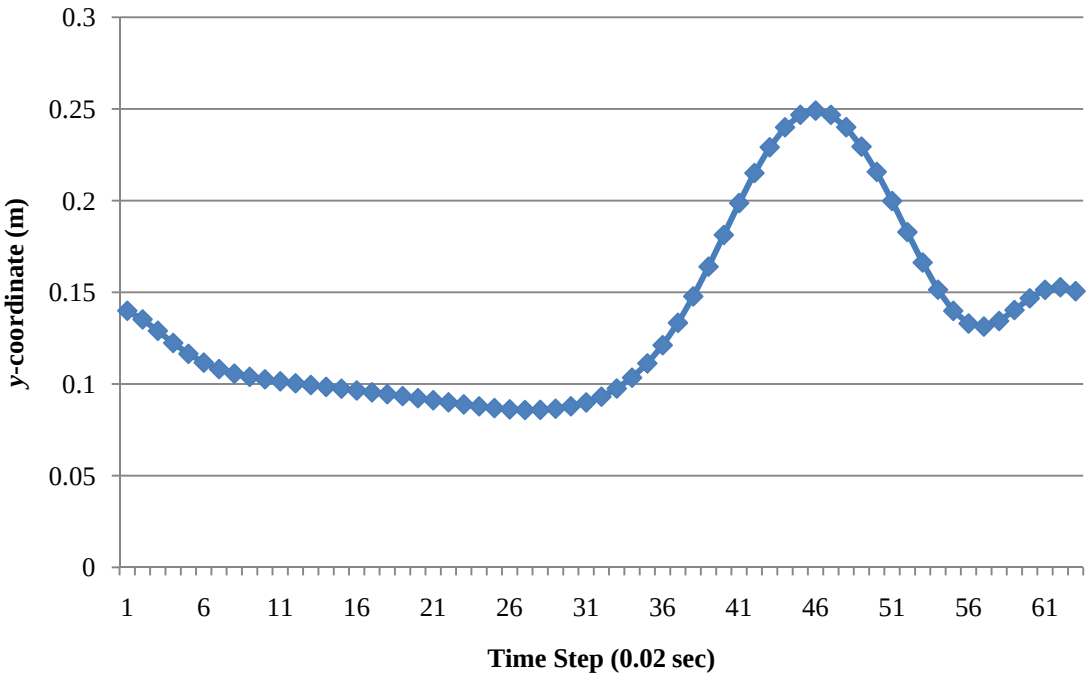

Figure 37: Reference ankle y-coordinate.
Each time step number on the horizontal axis is a 0.02-second interval.

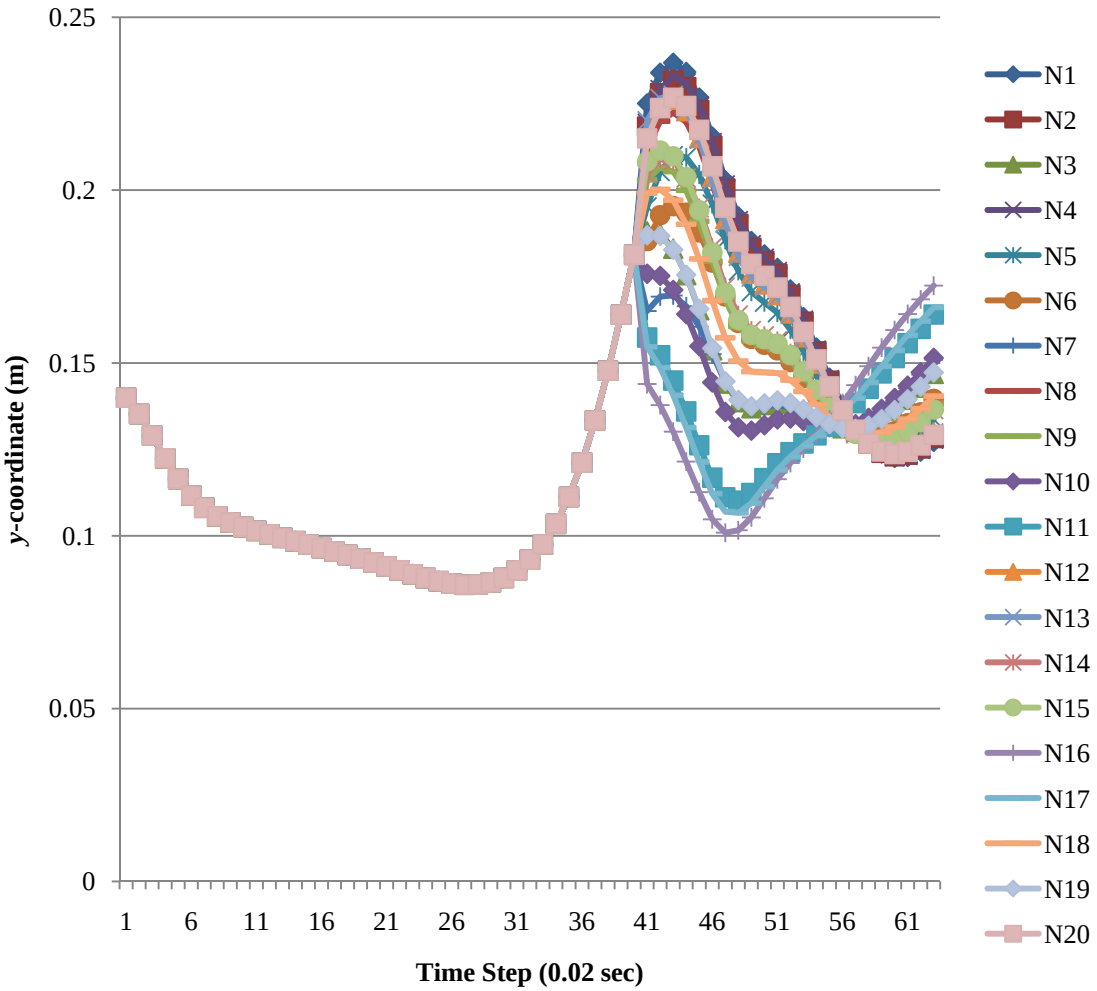

Figure 38: Ankle *y*-coordinate for BBO open loop control with 20 different initial conditions (denoted by *N*). Each time step number on the horizontal axis is a 0.02-second interval. The reference ankle coordinate is shown in Figure 37.

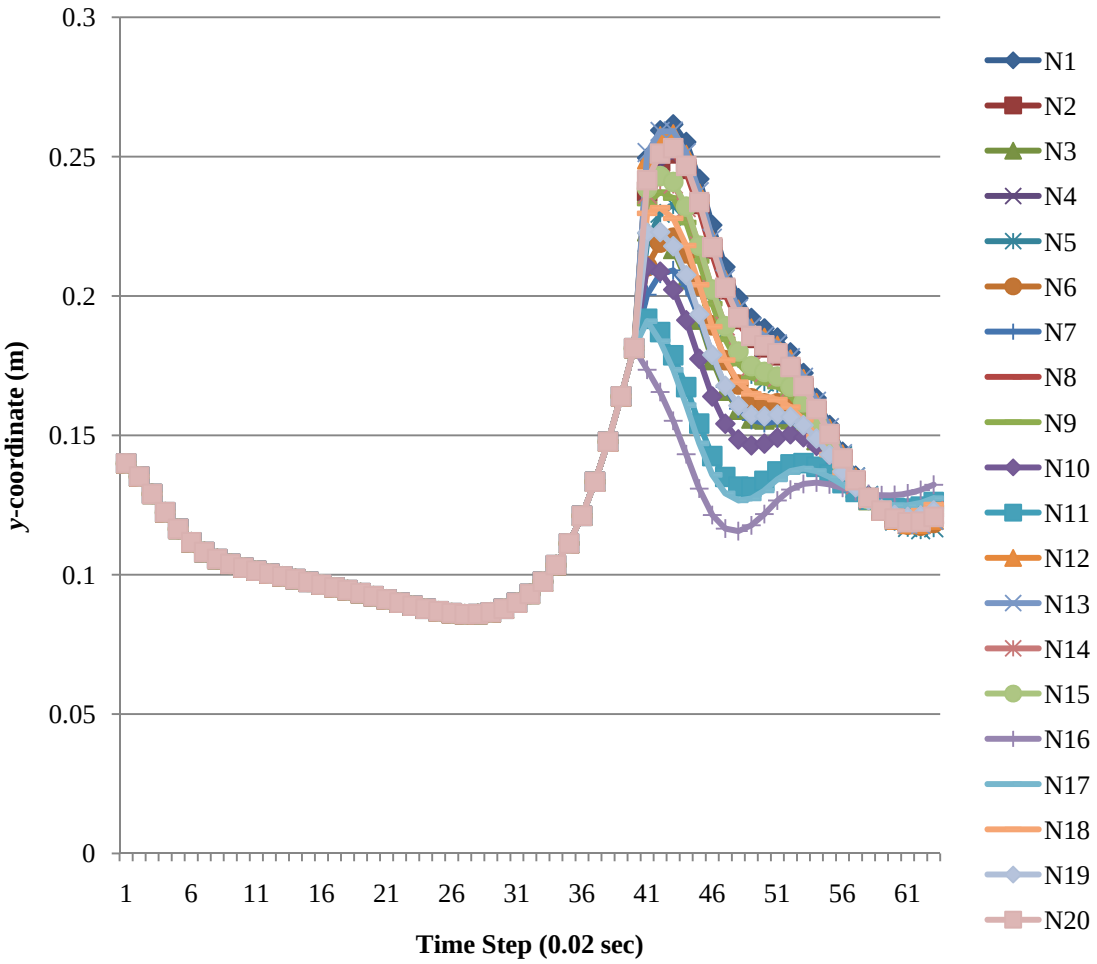

Figure 39: Ankle *y*-coordinate for ANN closed loop control with 20 different initial conditions (denoted by *N*). Each time step number on the horizontal axis is a 0.02-second interval. The reference ankle coordinate is shown in Figure 37.

We conclude from these simulations that ANNs are capable of providing robustness against variable initial conditions, feedback uncertainties, control signal noise, and measurement noise. Conditions that cause high costs have consistently lower costs when we implement the ANN closed loop control and closed loop control provides a better cost on average than open loop control alone. High costs are dangerous and could lead to stumbles and falls. It is of great value that the closed loop ANN control is able to reduce these high costs and avoid potential falls. The ANN closed loop control is also

able to produce more consistent results despite the noise. Notice the tighter grouping of the ANN costs over the BBO costs in Figure 34 through Figure 36, and the tighter grouping of ankle coordinates in Figure 39 (ANN closed loop control) than in Figure 38 (BBO open loop control). This consistency is helpful to the user for training, optimum performance, and health reasons. We can reasonably assume that with a larger set of training patters, a larger population size, and more generations of BBO training, that we could improve the performance of the ANN closed loop control. With the modification of the cost function used we can also modify exactly what type of performance the ANN will give. However, this control is severely limited by the energy characteristics of our rotary hydraulic actuator. When the rotary actuator is not able to provide and sustain the torque needed for angle tracking, it produces very high costs. This indicates we need to change the design to enable the prosthesis to respond to the extra energy requirements of the conditions we present in this thesis.

CHAPTER 6

EMBEDDED IMPLEMENTATION

This chapter discusses the future plan for microcontroller implementation of the open and closed loop controls for the prosthesis. Once we develop the controls, it is no small task to implement these controls in the real system. We have already seen in Chapter 1 that microcontrolled prostheses have the clear advantage over other forms of control. It is evident that this is the best way to implement complex control in a low power environment. We discuss preliminary hardware and software design of the microcontroller infrastructure. Our discussion includes the notion of state recognition and its effect on how we control the prosthesis. We have had little development on this subject for our prosthesis and we include this subject in the future work section.

6.1 Hardware

The intended microcontroller for the prosthetic knee is a Microchip dsPIC33FJ256GP710. This microcontroller is a 16 bit controller with DSP capabilities and we choose it because it meets all of the preliminary hardware performance requirements (such as speed, memory, and microcontroller peripheral requirements). We store the open and closed loop control mechanisms on the microcontroller. The microcontroller will generate the control signals through pulse width modulation (PWM) channels. The microcontroller sends these signals to driving amplifiers that will control the hydraulic valves. The A/D inputs on the microcontroller will read the sensor data and the information used for state recognition, closed loop control, tuning, and logging. We discuss state recognition in Section 6.2.

In addition to the basic sensory input and control output, the microcontroller will also have user input devices such as buttons or switches to control different modes of operation (discussed in Section 6.2). We will communicate pertinent or emergency information to the amputee through user output devices such as LEDs, vibrations, or audible beeps. We may implement auxiliary tuning sensors in the final design to help with initial setup and maintenance of the prosthesis. We will require communication to a PC for tuning and maintenance. We may also implement wireless communication to communicate to other user devices or a PC. Finally we need to incorporate some flash memory that will store log information or backup information. A rechargeable battery that can be charged with a household AC power outlet will power the system. A power distribution system will properly allocate the power to each power consuming device. Figure 40 shows the microcontroller hardware interfaces.

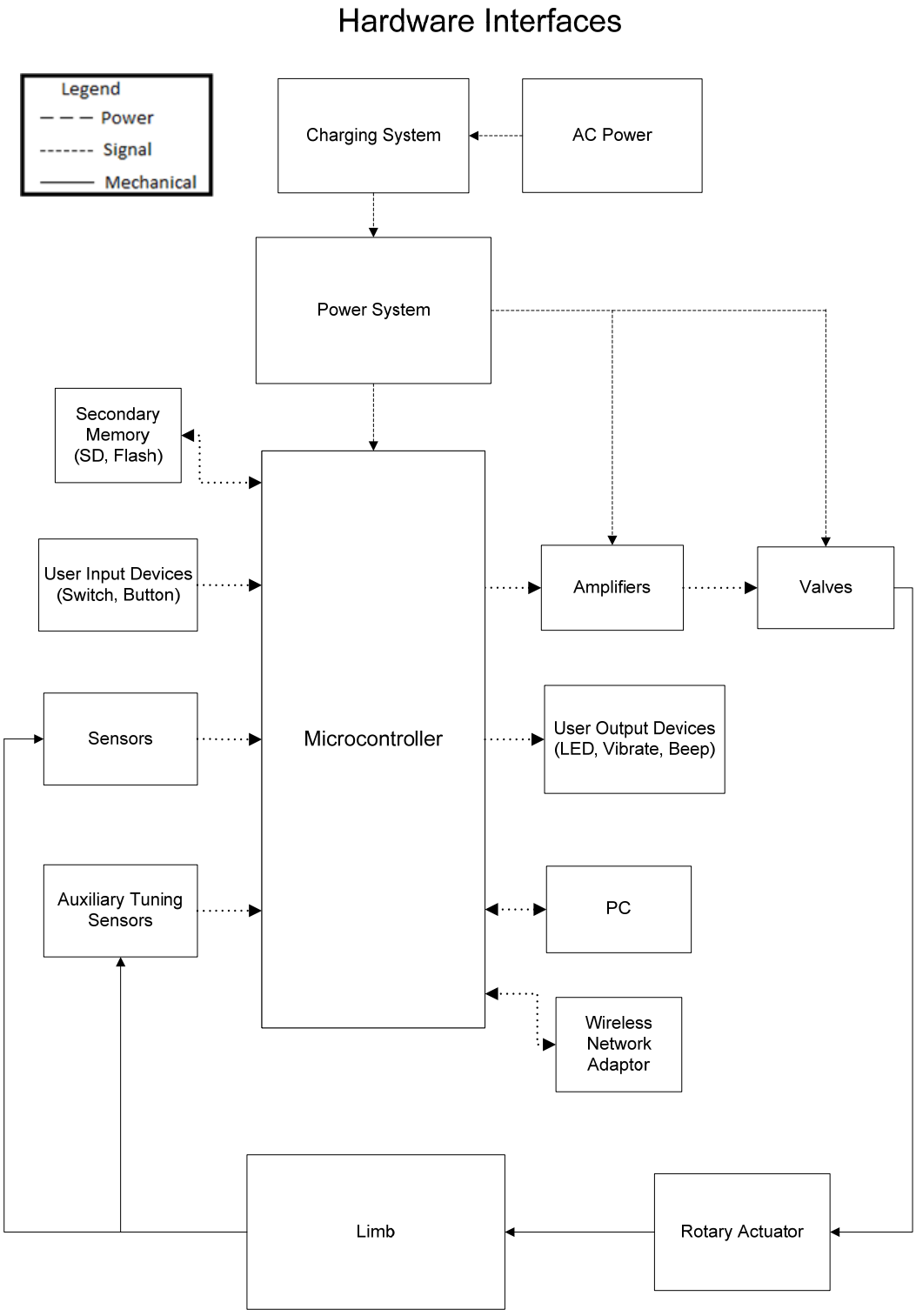

Figure 40: Microcontroller hardware interfaces.

6.2 Software Design and Interfaces

The software we implement in the microcontroller will consist of much more than the open and closed loop control that we discussed in the preceding chapters of this thesis. State recognition and its effect on the control is a significant area of research we need to address in the future. State recognition means that the microcontroller will be able to recognize what category of action the prosthesis is engaged in. These categories include but are not limited to, slow walking, normal walking, fast walking, running, stair ascent, stair descent, sitting down, standing up, and stumble detection. We need to design open and closed loop controllers for each state of operation. Also, the microcontroller needs to know where the prosthesis is in the gait cycle (i.e., stance phase or swing phase). State recognition precedes any control decision.

The microcontroller program begins with a check to see what mode the prosthesis is in. There will be several modes of operation. Normal mode is for regular prosthetic operation. Tuning mode is the same as normal mode with the exception that it has the ability for setup and maintenance of the prosthesis including data transfer and software updates. A button, switch, or by recognizing that the prosthesis is connected to a computer will trigger different modes.

In normal mode, first we read the various sensor values (exact sensors are still to be determined) and generate a particular control based on the sensor readings. We determine this control by first calculating what class of activity we are engaged in. Then we derive where we are in the gait cycle. We can then apply the predetermined open loop control for that particular activity and then closed loop control to compensate for any errors in the trajectory angles.

During any mode of operation we always check to see if we should enter fail safe mode. The software triggers this mode when errors become too large, we detect a stumble, or we encounter an unknown state. Fail safe mode will then lock the leg in a straight position to prevent collapse and injury to the user. We can exit the fail safe mode through a manual override by the user.

We will also encrypt and store log data on some form of external memory such as flash. An authorized person can then access this data and the it can be used for analytical purposes or tuning for individual use. The code will also routinely check for any reportable conditions such as low battery, full memory, and any other malfunctions. The microcontroller may report these conditions to the user via audible beeps, LEDs, or vibrations. Figure 41 shows the hardware software interfaces, with a general flow of how the software will run in the prosthesis.

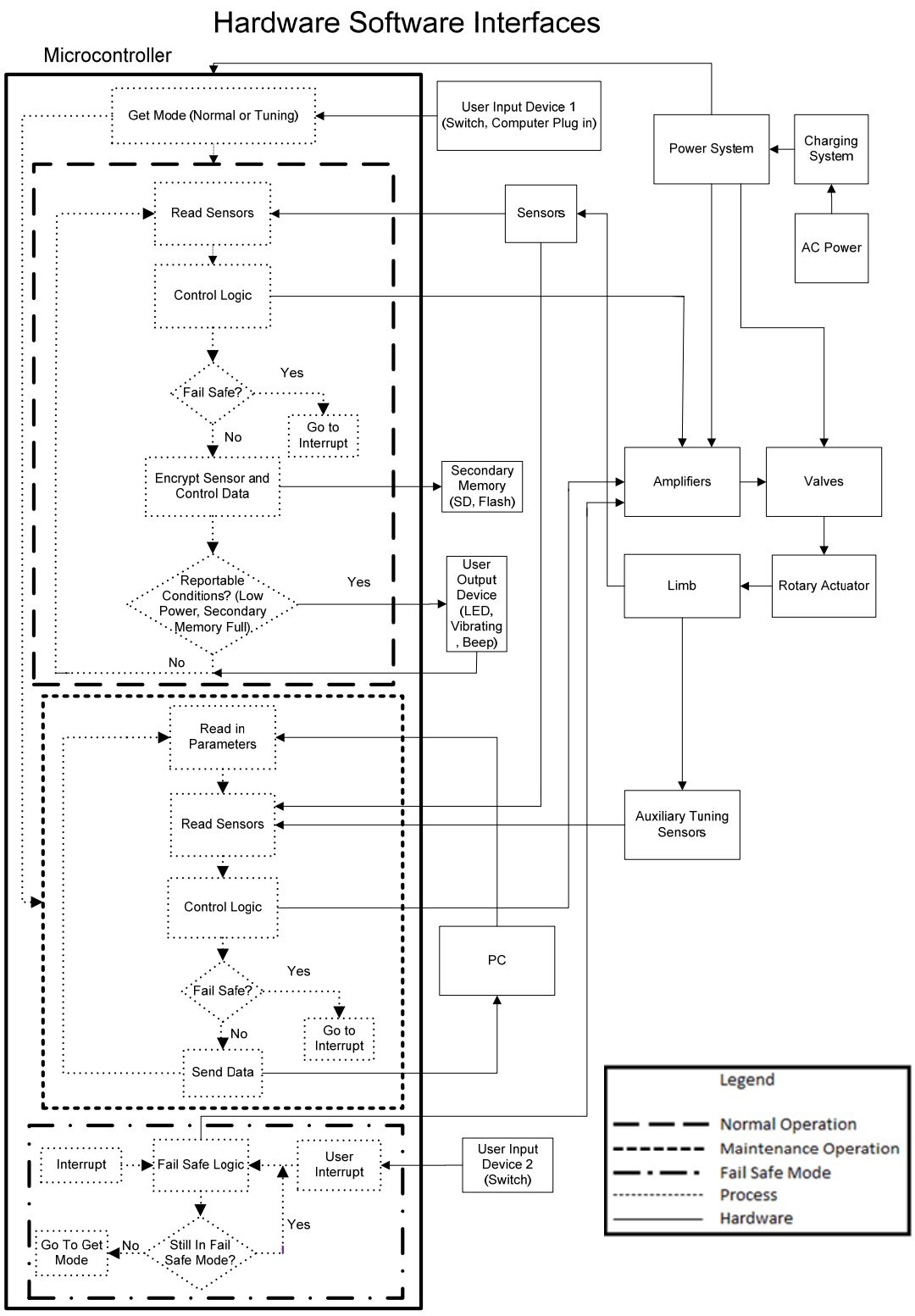

Figure 41: Hardware software interfaces.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

We have proposed a new hydraulic knee design, and have shown that BBO is able to generate near locally optimal solutions for the prosthetic knee. Gradient descent results show that the BBO solution is near the local optimum. The control solution provides good angle tracking and reasonable user hip torque, given the system limitations. These solutions provide support for continued research into semi-active microprocessor controlled prostheses. Adding the foot dynamics to our system model should improve the performance of the prosthesis and enable BBO to find a better control solution.

We have shown the potential of ANNs for closed loop control. ANNs have shown to be effective at reducing the effects of noisy measurements and controls, as well as disturbances in the user interaction with the prosthesis. We have seen that the BBO open loop control is sensitive to the initial conditions. We have shown that ANNs can reduce the negative effects of varying initial conditions. ANNs have demonstrated we can use them effectively as a closed loop controller in the prosthetic knee design.

We have given an initial design layout for the embedded system implementation of the open and closed loop controls. This implementation design also includes many of the other design parameters needed in the real system including the power system, data storage, prosthesis tuning and maintenance, and the design concept of a fail safe mode.

Future work includes improvements in the mechanical design of the prosthetic knee, and actual embedded system implementation of the valve controls. We can accomplish this through a hardware in the loop simulation for proof of concept and then adapt it to actual hardware. There are also many other issues we need to address before prosthetic implementation of the control, such as sensor selection [33] and gait phase recognition [34]. Also, a commercial prosthesis needs to function correctly in various operating modes, such as running, going up and down stairs, standing up, and sitting down. A commercial prosthesis also needs to implement user intent recognition [38] and stumble detection and recovery [39], and it needs to have a reliable and long-lasting power source [40]. The results that we present are for evolutionary open loop control, artificial neural network closed loop control, and embedded implementation of a novel hydraulic prosthesis design, and thus form an integral component of a large, multifaceted, multidisciplinary research effort.

REFERENCES

[1] Kulkarni J, Gaine W, Buckley J, Rankine J, Adams J, “Chronic low back pain in traumatic lower limb amputees,” *Clinical Rehabilitation*, vol. 19, 2005, pp. 81–86.

[2] Gailey R, Allen K, Castles J, Kucharik J, Roeder M, “Review of secondary physical conditions associated with lower limb amputation and long-term prosthesis use,” *Journal of Rehabilitation Research Development*, vol. 45, 2008, pp. 15–29.

[3] Modan M, Peles E, Halkin H, Nitzan H, Azaria M, Gitel S, Dolfin D, Modan B, “Increased cardiovascular disease mortality rates in traumatic lower limb amputees,” *American Journal of Cardiology*, vol. 82, 1998, pp. 1242–1247.

[4] Seymour R, Engbretson B, Kott K, Ordway N, Brooks G, Crannell J, Hickernell E, Wheeler K, “Comparison between the C-leg microprocessor-controlled prosthetic knee and non-microprocessor control prosthetic knees: a preliminary study of energy expenditure, obstacle course performance, and quality of life survey,” *Prosthetics and Orthotics International*, vol. 31, 2007, pp. 51 – 61.

[5] Seroussi R, Gitter A, Czerniecki J, Weaver K, “Mechanical work adaptations of above-knee amputee ambulation,” *Archive of Physical Medicine Rehabilitation*, vol. 77, 1996, pp. 1209–1214.

[6] Johansson J, Sherrill D,Riley P, Bonato P, Herr H, “A clinical comparison of variable-damping and mechanically passive prosthetic knee devices,” *American Journal of Physical Medicine and Rehabilitation*, vol. 84, 2005, pp. 563–575.

[7] Chin T, Machida K, Sawamura S, Shiba R, Oyabu H, Nagakura Y, Takase I, Nakagawa A, “Comparison of different microprocessor controlled knee joints on the

energy consumption during walking in trans-femoral amputees: Intelligent knee prosthesis (IP) versus C-leg," *Prosthetics and Orthodontics International*, vol. 30, 2006, pp. 73–80.

[8] Bellmann M, Schmalz T, Blumentritt S, "Comparative biomechanical analysis of current microprocessor-controlled prosthetic knee joints," *Archive of Physical Medicine and Rehabilitation*, vol. 91, 2010, pp. 644–652.

[9] Segal A, Orendurff M, Klute G, McDowell M, Pecoraro J, Shofer J, Czerniecki J, "Kinematic and kinetic comparisons of transfemoral amputee gait using C-Leg and Mauch SNS prosthetic knees," *Journal of Rehabilitation and Research Development*, Vol. 43, 2006, pp. 857–870.

[10] Simon D, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, 2008, pp. 702-713.

[11] Abo-Hammou Z S, Yusuf M, Mirza N M, Mirza S M, Arif M, Khurshid J, "Numerical solution of second-order, two-point boundary value problems using continuous genetic algorithms," *International Journal for Numerical Methods in Engineering*, vol. 6, 2004, pp.1219-1242.

[12] Crispin Y, "An evolutionary approach to nonlinear discrete-time optimal control with terminal constraints, in Informatics in control, automation and robotics I (Braz J, Vieira A, Encarnacao B, Editors)," *Springer Netherlands*, 2006, pp. 89-97.

[13] Lee S, Fink W, von Allmen P, Petropoulos A, Russell R, Terrile R, "Evolutionary computing for low-thrust navigation," *American Institute of Aeronautics and Astronautics Space Conference*, 2005.

[14] Yang Z, Fang J, Qi Z, “Flight midcourse guidance control based on genetic algorithm,” *Conference on Genetic and Evolutionary Computation*, 2005, Washington, DC, USA, pp. 1501 – 1506.

[15] Pandy M, Anderson F, Hull D, “A parameter optimization approach for the optimal control of large-scale musculoskeletal systems,” *Transactions of the American Society of Mechanical Engineers*, vol. 114, 1992, 450 – 460.

[16] Yen V, Nagurka M, “Fourier-based optimal control approach for structural systems,” *American Institute of Aeronautics and Astronautics*, vol. 13, 1990, 2082 – 2087.

[17] Yokose Y, Izumi T, “Non-linear two-point boundary value problem obtaining the expansion coefficients by the dynamic GA and its application,” *Transactions on Electronics, Information and Systems*, vol. 124, 2005, pp. 2179 – 2186.

[18] Bergantz D, Barad H, “Neural network control of cybernetic limb prostheses,” *IEEE International Conference on Engineering in Medicine and Biology Society*, vol. 3, November 4 – 7, 1988, New Orleans, LA, USA, pp. 1486 – 1487.

[19] Albus J, “A new approach to manipulator control: the cerebellar model articulation controller,” *Transactions of the American Society of Mechanical Engineers, Journal of Dynamic Systems, Measurement, and Control*, September 1975, pp. 220-227.

[20] Elsley R, “Adaptive control of prosthetic limbs using neural networks,” *International Joint Conference on Neural Networks*, vol. 2, June 17 – 21, 1990, San Diego, California, USA, pp. 771 – 776.

[21] Kalanovic V, Popovic D, Skaug N, “Feedback error learning neural network for trans-femoral prosthesis,” *IEEE Transactions on Rehabilitation Engineering*, vol. 8, 2000, pp.71 – 80.

[22] Deluzio K, “Gait analysis,” May 2, 2011. http://me.queensu.ca/people/deluzio/GaitAnalysis.php.

[23] Mitiguy P, Reckdahl K, “Autolev tutorial,” *OnLine Dynamics, Inc.* Sep. 29, 2003. http://www.mae.ufl.edu/~fregly/egm4590/AutolevTutorial.pdf.

[24] van den Bogert A, Samorezov S, Davis B, Smith W, “Modeling and optimal control of an energy-storing prosthetic knee,” *Submitted to the journal of biomedical engineering for publication*.

[25] Ovreiu M, Simon D, “Biogeography-based optimization of neuro-fuzzy system parameters for diagnosis of cardiac disease,” *Genetic and Evolutionary Computation Conference*, July 7–11, 2010, Portland, Oregon, USA, pp. 1235 – 1242.

[26] Rarick R, Simon D, Villaseca E, Vyakaranam B, “Biogeography-based optimization and the solution of the power flow problem,” *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, December 4, 2009, San Antonio, Texas, USA, pp. 1003-1008.

[27] Kundra H, Kaur A, Panchal V, “An integrated approach to biogeography based optimization with case based reasoning for retrieving groundwater possibility,” *Proceedings of the 8th Annual Asian Conference and Exhibition on Geospatial Information, Technology and Applications*, August 2009, Singapore.

[28] Panchal V, Singh P, Kaur N, Kundra H, “Biogeography based satellite image classification,” *International Journal of Computer Science and Information Security*. vol. 6, 2009, pp. 269-274.

[29] Smith R, Minton R, “Calculus concepts and connections,” *McGraw-Hill*, 2006, pp. 655-667.

[30] Kirk D, “Optimal control theory: an introduction,” *Prentice-Hall Inc*, 1970.

[31] Jang J, Sun C, Mizutani E, “Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence,” *Prentice Hall Inc*, 1997

[32] Dosen S, Popovic D, “Accelerometers and force sensing resistors for optimal control of walking of a hemiplegic,” *IEEE Transactions on Biomedical Engineering*, vol. 55, 2008, pp. 1973 – 1984.

[33] Williamson R, Andrews B, “Detecting absolute human knee angle and angular velocity using accelerometers and rate gyroscopes,” *Medical & Biological Engineering & Computing*, vol. 39, 2001, pp. 294 – 302.

[34] McDonald C, Smith D, Brower R, Ceberio M, Sarkodie-Gyan T, “Determination of human gait phase using fuzzy inference,” *IEEE International Conference on Rehabilitation Robotics*, June 12-15 2007, Noordwijk, The Netherlands, pp. 661 – 665.

[35] Gu J, Ding X, Wang S, Wu Y, “Action and gait recognition from recovered 3-D human joints,” *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 40, 2010, pp. 1021 – 1033.

[36] Pappas I, Popovic M, Keller T, Dietz V, Morari M, “A reliable gait phase detection system” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 9, 2001, pp. 113 – 125.

[37] Zhang J, Pu J, Chen C, Fleischer R, “Low-resolution gait recognition,” *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 40, 2010, pp. 986 – 996.

[38] Varol H, Sup F, Goldfarb M, “Multiclass real-time intent recognition of a powered lower limb prosthesis,” *IEEE Transactions on Biomedical Engineering*, vol. 57, 2010, pp. 542 – 551.

[39] Zahedi S, Sykes A, Lang S, Cullington I, “Adaptive prosthesis – a new concept in prosthetic knee control,” *Cambridge University Press, Robotica*, vol. 23, 2005, pp. 337-244.

[40] Dellon B, Matsuoka Y, “Prosthetics, exoskeletons, and rehabilitation” *IEEE Robotics & Automation Magazine*, vol. 14, 2007, pp. 30 – 34.

APPENDIX

Here we show the prosthesis dynamic equations in Matlab format, along with the variable definitions and units.

Table of variables:

Variable	Definition	Units
phi_1	Thigh angle	Radians
phi_1p	Thigh angle velocity	Radians/Second
phi_1pp	Thigh angle acceleration	Radians/Second^2
phi_k	Knee angle	Radians
phi_kp	knee angle velocity	Radians/Second
phi_kpp	Knee angle acceleration	Radians/Second^2
M_h	Hip moment (torque)	Newton*Meters
M_k	Knee moment (torque)	Newton*Meters
Fy_h	Hip *y*-axis force	Newtons
Fx_h	Hip *x*-axis force	Newtons
y_hpp	Hip *y*-axis acceleration	Meters/Second^2
x_hpp	Hip *x*-axis acceleration	Meters/Second^2

Stance phase hip angle dynamics:

phi_1pp = ((I_shank + m_shank * (CM_shank - L_shank)^2 + L_shank * m_trunk * (L_shank + L_thigh * cos(phi_k)) + L_shank * m_thigh * (L_shank - cos(phi_k) * (CM_thigh - L_thigh))) * (M_k + g * L_shank * m_thigh * sin(phi_1 + phi_k) - Fx_h * L_shank * cos(phi_1 + phi_k) - g * m_shank * sin(phi_1 + phi_k) * (CM_shank - L_shank) - L_shank * sin(phi_1 + phi_k) * (Fy_h - g * m_trunk) - L_shank * sin(phi_k) * (L_thigh * m_trunk - m_thigh * (CM_thigh - L_thigh)) * phi_1p^2) - (I_shank + m_thigh * L_shank^2 + m_trunk * L_shank^2 + m_shank * (CM_shank - L_shank)^2) * (M_h + Fx_h * L_shank * sin(phi_1) * sin(phi_k) + g * m_thigh * (L_shank * sin(phi_k) * cos(phi_1) - sin(phi_1) * (CM_thigh - L_thigh - L_shank * cos(phi_k))) - Fx_h * cos(phi_1) * (L_thigh + L_shank * cos(phi_k)) - g * m_shank * sin(phi_1+phi_k) * (CM_shank - L_shank) - L_shank * sin(phi_k) * cos(phi_1) * (Fy_h - g * m_trunk) - sin(phi_1) * (Fy_h - g * m_trunk) * (L_thigh + L_shank * cos(phi_k)) - L_shank * sin(phi_k) * (L_thigh * m_trunk - m_thigh * (CM_thigh - L_thigh)) * (phi_1p^2 - (phi_1p + phi_kp)^2))) / ((I_shank + m_shank * (CM_shank - L_shank)^2 + L_shank * m_trunk * (L_shank + L_thigh * cos(phi_k)) + L_shank * m_thigh * (L_shank - cos(phi_k) * (CM_thigh - L_thigh)))^2 - (I_shank + m_thigh * L_shank^2 + m_trunk *

L_shank^2 + m_shank * (CM_shank - L_shank)^2) * (I_shank + I_thigh + m_shank * (CM_shank - L_shank)^2 + m_trunk * (L_shank^2 * sin(phi_k)^2 + (L_thigh + L_shank * cos(phi_k))^2) + m_thigh * (L_shank^2 * sin(phi_k)^2 + (CM_thigh - L_thigh - L_shank * cos(phi_k))^2)));

Stance phase knee angle dynamics:

phi_kpp = - ((I_shank + I_thigh + m_shank * (CM_shank - L_shank)^2 + m_trunk * (L_shank^2 * sin(phi_k)^2 + (L_thigh + L_shank * cos(phi_k))^2) + m_thigh * (L_shank^2 * sin(phi_k)^2 + (CM_thigh - L_thigh - L_shank * cos(phi_k))^2)) * (M_k + g * L_shank * m_thigh * sin(phi_1 + phi_k) - Fx_h * L_shank * cos(phi_1 + phi_k) – g * m_shank * sin(phi_1 + phi_k) * (CM_shank - L_shank) - L_shank * sin(phi_1 + phi_k) * (Fy_h - g * m_trunk) - L_shank * sin(phi_k) * (L_thigh * m_trunk - m_thigh * (CM_thigh - L_thigh)) * phi_1p^2) - (I_shank + m_shank * (CM_shank - L_shank)^2 + L_shank * m_trunk * (L_shank + L_thigh * cos(phi_k)) + L_shank * m_thigh * (L_shank - cos(phi_k) * (CM_thigh - L_thigh))) * (M_h + Fx_h * L_shank * sin(phi_1) * sin(phi_k) + g * m_thigh * (L_shank * sin(phi_k) * cos(phi_1) - sin(phi_1) * (CM_thigh - L_thigh - L_shank * cos(phi_k))) - Fx_h * cos(phi_1) * (L_thigh + L_shank * cos(phi_k)) - g * m_shank * sin(phi_1 + phi_k) * (CM_shank - L_shank) - L_shank * sin(phi_k) * cos(phi_1) * (Fy_h - g * m_trunk) - sin(phi_1) * (Fy_h – g * m_trunk) * (L_thigh + L_shank * cos(phi_k)) - L_shank * sin(phi_k) * (L_thigh * m_trunk - m_thigh * (CM_thigh - L_thigh)) * (phi_1p^2 - (phi_1p + phi_kp)^2))) / ((I_shank + m_shank * (CM_shank - L_shank)^2 + L_shank * m_trunk * (L_shank + L_thigh * cos(phi_k)) + L_shank * m_thigh * (L_shank - cos(phi_k) * (CM_thigh - L_thigh)))^2 - (I_shank + m_thigh * L_shank^2 + m_trunk * L_shank^2 + m_shank * (CM_shank - L_shank)^2) * (I_shank + I_thigh + m_shank * (CM_shank - L_shank)^2 + m_trunk * (L_shank^2 * sin(phi_k)^2 + (L_thigh + L_shank * cos(phi_k))^2) + m_thigh * (L_shank^2 * sin(phi_k)^2 + (CM_thigh - L_thigh - L_shank * cos(phi_k))^2)));

Swing phase knee angle dynamics:

phi_kpp = (M_k - CM_shank * g * m_shank * sin(phi_1 + phi_k) - CM_shank * L_thigh * m_shank * sin(phi_k) * phi_1p^2 - CM_shank * m_shank * sin(phi_1 + phi_k) * y_hpp - CM_shank * m_shank * cos(phi_1 + phi_k) * x_hpp - (I_shank + CM_shank * m_shank * (CM_shank + L_thigh * cos(phi_k))) * phi_1pp) / (I_shank + m_shank * CM_shank^2);