INVESTIGATING DIFFERENCES BETWEEN OPTIMAL BIPEDAL WALKING GAITS GENERATED USING DIFFERENT JACOBIAN CALCULATION AND DIRECT COLLOCATION APPROACHES

A Thesis

by

VERONICA ANN KNISLEY

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Pilwon Hur
Committee Members,	Swaroop Darbha
	Raktim Bhattacharya
Head of Department,	Andreas Polycarpou

May 2020

Major Subject: Mechanical Engineering

ABSTRACT

There are several applications in the area of bipedal robotics, many of which require one to find some form of optimal trajectory. In practice, it is good to convert such an optimal control problem into a discretized nonlinear program (NLP) using a method like direct collocation. There are many different elements which are required to complete such an optimization. Although previous studies have compared different ways of structuring and computing such elements in other fields, work has not been done in bipedal robotics specifically to determine which of these methods produce the ideal combination of computational efficiency and consistency with a desired baseline.

This study sought to compare such conditions for two different walking models. For the two degree of freedom compass walker, Hermite-Simpson and trapezoidal collocation methods were compared, and numerical and symbolic Jacobian calculation were examined. For the five-link kneed biped with a torso, the same Jacobian calculation methods were reviewed, along with symbolic and numerical methods of calculating the walker's joint accelerations and the inclusion or exclusion of these accelerations from the set of decision variables. This was accomplished by developing NLPs for each scenario and running repeated optimizations of each type in MATLAB with randomized initial guesses.

For the compass walker, the combination of numerical differentiation and trapezoidal collocation was determined to be ideal, due to its relatively quick computation time and ease of implementation. For the five-link biped, including joint accelerations in the decision variables seemed to make the optimization more robust to the initial guess and faster on a by-iteration basis. Calculating accelerations symbolically was quicker than doing so numerically, and numerical differentiation was recommended due to its scalability to higher dimensional systems. Overall, these results will improve the efficiency and result of bipedal walking trajectory optimization.

DEDICATION

To my amazing parents–along with my Granny, Grandpa, and Nana–who have supported me in all my endeavors.

ACKNOWLEDGMENTS

I would like to thank my research advisor and committee chair, Dr. Pilwon Hur, for his guidance and encouragement over the course of this project. I also thank Dr. Swaroop Darbha and Dr. Raktim Bhattacharya, my committee members, for their advice, as well as Dr. Sivakumar Rathinam for his time and valuable input.

Additionally, I would like to thank Dr. Kenneth Chao, who acted as a mentor to me before his graduation in May 2019; he was always willing to answer my questions and recommend sources for me to study. Lastly, I owe gratitude to the other members of HUR Group, as they were always willing to help me.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Pilwon Hur (advisor) and Professor Swaroop Darbha of the Department of Mechanical Engineering and Professor Raktim Bhattacharya of the Department of Aerospace Engineering.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a teaching assistantship through the Texas A&M University College of Engineering.

NOMENCLATURE

DOF	Degrees of Freedom
DOA	Degrees of Actuation
ND	Numerical Differentiation
SD	Symbolic Differentiation
H-S	Hermite-Simpson Collocation
TPZD	Trapezoidal Collocation
NLP	Nonlinear Program
DV	Decision Variable
BG	Baseline Gait
IG	Initial Guess
ANOVA	Analysis of Variance

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

FIGURE		Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Research and Applications of Walking Models

Bipedal walking research has taken many forms over the last several decades. In fact, the process of walking can be viewed from either a biomechanical or robotic perspective. In the biomechanical field, research can include the study of muscle behavior during walking [**?**, 1, 2]. In the area of robotics, the models can be far simpler and only give rough approximations of human walking. Some models studied include upper limbs [3] or upper bodies [4, 5], while others do not even include feet or knees [6, 7]. Regardless of their complexity, however, these walkers can yield two classes of information which are useful to robotics researchers. First, they provide valuable information on the underlying dynamical features of human gait. Secondly, they give us a canvas on which to experiment with passive or active [8] methods of generating human-like walking patterns. It is fairly easy to see the value of walking research from a biomechanics perspective; however, why is it valuable in robotics? What applications does it hold?

One key application for the development of walking robots, whether bipedal or otherwise, is in the traversing of rough terrain. Legged mechanisms are more effective at going over uneven surfaces than objects with wheels or treads [9]. The robot RHex is one six-legged robot which was designed with such ideas in mind [10]. However, research has also been done in this area with bipedal models specifically; some researchers, such as Chao [11] and Dai and Tedrake [12], have worked to develop cost functions for trajectory optimization which can better handle uncertainties in ground heights. The real world is not flat, and there are many areas in which having robots which can traverse rough terrain would be useful.

Another critical application of bipedal robotics is in the development of assistive devices. Walking dynamics and control play a key role in the creation of powered, lower-limb prostheses [13], which can allow amputees to walk in a relatively more natural way again. Additionally, powered exoskeletons can serve multiple purposes. Some can be used in the rehabilitation process

[14], while others otherwise improve or restore the ability of people to walk. All of these devices can improve the mobility, independence, or confidence of their users. From the previous examples, it is clear that there is a need for studying bipedal robotics. Many of these applications require us to generate walking gaits; that being said, not just any gait will be satisfactory.

1.2 Desirable Gait Characteristics

There are several qualities and characteristics which should be kept in mind when generating a walking gait. For example, it is important to have smooth control signals, as well as smooth trajectories at the position, velocity, and acceleration levels. It is also important for the gait to be accurate, if an expected solution is already known, and it may be necessary for the determination of the gait to be repeatable. Another consideration is that it might be necessary for a gait to be computed quickly. Some walking applications even strive to achieve real-time trajectory generation [13]. In autonomous robotic walking, which admittedly is beyond the scope of this project, rapid trajectory generation is shown to be important for working with sensors and avoiding obstacles [15].

In addition to these general requirements, there may be additional, case-by-case desirable traits. These can be captured through the model of a trajectory optimization problem. Such a problem formulation allows one to determine an optimal control signal $u(t)$ and state trajectory $x(t)$ of a system by minimizing or maximizing an objective value, subject to constraints. One common objective in bipedal walking is to minimize a quantity called cost of transport (COT), perhaps in combination with other terms [16]. This is a ratio of the energy input to a system to its weight and distance traveled. Another common cost function, which is employed in this study, is the integral of the square of the input torques over the course of a step [5]; minimizing this objective will reduce the control effort required. Constraints in the optimization can account for other factors such as joint angle limitations or bounds on the torques which can be applied to the system.

1.3 From the Optimal Control Problem to Direct Collocation

Trajectory optimization problems are just one class of optimal control problem. There are methods in place to solve such optimal control problems, such as Pontryagin's Maximum Principle, which states:

$$\frac{dx}{dt} = \frac{\partial H}{\partial y} \tag{1.1}$$

$$\frac{dy}{dt} = -\frac{\partial H}{\partial x} \tag{1.2}$$

$$H = \max_{u \in \Omega} H(u) \tag{1.3}$$

where y is given as the partial derivative of the Lagrangian L–or the integrand of the cost function–with respect to $\dot{x}(t)$, and H, the Hamiltonian, is given to be:

$$H = y^T \frac{dx}{dt} - L \tag{1.4}$$

The above equations are reported in [17]. However, due to the complexity of walking dynamics, along with the multitude of additional constraints required for walking optimizations, such analytical, continuous-time solutions can be difficult to determine. Fortunately, there are multiple transcription methods in place which use computational approaches to solve these problems. These methods convert a continuous-time trajectory optimization problem into a discretized nonlinear program.

One class of methods is the shooting methods. With these methods, either the entire trajectory (in single shooting) or segments of the trajectory (in multiple shooting) are forward simulated based on an initial guess. Through repeated iterations, an optimal set of decision variables can be determined [17, 18]. The extremely complex dynamics of bipeds, particularly those of more complicated models like a walker with five links, make this method unsuitable for such problems.

One valuable technique is to discretize the continuous trajectory with respect to time and find control inputs and states along the trajectory only at specific points. After the optimization is complete, spline interpolation can be used to generate continuous control signals and joint trajectories.

This technique is called direct collocation [18], and it is illustrated in Figure 1.1. In order to effectively use this method, it is necessary to satisfy the system's dynamics at each collocation point.

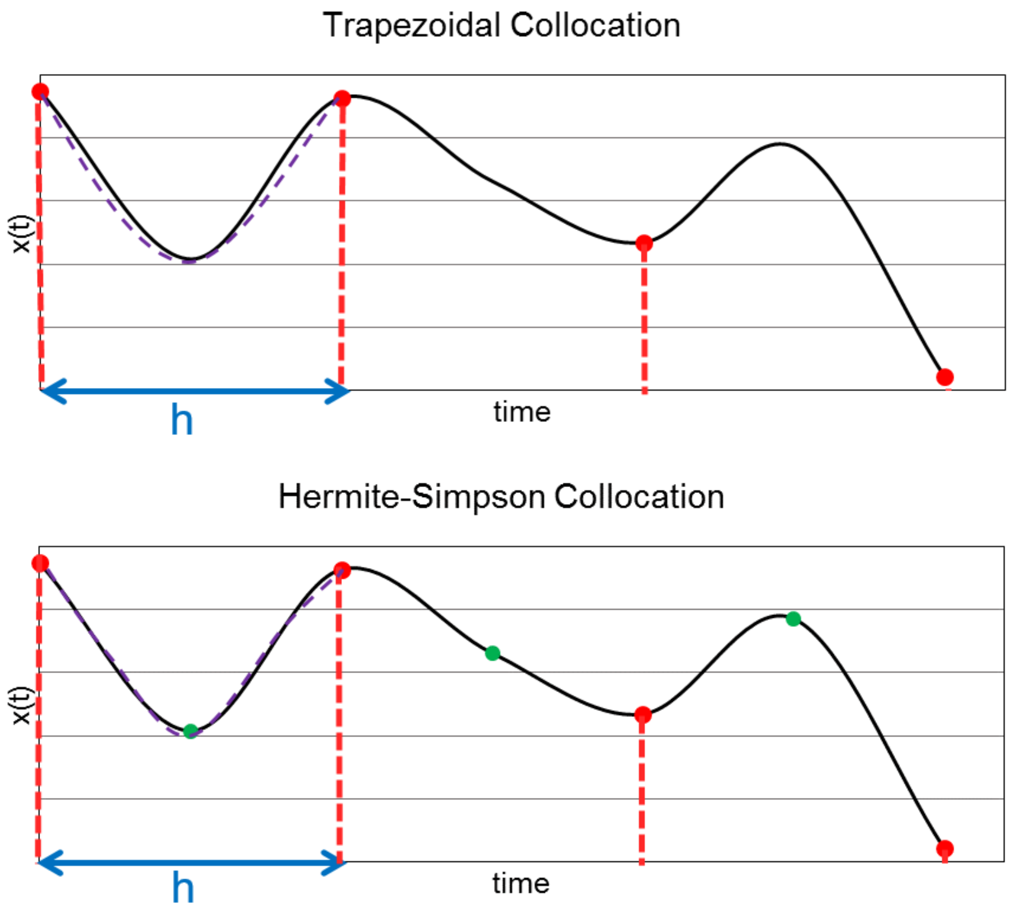


Figure 1.1: Visualization of direct collocation.

There are several ways in which the continuous trajectory can be discretized. The top half of the figure represents trapezoidal collocation (TPZD). Using this technique, the states are approximated by quadratic splines, while the dynamics and control inputs are approximated as linear splines. Only the states and control values at the collocation points, or end points of the trajectory segments, are calculated in this case. The bottom half of the figure represents Hermite-Simpson collocation (H-S). With this method, all key results of the optimization are approximated with one higher order, meaning that the states are considered to be cubic while the dynamics and control input are quadratic [5]. With this method the states and control input at the midpoint of each trajectory segment are also required. Each collocation method uses different equations to define the constraints; these are detailed later in this document. It should be noted that in my optimizations, all points are taken to be equally-spaced, although this is not required in general.

Direct collocation is an increasingly popular trajectory optimization technique in the world of bipedal robotics. Five-link bipedal walking was one of the example cases used in Matthew Kelly's instructional document on direct collocation [5]. Chao [11] and Dai and Tedrake [12] used direct collocation as the framework for their robust trajectory optimization formulations. Chao and Hur [16] also used direct collocation to compare the gaits generated by walking models allow slipping, not allowing slipping, and including springs at the model's toe joints. Hereid et al. used direct collocation, along with the the system's hybrid zero dynamics (HZD), to generate gaits for the robot DURUS [19]. Overall, direct collocation is becoming a valuable technique in this field.

In any type of numerical or symbolic computation, it is useful to consider the inversion of matrices. In particular, as will be seen later in this document, finding the accelerations at each of the model's joints requires finding the solution to a set of linear equations. The inversion of matrices is a very costly calculation. There are different ways to avoid such calculations. For example, Chao and Hur included joint accelerations in their decision variables [16]. This study will also consider the differences between the one-time symbolic derivation of the five-link model's joint accelerations and their repeated numerical calculation; each of these methods involves solving a system of linear equations, either numerically or symbolically. The specific methods used in either case do not explicitly invert the models' inertia matrices, but they may still be costly.

Many different types of computer software can be used to solve direct collocation problems. In this study, the open-source mexIPOPT package [20], or the MATLAB version of IPOPT, was used. This solver uses the interior point algorithm to converge to an optimal solution. Two of the inputs which IPOPT requires are the gradient of the cost function and the Jacobian of the constraints. The

Jacobian is particularly important, as it is employed in both solving the barrier problems within the algorithm and correcting the search direction [21]. The derivatives therein can be calculated in multiple ways. In this study, numerical and symbolic differentiation are the focus. More specifically, the central difference method for numerical differentiation and analytical expressions calculated with MATLAB's symbolic toolbox were compared. Both collocation methods and differentiation methods have been compared in previous studies.

1.3.1 Previous Studies Comparing Collocation or Differentiation Methods

Although there have not been similar studies performed on bipedal walkers like those in this study, other fields have compared the performance of different collocation methods during the optimization process, as well as differentiation methods in a variety of settings. One example of the former was a study performed by Nie and Kerrigan [22], whose application was in the aerospace field. They used two separate discretization methods, H-S and Legendre Gauss-Radau (LGR) in the formation of certain constraints called rate constraints. Although the methods were not really being compared against each other, the former method was shown to have a slightly faster computational time. Another study, this time in mobile robotics, was done by Pardo et al. [23]. They optimized the trajectory of a rolling, "ball-balancing" robot called Rezero with both H-S and TPZD. Although there were not statistically significant differences between the resulting computational times when IPOPT was used, the accuracy of H-S was significantly higher.

Using different differentiation methods, whether in an optimization framework or not, can also yield differences in both accuracy and computational time. Three main differentiation methods which have been examined in the literature are the aforementioned ND and SD, along with a method called automatic differentiation (AD). The latter, which is not examined in this study but would be valuable to use in future work, has the accuracy of SD without the need to store large symbolic expressions [24].

Durrbaum et al. used AD and SD to calculate the Jacobian for the inverse dynamics of both planar and spatial "parallel robots" [25]. They found that while the actual computational time was lower for the latter, the former would work better with increasingly complex systems. Giftthaler et al. looked at all three differentiation methods in multiple robotics contexts [26]. When calculating the derivatives of the forward dynamics of the quadrupedal robot HyQ, ND was the least accurate method, and AD was faster than ND. AD was also faster, per iteration, when it was used with Sequential Linear Quadratic optimal control on this same robot and when using the multiple shooting method to determine the trajectory of a robot arm. Lastly, Falisse et al. found that their implementation of AD was faster than both ND and a previous implementation of AD in a direct collocation framework when it was used for controlling pendulums and musculoskeletal walking models [2].

For either of these types of modifications, it may turn out that one "setting" is not optimal in all aspects; for example, TPZD could be faster but less accurate. In cases like this, it is necessary to consider tradeoffs between characteristics like efficiency and accuracy. This is a need which has been previously expressed in the literature [27, 28]. At least one study has looked into these tradeoffs in more detail, albeit with respect to the number of nodes. Lin and Pandy found that increases in accuracy due to an increased number of nodes did not offset the increasingly large computational time [1].

1.4 Research Question and Outline of Discussion

The above studies illuminate a few gaps in the literature. First, although robotics studies have been performed on factors such as differentiation methods and collocation methods, such studies have not been for robotic bipedal walkers. Similarly, although some studies have looked into bipedal walking, these have been biomechanical studies which include muscle behavior, which is not a major concern for the most common robotic walkers. In either of the aforementioned cases, it is also notable that studies typically did not look at both collocation and differentiation methods together. There is a need for a contribution examining such factors in a bipedal robotics context.

The research question addressed is this study is the following: "How do Jacobian differentiation methods, collocation methods, and decision variable setups affect the efficiency and accuracy of walking gait calculation, and which combination is the best to use?" This was explored by finding optimal walking gaits for two different point-foot walkers, the compass walker and the five-link walker, using various combinations of these settings. The results were then compared in terms of CPU time, deviation from an expected gait, and average CPU time per iteration.

The study is detailed in the remainder of this document. Chapter 2 describes the two walking models and the constraints and objective function needed to set up gait-related nonlinear programs. Chapter 3 explains how this was coded in MATLAB. Chapter 4 gives details of the simulation-based experimental procedure and data analyses performed. Chapter 5 gives experimental results and discussion, and Chapter 6 gives conclusions and future work.

2. DEVELOPMENT OF WALKING NONLINEAR PROGRAMS

2.1 Compass Walker

The compass walker is the simplest walking model. It has two degrees of freedom (DOF) and neither knees nor feet. Its legs are also treated as point masses. This version of the model only has one degree of action (DOA) at the hip, so the model is underactuated. At any given point in time, the tip of the so-called stance leg acts as a pivot point, and the swing leg comes forward. Figure 2.1 gives one representation of the model. It is important to note that with this model, there is no way to actively prevent the swing leg from scraping along, or even underneath, the ground. This phenomenon is called foot scuffing [29].

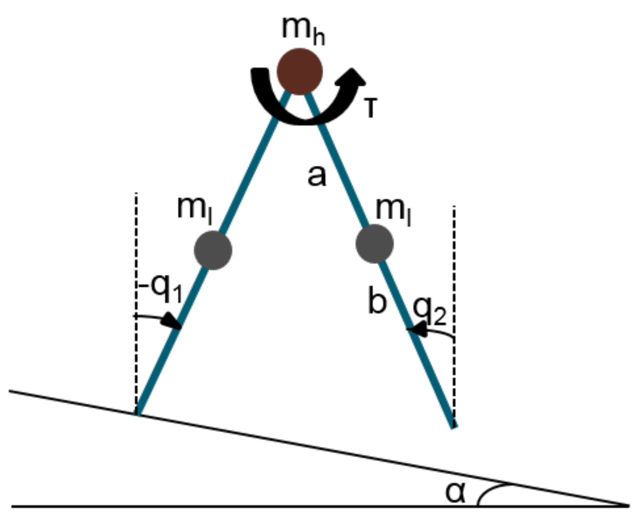


Figure 2.1: Compass walker.

Like all walking models, the compass walker is a hybrid system, i.e., it consists of phases of both continuous and discrete dynamics [30]. For walkers, the continuous dynamics are called the single support phase of the gait. This phase of walking is modeled by the Euler-Lagrange equations of motion:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B\tau \tag{2.1}$$

where *M* is the mass or inertia matrix, *C* is a matrix containing the centrifugal and Coriolis force terms, and *G* is a vector giving the gradient of the system's potential energy, which in this case consists solely of gravitational terms, with respect to the joint positions. In this instance, *B* is a matrix distributing the torque at the hip to each link, and tau is a scalar, the single hip torque.

The other phase of the gait is double-stance, where the tips of both legs touch the ground. This phase is approximated as being instantaneous, at the point that the tip of the swing leg hits the ground. At this point, the swing and stance legs, which remain in the same configuration, switch roles, and the tip of the former stance leg is freed to leave the ground. Kinetic energy is lost during this impact, leading to an instantaneous change in the joint velocities. The post-impact velocities, along with the impulsive forces at the impact with the ground, can be found using the following system of linear equations:

$$\begin{bmatrix} M_e & -J^T \\ J & 0 \end{bmatrix} \begin{pmatrix} \dot{q}_e^+ \\ F \end{pmatrix} = \begin{pmatrix} M_e \dot{q}_e^- \\ 0 \end{pmatrix} \tag{2.2}$$

where M_e is the inertia matrix for a model of the system with two additional states, which in this case are the horizontal and vertical positions of the former stance leg tip; and *J* is the Jacobian matrix mapping the tip of the former swing leg to each state.

For purposes of this experiment, four different optimization settings were tested. In two settings, symbolic differentiation (SD) was used to find the Jacobian of the constraints and the gradient of the objective function; the other two settings used numerical differentiation (ND). In a similar fashion, two of the settings transcribed the optimal control problem using Hermite-Simpson collocation (H-S), while the remaining two used trapezoidal collocation (TPZD).

2.1.1 Parameters, Decision Variables, Objective, and Constraints

The four main components of an NLP are parameters; decision variables (DVs); the objective, or cost, function; and the set of equality and inequality constraints. For the compass walker, the parameters include the lengths and masses of the legs, the hip mass, the value of the gravitational acceleration, and the ground slope. The values used in this study, which are predominantly based on those in [6], are given in Table 2.1. The total leg length is $(a + b)$.

Abbreviation	**Meaning**	**Value**
a	upper leg length	0.5 m
b	lower leg length	0.5 m
g	gravitational constant	9.81 m/s^2
m_h	hip point mass	10 kg
m_l	leg point mass	5 kg
α	ground slope	3°(downward)

Table 2.1: Compass walker parameter values.

There are four main categories of decision variables used in this setup, which are all stored in one large column vector called X. The states are listed first, and they alternate between joint positions and joint velocities, starting with q_1, or the angle of the stance leg with respect to the vertical, and ending with $\dot{q}_2$, or the velocity of the hip joint. The hip torques at all points are given next. After this is the time step, h, which is defined as the distance between two adjacent collocation points for TPZD and the distance between an adjacent midpoint and collocation point for H-S. In either case, this value is the same between all points. The transpose of X is given in Equation 2.3 below; here, n is the total number of points, including midpoints when needed. Technically, the states and torques are also considered to be column vectors, but they are written as row vectors here to make the expression less complicated.

$$X^T = \left[q_{1(1\times n)}\ \dot{q}_{1(1\times n)}\ q_{2(1\times n)}\ \dot{q}_{2(1\times n)}\ \tau_{(1\times n)}\ h\right] \tag{2.3}$$

Every optimization problem requires an objective, or cost, function. One frequently used cost function in bipedal robotics is torque-squared. This measure, expressed mathematically in Equation 2.4, sums up the squares of all *m* torques on the model at all times over the course of a step. Since a lower value of this measure shows higher efficiency of motion, this objective must be minimized. The exact implementation of this cost function is described in section 2.3.

$$F = \int_0^T \sum_{i=1}^{m} \tau_i^2 dt \tag{2.4}$$

Several different constraints are included in the model of this NLP. First, periodicity and configuration constraints are required. The configuration of the walker at the end of a step must be the mirror of that at the start of that step. Additionally, the post-impact velocities must mirror the velocities at the very start of the step. Regarding the configuration of the model, the tip of the swing leg must be on the ground at the start of a step, the stance leg must start out in front of the swing leg, and the length of a step should be within a prescribe range. Both the horizontal and vertical positions of the tip of the swing leg can be determined through the kinematics of the model.

The most critical constraints of this NLP are the continuous dynamics constraints. More specifically, it must be ensured that the system dynamics are satisfied at the collocation points. For a given state variable x, the following must be satisfied:

$$x[i+1] - x[i] = \int_{t[i]}^{t[i+1]} f(x,u,t)dt \tag{2.5}$$

where *f(x,u,t)* gives the dynamics of that state and *i* is the index of one of the collocation points. The equations of motion of the system were found either by hand or using Wolfram Mathematica, depending on the model's complexity. Given the discretized nature of an NLP, the integral above must be approximated; in this instance, this is done through either H-S or TPZD. This will be described in more detail in section 2.3.

When H-S is used, it is also important to look at the values of the states and control inputs at the midpoints between collocation points. The handling of the states will be discussed in section 2.3. The control inputs, however, are constrained to fit along a quadratic spline trajectory. These are the last constraints needed for this optimization.

2.2 Five-Link Walker

The underactuated five-link bipedal walker is far more complex than the compass walker. It has five DOF and four DOA, with an actuator at each hip and each knee. The legs are considered to be rigid bodies and thus have inertia; like the compass walker, however, this model has point feet. Equations 2.1 and 2.2 define the dynamics of this model as well. An illustration of this walker is presented in Figure 2.2.

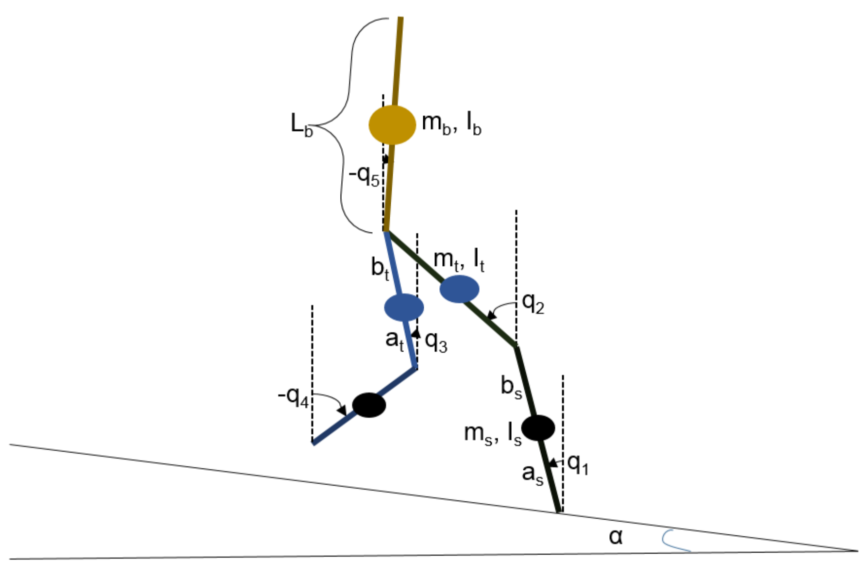


Figure 2.2: Five-link walker.

Just as this model's complexity is far higher than that of the compass walker, so are its continuous dynamics. Therefore, it was of interest to compare numerical and symbolic implementation and differentiation of these constraints. Three of the optimization settings tested for the five-link walker thus included various combinations of symbolic and fully numerical collocation constraints and numerical or symbolic derivation of the Jacobian. The last setting tested used slightly different constraints, as the accelerations at each collocation point were included as DVs in order to improve the problem's sparsity. All four settings employed trapezoidal collocation, and the non-collocation constraints were differentiated using ND.

2.2.1 Parameters, Decision Variables, Objective, and Constraints

The NLP of the five-link walker is more complicated than that of the compass walker. For the parameters, in addition to the ground slope and the gravitational constant, there are lengths, masses, and inertias for each limb (thigh, shank, and torso). Also, the number of points, DOF, and DOA are parameters. Many of the numbers for the five-link model are based on [31]; the values of the parameters are given in Table 2.2. The full length of the thigh, l_t, is $a_t + b_t$. The full length of the shank, l_s, is calculated similarly.

Abbrev.	Meaning	Value	Abbrev.	Meaning	Value
n	number of points	11 or 21	N	DOF	5
m	DOA	4	a_t	lower thigh length	0.25 m
b_t	upper thigh length	0.21 m	a_s	lower shank length	0.21 m
b_s	upper shank length	0.17 m	l_b	torso length	0.66 m
g	gravitational constant	9.81 m/s^2	m_s	shank mass	2.68 kg
m_t	thigh mass	6.65 kg	m_b	torso mass	34.0 kg
I_{thigh}	thigh inertia	0.12 $kg \cdot m^2$	I_{shank}	shank inertia	0.039 $kg \cdot m^2$
I_b	torso inertia	1.09 $kg \cdot m^2$	α	ground slope	0°(flat)

Table 2.2: Five-link walker parameter values.

Both the decision variables and objective see few major changes from the compass NLP. The DVs are arranged in a similar way to those for the compass walker, except that there are now five sets each of angular positions and velocities and four sets each of torques. Also, in the case where accelerations are included in the DVs, they are placed at the end of the DV vector. Torque-squared is still used as the cost function, although there are now four separate torques to sum up at each point in time.

The same types of constraints as are included in the compass optimization, except for those at midpoints, are needed for the five-link model. In addition, it is necessary to ensure that the legs are not hyper-extended at the knees; in other words, each leg must be constrained not to "bend" in the wrong direction. With the addition of knees, it is also possible to introduce constraints on foot clearance, or the ability of the tip of the swing leg to avoid touching the ground. At each non-end point along the gait, for this implementation, foot clearance is required to be at least 10^{-6} m, and at a point approximately one third of the way along the gait cycle, foot clearance has to be at least five centimeters. This is in order to approximate the clearance the foot would have if it were attached to an ankle, as is the case in real life. Lastly, in order to improve the likelihood of a reasonable gait, constraints are in place to ensure that the center of mass of the torso can never move backward horizontally between any two subsequent time steps.

3. DEVELOPMENT OF CODE FOR OPTIMIZATION PROCEDURE

3.1 General Structure

The MATLAB IPOPT interface, mexIPOPT, requires functions, bounds, parameters, and DVs to be handled in a certain way. The DVs must be in a vector, and the initial guess must be provided. The parameters must be stored in a data structure, which can then be called inside other functions, such as those for the constraints. The required function handles and DV and constraint bounds, as well as any special IPOPT options, are stored in data structures.

The first structure needed is the *funcs* structure. This contains function handles for several important parts of the problem, as outlined in Table 3.1. As the table mentions, the Jacobian and its structure are sparse matrices. In this study, these were found by recording the row, column, and value of nonzero elements in said matrices and forming the sparse matrix with MATLAB's *sparse()* command.

The other structure to include in the optimization is the options structure, which the author has called *opt*. The most important fields in this structure are vectors giving the upper and lower bounds for both the DVs and the constraints. There are also special IPOPT settings which can be specified here, such as the convergence tolerance and the maximum number of iterations permitted [32]. The values used in the present study can be found in Table 3.2. IPOPT can optionally take a user-defined Hessian, but as seen in row five of this table, it was approximated by IPOPT itself in this case.

Field	Definition
funcs.objective	objective function value for a given set of DVs
funcs.gradient	gradient of objective function evaluated for a given set of DVs
funcs.constraints	vector with value of each constraint for a given set of DVs
funcs.jacobian	sparse matrix with Jacobian of constraints for a given set of DVs
funcs.jacobianstructure	sparse matrix with ones where Jacobian can be nonzero

Table 3.1: Fields in function data structure (*Funcs*).

Field	Value
mu_strategy	'adaptive'
max_iter	1000 (compass) or 10000 (five-link)
tol	10^{-8}
acceptable_tol	10^{-6}
hessian_approximation	'limited-memory'
derivative_test	'first-order'

Table 3.2: IPOPT options.

3.2 Commonalities Between All Settings

The main code required to run these optimizations can be divided into three main categories: constraints, the cost function, and the differentiated terms. The "differentiated terms" include the gradient of the cost function, the Jacobian of the constraints, and the pattern of this Jacobian matrix; these will be discussed in sections 3.3.1.1.

The constraints were not all written in one file; rather, they were broken down into five separate sets of constraints and later aggregated within a separate function. The first set of constraints, the linear constraints, includes joint configuration periodicity and initial thigh orientations. An example of the former, for the compass walker with n = 11, is:

$$X(1) - X(33) = 0 \tag{3.1}$$

In this example, the stance leg angle at the first collocation point is made equal to the swing leg angle at the last collocation point. An example of the latter would be:

$$X(1) \geq 0 \tag{3.2}$$

In this case, The initial angle of the stance leg must be positive at the first collocation point. For all of these constraints, the upper and lower bounds are stored separately from the constraint expressions. The way to enter constraint expressions is demonstrated in Figure 3.1.

```
n = auxdata.n;
N = auxdata.N;
c = zeros(2+N,1);

%constraints - specific to 5-link
%initial sign of positions
c(1) = X(2*n+1); %stance hip
c(2) = X(4*n+1); %swing hip
%periodicity of positions
c(3) = X(7*n) - X(1); %from old swing shank to new stance shank
c(4) = X(5*n) - X(2*n+1); %from old swing hip to new stance hip
c(5) = X(3*n) - X(4*n+1); %from old stance hip to new swing hip
c(6) = X(n) - X(6*n+1); %from old stance shank to new swing shank
c(7) = X(9*n) - X(8*n+1); %still torso
```

Figure 3.1: Linear constraints for five-link walker.

The next set of constraints, the kinematic constraints, only has two entries. The first is the constraint requiring that the swing foot begin on the ground. This constraint is given in Equation 3.3 for the compass walker; it compares the foot position based on cosines of the joint angles to the foot position based on the right triangle between the legs' endpoints. Figure 3.2 illustrates this.

$$l \cos X(1) - l \cos X(43) = -(-l \sin X(1) + l \sin X(43)) * \tan \alpha \tag{3.3}$$

The second constraint in this set is for the step length. It is also found kinematically.

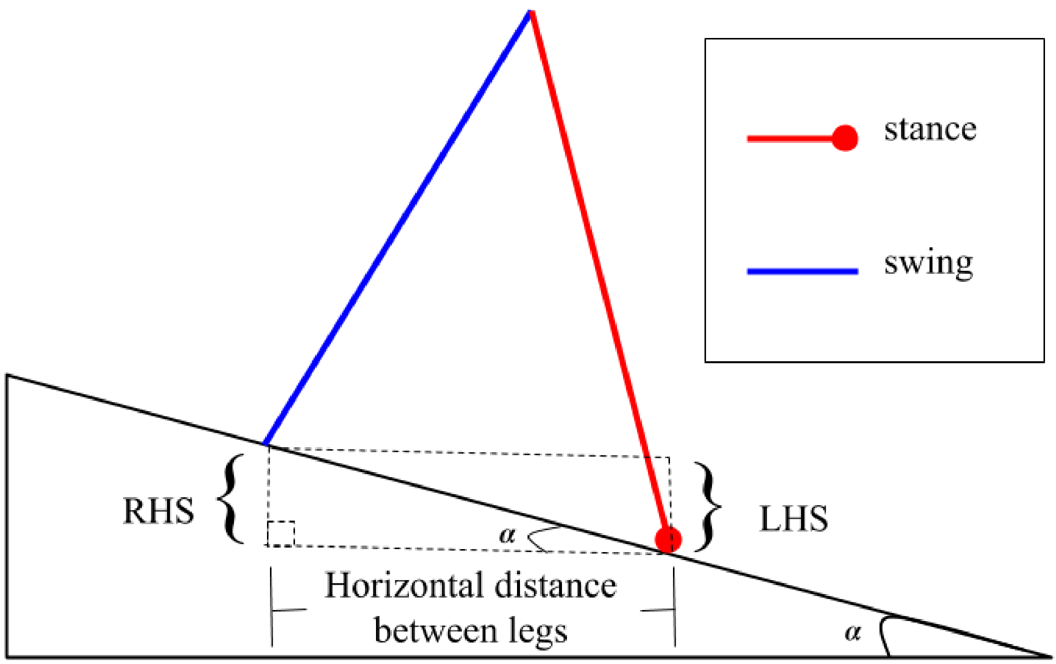


Figure 3.2: Two methods of calculating swing foot position. LHS and RHS refer to equation 3.3.

The five-link walker has an additional constraint in this category; this is a constraint requiring that the tip of the swing foot must be at least five cm (0.05 m) above the ground at a fixed point along the gait. The exact point at which this constraint was applied can be found in Table 3.3.

Number Collocation Points	Point of Application
11	4
21	7
101 (Baseline Gait Only)	31

Table 3.3: Point at which 5 cm clearance constraint is applied.

The next constraints included in the optimization are the constraints which must be satisfied at some, or all, collocation points. For both walking models, these constraints include the continuous dynamics, which are explained in more detail in the next subsection.

The five-link walker has three additional sets of constraints in this section. First, foot clearance is required to be greater than or equal to 10^{-6} at all points along the gait except for the very first and very last. This is done using the same methodology as in Equation 3.3, except that the foot positions are evaluated at different points. Another set of constraints applied at the non-end points of the five-link walker's gait requires the center of mass of the torso, which is taken to be at its center, to never move backwards horizontally. This is accomplished by saving the horizontal position at the previous collocation point, using kinematics to determine its present position, and finding the difference between the two. The last additional point constraints on this model, which are applied at all points, restrict knee extension. These are shown in Figure 3.3, and their value must be greater than or equal to zero.

```
for ii=1:n
    %knees: (hip angle) - (knee angle) > 0
    c(c_ind) = X(ii+2*n) - X(ii); %non-overextension of stance knee
    c_ind = c_ind+1;
    c(c_ind) = X(ii+4*n) - X(6*n+ii); %non-overextension of swing knee
    c_ind = c_ind+1;
end
```

Figure 3.3: Knee extension constraints. Here, "hip angle" refers to the angle of the upper leg, while "knee angle" refers to that of the lower leg.

Another set of constraints is the impact constraints. For each walking model, a MATLAB function was created which uses Equation 2.2 to calculate post-impact velocities. The impact constraints call this function with the states at the last collocation point. They then subtract the post-impact velocities from the velocities at the corresponding (flipped) velocities at the first collocation point. These constraints must be equal to zero.

The objective function, or the torque-squared, is expressed as in Equation 2.4. The integral in this equation must be discretized using the methods in section 3.3.1.2.

3.3 Variations Between Optimization Settings

3.3.1 Variation in Compass Runs

3.3.1.1 Different Differentiation Methods

As previously mentioned, IPOPT requires the Jacobian of the constraints and the gradient of the objective function. When numerical differentiation is used, getting these derivatives is very straightforward. The function *get_numerical_diff()* was written to return the row, column, and value of nonzero elements of the Jacobian of a certain function, which is passed to *get_numerical_diff()* as a function handle. In order to improve accuracy, the central difference approximation was used. This method is explained in Equation 3.4.

$$\frac{df}{dx} \approx \frac{f(x + \epsilon \times perturb) - f(x - \epsilon \times perturb)}{2\epsilon} \tag{3.4}$$

Here, *perturb* is a vector of all zeros except for one element, the element corresponding to the variable with respect to which the derivative is being taken. Epsilon was made equal to $1e-6$. The main body of the function is shown in Figure 3.4, and this same function was called for all contraints, as well as the objective.

```
%derivatives
for jj = 1:length(X)
        perturb_vec = zeros(size(X));
        perturb_vec(jj) = 1;
        c_prev = constraint_fxn(X - ep*perturb_vec,auxdata);
        c_after = constraint_fxn(X + ep*perturb_vec,auxdata);
     for ii = 1:length(c_prev)
        c_deriv = (c_after(ii)-c_prev(ii))/(2*ep);
     if c_deriv ~= 0
         i(end+1) = ii;
         j(end+1) = jj;
         k(end+1) = c_deriv;
     end
    end
end
```

Figure 3.4: Numerical calculation of derivatives. Vector i gives rows for the sparse matrix, j gives columns, and k gives the value of the elements there.

The process to get symbolic derivatives was more involved. Rather than calling the constraint function repeatedly, it was necessary to create a separate function for the Jacobian of each set of constraints. MATLAB's symbolic toolbox was used to achieve this. First, symbolic variables were created to represent the DVs; next, the constraints were set up symbolically. After this, the symbolic toolbox's *diff()* function was used to find the derivative of each constraint with respect to each decision variable. The resulting elements, as well as their row and column locations, were then stored in the same way as in the numerical case.

These results then had to be saved for future use. The row (i) and column (j) vectors were stored in a *.mat* file so that the vectors for all the sets of constraints could be aggregated later on. The Jacobian elements were saved using the *matlabfunction()* function, which saves a symbolic expression as a *.m* file.

The Jacobian structure, which is a constant, can be calculated either numerically or symbolically, and the method used was not changed for different settings. For the five-link walker, it was derived numerically, and it was derived symbolically for the compass walker. When determined numerically, a perturbed version of the initial guess was used to numerically determine the zero and nonzero elements; it could thus be different each time. When determined symbolically, the same *.mat* files that were used to construct the Jacobian were aggregated to create this sparse structure. The one exception was with the Jacobian structure for the constraints on the control input at the midpoints when H-S was used; in that case, several 1s were added in a block, without reference to the symbolically-determined nonzero derivative locations.

3.3.1.2 Different Collocation Methods

The variation between transcription methods is a bit more straightforward than that between differentiation methods. TPZD is the simplest case. For this collocation method, all points are also collocation points. Equation 2.5 takes the following form when this method is used:

$$x[i+1] - x[i] = \frac{h}{2}(f[i+1] + f[i]) \tag{3.5}$$

where h is the time step and i is the index of one of the collocation points. This expression is used at all points except for the last, and it is applied for each position and velocity.

The H-S case is a bit more complicated. With this method, only every other point is a collocation point. At the collocation points, the following collocation constraints are applied:

$$x[i+1] - x[i] = \frac{h}{6}(f[i] + 4f[i+1/2] + f[i+1]) \tag{3.6}$$

When implemented in MATLAB, [i+1] becomes [i+2] and [i+1/2] becomes [i+1]. The states at the midpoints are found using the following equation:

$$x[i+1/2] = \frac{1}{2}(x[i] + x[i+1]) + \frac{h}{8}(f[i] - f[i+1]) \tag{3.7}$$

where the indices are adjusted in the same way as above. All of these collocation equations come from [5].

It is useful to also constrain the torques at the midpoints, as the dynamics at the midpoints depend on them as well. In this study, this was accomplished by calculating the expected midpoint torques along quadratic splines. In this case, the spline coefficients were determined based on the torques at the collocation points. These expected midpoint torques were then subtracted from their corresponding DVs in an effort to force them to be equivalent. Figure 3.5 shows this. Since there are eleven collocation points in this example, there are twenty-one total points in the H-S implementation.

```
u_midpts = quadinterp(0:2*h:20*h,X(85:2:105),h:2*h:19*h);
%fprintf('u_midpts: %f; X(105): %f\n',u_midpts(10),X(105));
%length(u_midpts)

m = 1;
for kk = 2:2:20
    c(c_ind) = X(kk+84) - u_midpts(m);
%     if kk == 20
%         fprintf('Last midpoint: %f (u_midpt = %f, X(104) = %f)\n'
%     end
    c_ind = c_ind+1;
    m = m+1;
end
```

Figure 3.5: Constraints on midpoint control torques.

3.3.2 Variation in Five-Link Runs

The four settings tested for the five-link walker are different from those used with the compass walker. All four test cases for the five-link biped use TPZD, and all constraints, as well as the objective function, are differentiated numerically except for–in some cases–the collocation constraints.

One setting has its constraints set up and differentiated in the same way as the compass TPZD-ND setting. A second setting uses the same differentiation method, but in the collocation constraints themselves, the joint accelerations required in Equation 3.5 are found using a symbolic expression rather than a numerical one. The symbolic function for the accelerations was found by creating symbolic variables for the DVs, finding the accelerations symbolically, and saving the result as a *.m* file with *matlabfunction()*.

This same partiially-symbolic set of constraints was also used in a third optimization setting. In this setting, however, the differentiation for the collocation constraints was done symbolically as well. This symbolic calculation was done in several parts, as described in Table 3.4. In this table, *lin* corresponds to linear or cross-term parts of the constraints, *dc* corresponds to the continuous dynamics at the current point, and *dn* corresponds to the dynamics at the next collocation point.

	x = Position	**x = Velocity**
lin	$x[i+1] - x[i] - (\frac{h}{2}(f[i+1] + f[i]))$	$x[i+1] - x[i]$
dc	—	$-\frac{h}{2}f[i]$
dn	—	$-\frac{h}{2}f[i+1]$

Table 3.4: Breakdown of symbolic Jacobian for collocation constraints of five-link biped.

As was the case with the compass model's symbolic differentiation, the nonzero-element row and column indices were stored in *.mat* files and combined into one vector for all the constraints, with row and column adjustments being made for the constraints at the non-initial collocation points. The Jacobian elements from *lin*, *dc*, and *dn* were also combined in order to yield the complete derivative of a constraint with respect to each decision variable. This was done by checking for (row, column) matches between the different lists of rows and columns and summing the corresponding derivative terms.

The last five-link setting is quite different from the other three because the accelerations at each collocation point are added to the DVs. This changes some of the constraints. First, separate continuous dynamics constraints are added to verify that the positions, velocities, accelerations, and torques at every collocation point satisfy the Euler-Lagrange equations of motion. Next, the collocation constraints are simplified because they can use the indices of the acceleration values directly from the DV vector, rather than calculating them first.

4. SIMULATION-BASED EXPERIMENT AND STATISTICAL ANALYSES

4.1 Baseline Gaits and "Accuracy" as Consistency with a Baseline

Before discussing the full experimental procedure, one element of the data analysis must be described. One of the key factors examined in this study was the accuracy of gaits resulting from each set of optimization settings. In order to do this, it was first necessary to establish a baseline gait (BG) which could then be compared against others. For the compass walker, this gait was determined through a passive forward simulation of a step on a three-degree downslope, which is the same slope used in the simulations; meanwhile, for the five-link walker, an optimization with a finer "mesh" was performed to get it.

The initial point for the compass gait forward simulation, which is the point just after the heel strike impact, was found by utilizing the concept of a Poincaré map [9]. This concept is demonstrated in Figure 4.1. It is possible to discretize a process like walking by mapping the states at one point in a step–the states on a so-called *Poincaré section*–to the states at that same point in the next step. Since these walking gaits are meant to be periodic, these states should map to themselves, as in the equation below:

$$P(x^*) = x^* \tag{4.1}$$

where P is the Poincaré map and x* is a fixed point. This fixed point can thus be found using a root finding procedure such as the Newton-Raphson method:

$$x[i+1] = x[i] - (\nabla F(x[i]))^{-1} F(x[i]) \tag{4.2}$$

In this case, the function to find the root of is $F(x) = P(x) - x$. The resulting gait is shown in Figure 4.2.

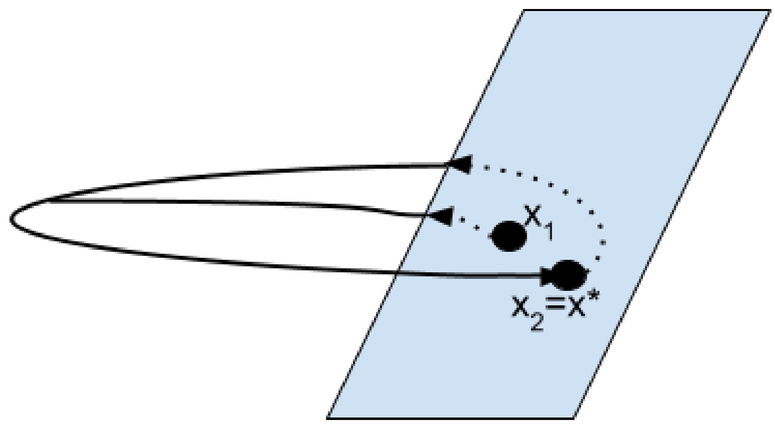


Figure 4.1: Illustration of Poincaré map.

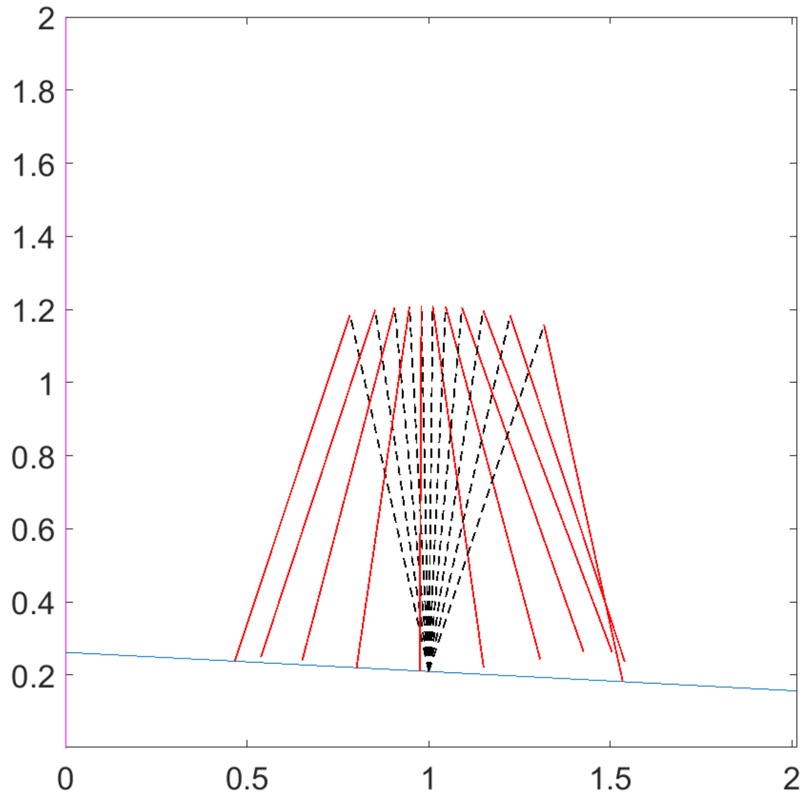


Figure 4.2: Baseline gait for compass walker. Stance leg in black, swing leg in red.

The BG for the five-link biped was determined using methods similar to those described in the next section. One optimization was run using a randomized initial guess, symbolic dynamics constraints, SD, and 101 collocation points. It should be noted that a different initial guess could potentially result in a different gait. The baseline gait is shown in Figure 4.3.

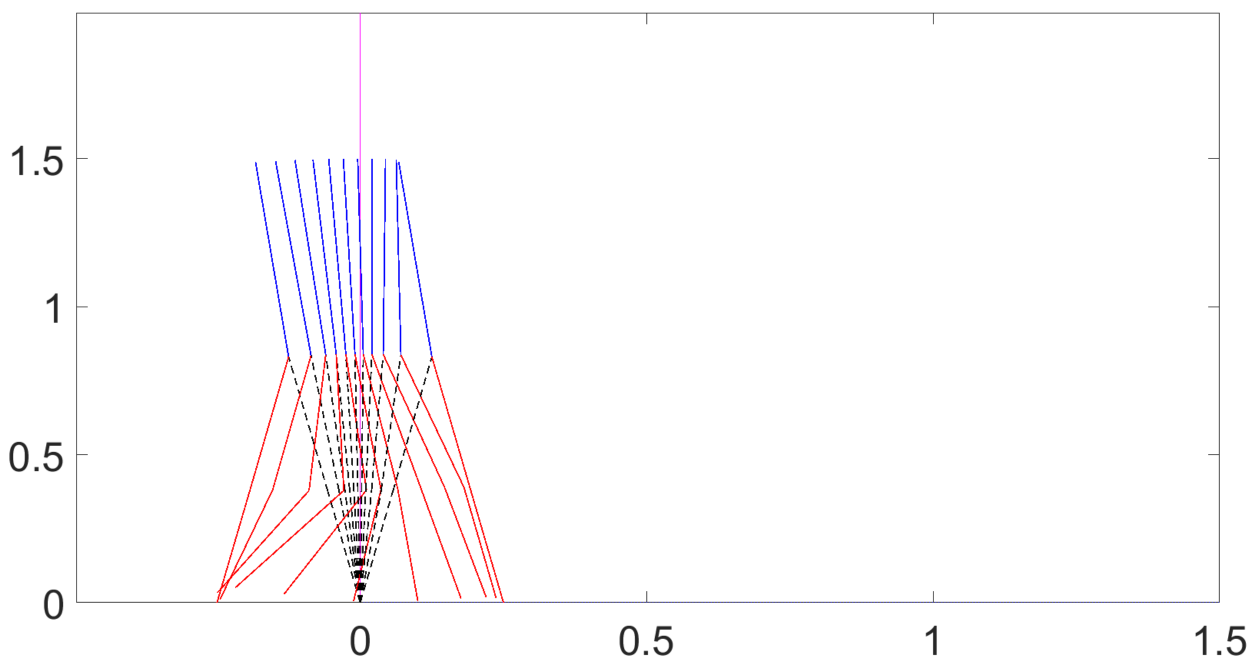


Figure 4.3: Baseline gait for 5-link model. Stance leg in black, swing leg in red, torso in blue.

These gaits were then compared against each generated optimal gait. For either model, the states along these BGs at eleven or twenty-one constant intervals as required were saved into a *.mat* file. During data analysis of the optimal gaits, the difference between the value of each state at each BG saved point and the value at the corresponding collocation point along the optimal gait was calculated. Summing the square of the differences for a given state gave a measure of accuracy for each state. For the compass gait optimizations using Hermite-Simpson, only the state values at the collocation points (i.e., at every other point) were compared for this measure.

4.2 Experimental Setup and Execution

The NLPs described previously were used to generate optimal gaits. All optimization runs were completed using the student version of MATLAB R2019a running on a Dell Latitude E6540 laptop; this computer has an Intel Core i7-4800MQ processor and is running Windows 7 Professional with Service Pack 1. Runs were completed during February 2020, and while they were being completed, MATLAB was the only manually-running program on the computer. Additionally, the laptop was not plugged in during these runs, as resulting changes in the computer's power settings or CPU usage could confound timing results. The mexIPOPT package for MATLAB was used to solve the NLPs with the solver mumps.

In order to obtain statistical data, each optimization setting was tested multiple times. Most of these optimization runs started with a randomized initial guess (IG). First, a vector of zeros was created, and then a for loop was used to populate each element of the vector with a value between zero and one, as determined by MATLAB's built-in rand() function. For other runs, a fixed IG using values of the states from the baseline gaits and zeros elsewhere was used. For the compass walker, eleven collocation points were used in each optimization. For the five-link biped, some runs used eleven collocation points, while others used twenty-one. Two numbers of points were used for the more complex walker in order to see whether increasing the number of collocation points would affect the results.

A script was used to automate the optimization runs and data collection for this experiment. First, the MATLAB command window and workspace were cleared, and all figures were closed. Then, any necessary folders were added to the MATLAB path, and the data for a fixed IG may or may not have been loaded, depending on the commenting in the script. The remaining code for the experiment was executed within a for loop. The number of iterations run at a time varied. Conditionals determined which settings would be used in any given iteration, and a key corresponding to these settings was assigned in order to facilitate the recording of data. The meanings of these keys are presented in Table 4.1.

Key	Compass	5-Link
1	H-S + ND	Symbolic Dynamics + SD without Accelerations as DVs
2	TPZD + ND	Symbolic Dynamics + ND without Accelerations as DVs
3	H-S + SD	Numerical Dynamics + ND without Accelerations as DVs
4	TPZD + SD	ND + Accelerations as DVs

Table 4.1: Optimization setting keys.

From this point, the script ran similarly for each method. The IG was initialized, and the data structure containing the necessary function handles, constraint bounds, and DV bounds was created. Then, the optimization was run. Afterwards, another script collecting results was called. In this script, the states, control inputs, and time step were extracted from the final vector of decision variables. The times at each collocation point were calculated, and walking tiles were generated. Interpolations of the states and control inputs were performed; these were also plotted. The preceding data and figures were then saved in a folder, for future access. Lastly, several pieces of data were then saved directly into an Excel spreadsheet, with each sheet holding data for a different key. These data included the date and time of the data's recording, the objective value of the run, the number of iterations and CPU time required, and the "accuracy" measure of each joint's position and velocity.

4.3 Statistical Methods Employed in Data Analysis

The data recorded in Excel spreadsheets was further analyzed. First, another column, time per iteration, was added to the spreadsheet, and these values were calculated by dividing the CPU time by the number of iterations for a given run. Next, the mean and sample standard deviation of each measure for each set of optimization settings were determined. Calculating the means and standard deviations gave a rough idea of which settings might be considered "better."

These indicators are only rough, however, and they do not distinguish between the effects of each individual modified setting, such as only the differentiation method, on their own. In order to determine this, an analysis of variance (ANOVA) was performed for each measured variable for the runs using a randomized initial guess. ANOVA is a technique which determines whether variations between sets of data are due to randomness or due to differences between the factors used in the study. Two-way ANOVAs were performed in two instances: in the analysis of the compass data and in the comparison between the acceleration and Jacobian calculation methods for the five-link walker (first three settings only). Meanwhile, a simple one-way ANOVA was used to compare the five-link setting which included acceleration in the DVs to the three settings which did not.

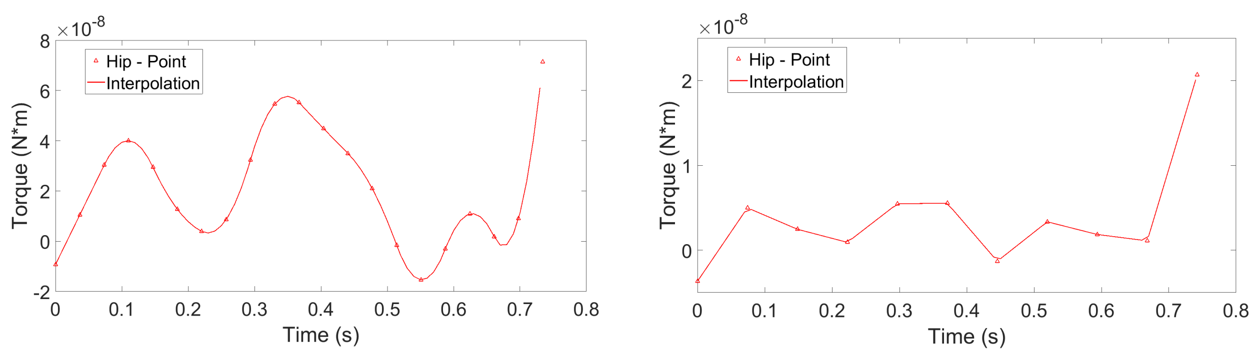
This analysis was performed using Minitab® Statistical Software[1] (Minitab19 software); this was completed on the same laptop as the MATLAB optimizations, but the software was accessed through Texas A&M University's Virtual Online Access Lab (VOAL). All two-way ANOVAs were also set up to return significance values of the interactions between factors. However, due to the fact that symbolic Jacobian calculation was only used in one five-link setting, the interactions could actually only be determined for the compass walker. A confidence level of 95% was assumed, so factors or interactions resulting in $p \leq 0.05$ were considered to be significant.

[1]MINITAB® and all other trademarks and logos for the Company's products and services are the exclusive property of Minitab, LLC. All other marks referenced remain the property of their respective owners. See minitab.com for more information.

5. RESULTS AND DISCUSSION

5.1 Compass Gait Results

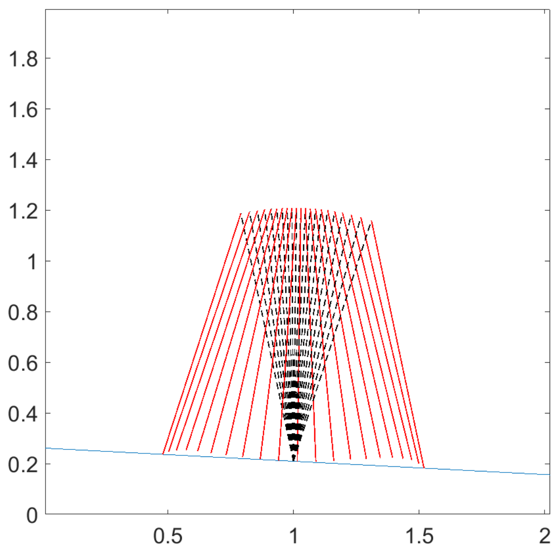
Six-hundred runs were performed for the compass walker. One-hundred runs for each of the four settings were found using randomized initial guesses, and fifty of each were found using the baseline gait as the initial guess. The four-hundred runs with a randomized IG were analyzed further. In 339 of the 400 random-IG gaits and all of the gaits generated from the baseline IG, the gait was, within a small error, the gait shown in Figure 4.2. The actual control signals generated varied, as the control values at each collocation point are effectively zero. Figure 5.1 gives two examples of interpolated control signals, one from H-S (quadratic spline interpolation) and one from TPZD (linear spline interpolation). These low-magnitude control signals are reasonable, considering that the most optimal gait achievable is a passive one ($\tau = 0$ everywhere). The generated gaits generally had a slightly higher (slower) period than the baseline gait (approximately 0.7344 s vs 0.7342 s).



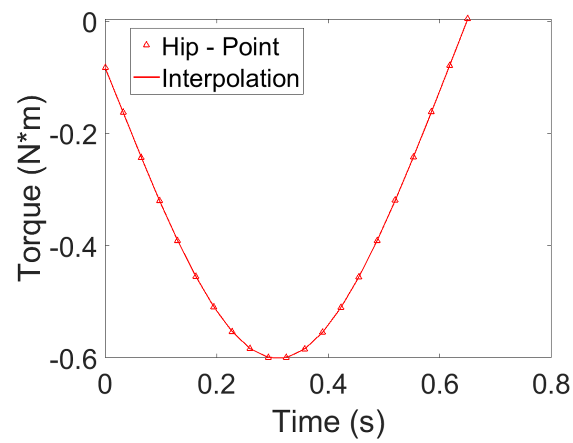
(a) Hermite-Simpson. (b) Trapezoidal.

Figure 5.1: Example compass walker "passive" control signals.

The remaining sixty-one gaits generated from random initial guesses were different. The walking tiles and a representative control signal of a representative "alternative" gait are shown in Figure 5.2 below, although the control signal was slightly different when TPZD was used. This different gait was likely another local minimum found because of the specific IGs used. This gait's period was approximately 0.65 s, and its objective value was approximately 0.1183 sq. (N*m). As is shown, these control inputs are higher than those for the "passive" gait.



(a) Walking tiles.



(b) Control signal.

Figure 5.2: Alternative compass gait.

Time-related statistics for the runs with a randomized initial guess are presented in Figures 5.3 and 5.4. The data presented therein include runs similar to both the baseline gait and the secondary, non-passive gait, as evidenced by the large error bars, which represent the sample standard deviations. At first glance, it appears that H-S is noticeably slower than TPZD and that symbolic differentiation may be slower than numerical differentiation in this study, when examining CPU time. These results will become more clear when the ANOVA results are provided in section 5.3.2.

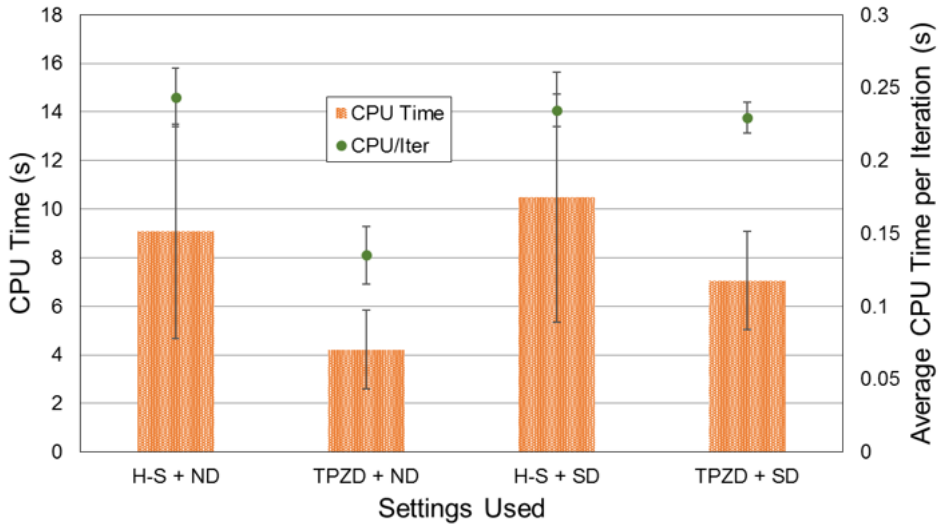


Figure 5.3: Overall CPU time and average CPU time per iteration for compass optimizations using randomized initial guess.

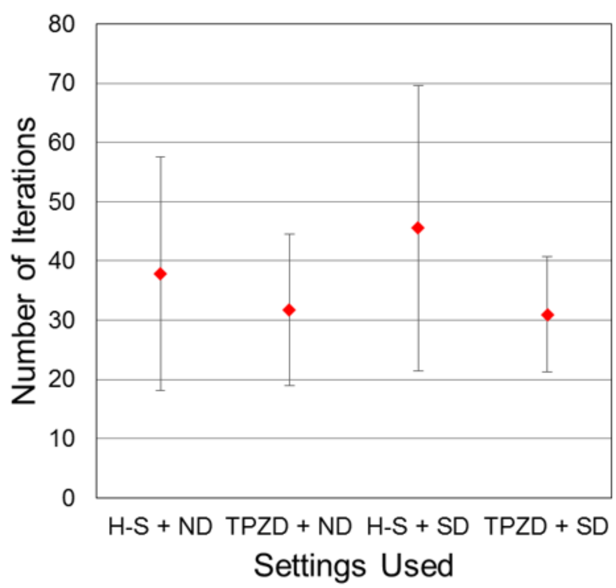


Figure 5.4: Number of iterations for compass optimizations using randomized initial guess.

The differences between the closeness of the different generated gaits to the baseline for each setting are less clear, at least when examining the gaits that clearly were converging towards a different local minimum. This can be seen in Figure 5.5, which gives the similarity to the baseline for each joint position. Only a few error bars are visible. Similarities for the joint velocities were also computed. This figure makes it very clear that a relatively small percentage of "alternative" gaits can still significantly skew results. For all data fields, ANOVAs were performed for the data both including and excluding these runs.

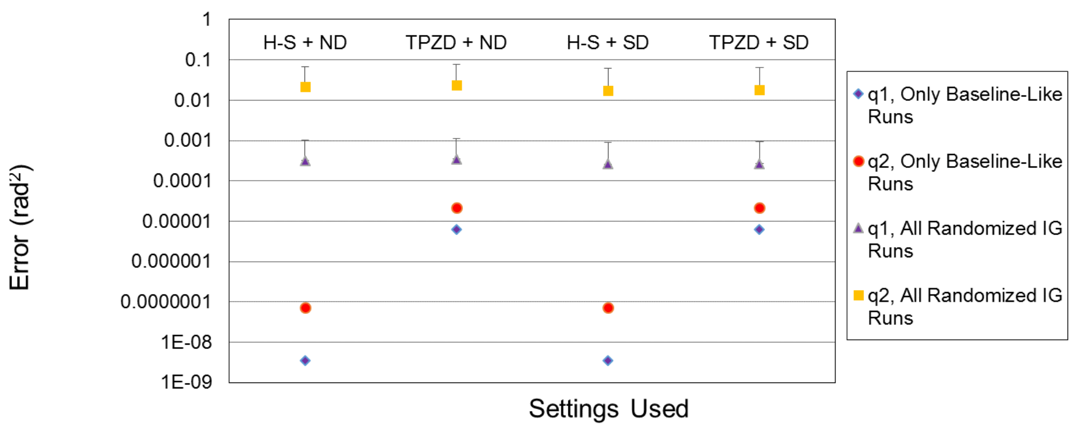


Figure 5.5: Measure of error between generated gaits and baseline for compass optimizations using randomized initial guess.

5.2 5-Link Walker Results

320 runs were completed for the 5-link biped. Twenty of each setting with each number of collocation points were performed with a randomized initial guess and with the baseline gait as the initial guess. The eighty randomized-IG runs for each number of collocation points were analyzed. Just as in the compass case, more than one gait was found for the 5-link walker, with the most common being the baseline gait in Figure 4.3. One generated control signal for this gait is shown in Figure 5.6. The control signal varied somewhat depending on the number of collocation points used, although the overall shape of the splines were the same. The objectives for eleven and twenty-one collocation points were approximately 4.7136 and 3.8801 sq. (N*m), respectively.

It is notable that the secondary gaits occurred far fewer times with the 5-link walker than with the compass walker. When eleven collocation points were used with a randomized IG, the alternative gaits came up six times; they only came up three times when twenty-one points were used. An alternative gait can be seen in Figure 5.7. Its period was approximately 1.5 s, and its average objective value was approximately 10.98 sq. (N*m) for the twenty-one- point runs and varied somewhat for the eleven-point runs, as different alternative gaits were found with that number of collocation points.

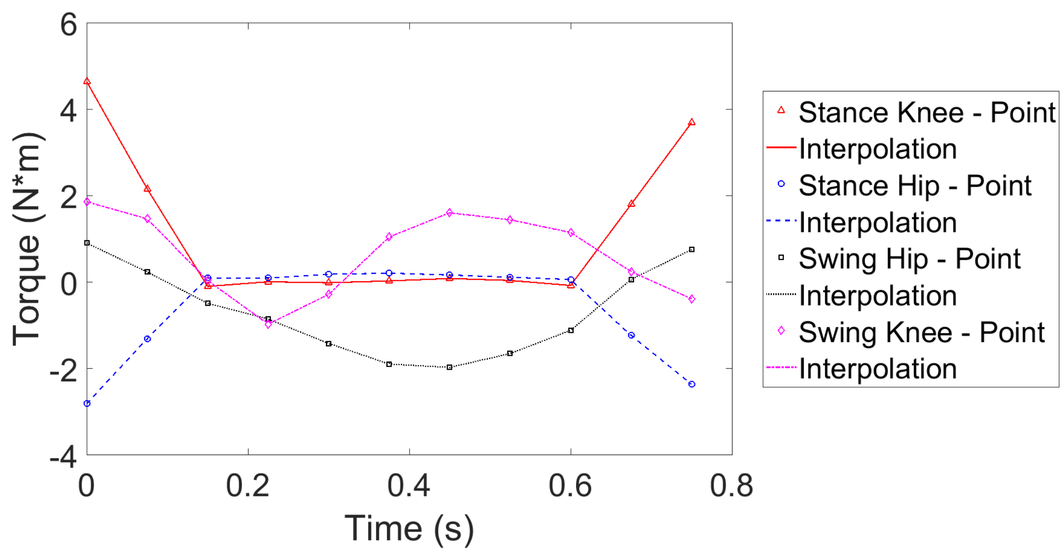
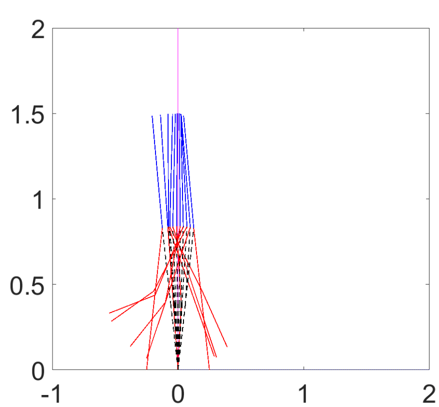
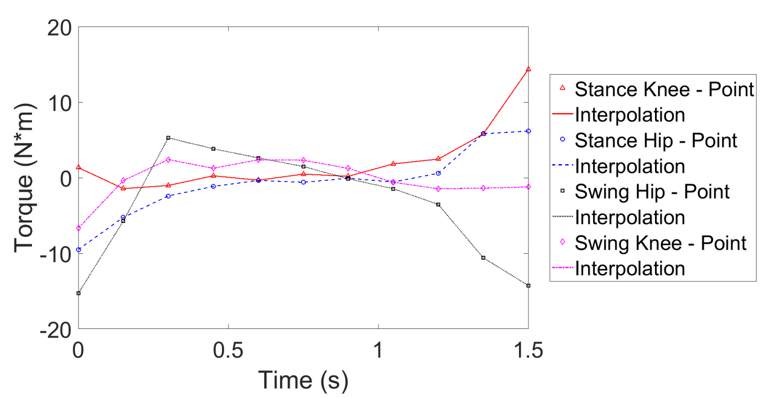


Figure 5.6: Control signals for 5-link gait.



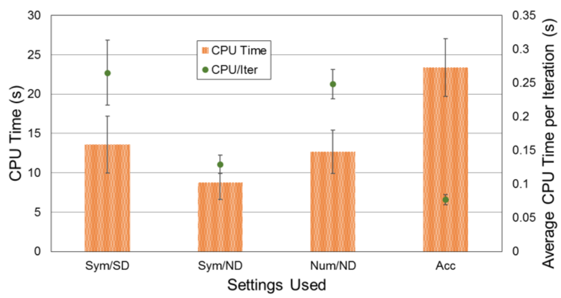
(a) Walking tiles.



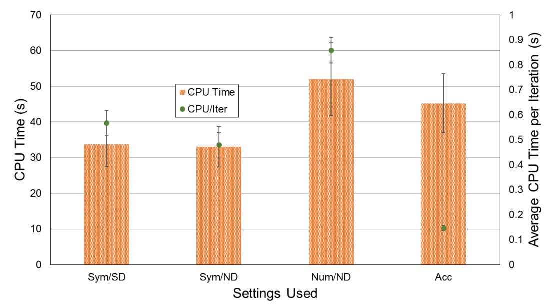
(b) Control signal.

Figure 5.7: Alternative 5-link gait.

The same forms of data collected for the compass walker were collected for the five-link walker for both eleven and twenty-one collocation points. The most intriguing observations from Figures 5.8 and 5.9 concern the runs where acceleration was included in the DVs (last column). Although these runs were significantly faster on a per-iteration basis, the number of iterations required for these runs was so much higher that the overall CPU performance was slower, except in the 21-point case, where the numerical acceleration + ND case was slower still. This will be examined more closely when presenting the ANOVA results and tradeoffs encountered in this study.

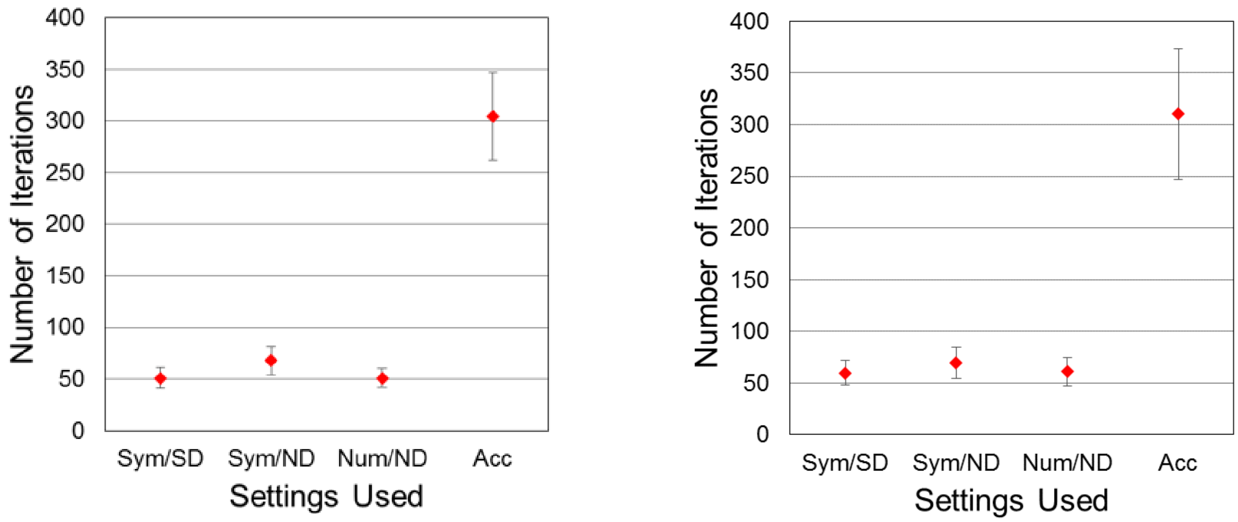


(a) 11 collocation points.



(b) 21 collocation points.

Figure 5.8: Overall CPU time and average CPU time per iteration for 5-link optimizations using randomized initial guess.

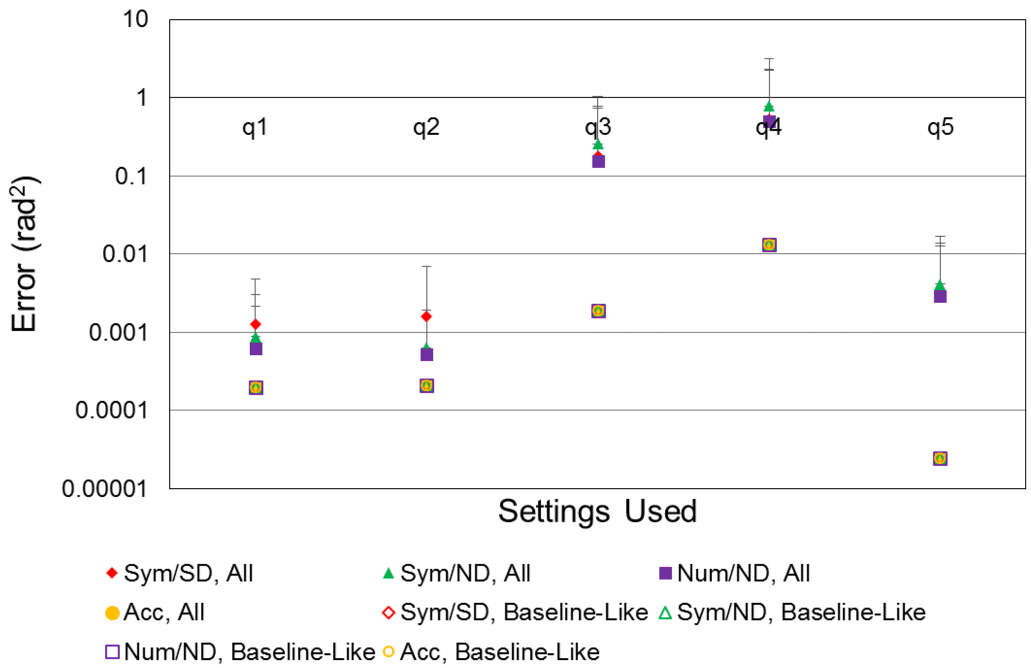


(a) 11 collocation points.

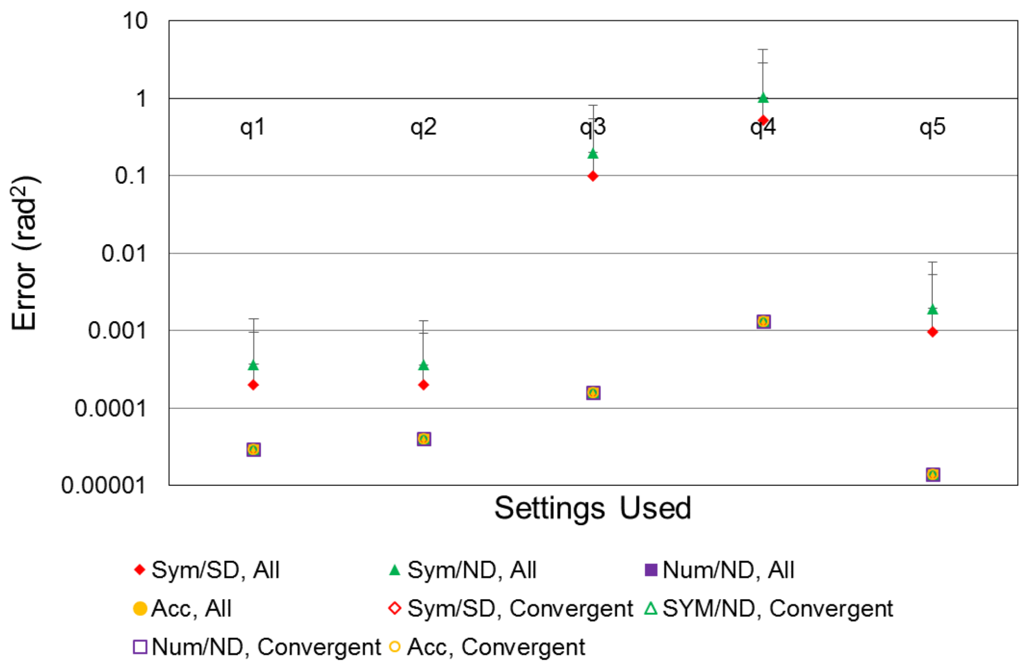
(b) 21 collocation points.

Figure 5.9: Number of iterations for 5-link optimizations using randomized initial guess.

It is also illustrative to see how the generated gaits compared to the baseline when using these settings. Figure 5.10 illustrates two interesting points. First, the mean errors when including the "alternative" gaits are significantly higher than otherwise, excluding the setting where acceleration is included in the decision variables for both numbers of collocation points and the numerical + ND setting with twenty-one collocation points. In those cases, the means are roughly the same, as no runs converged to the "alternative" gait. The second conclusion to glean from this figure is that when the "alternative" runs are excluded, the means are roughly the same for each setting. In that case, the standard deviation is also higher for the accelerations-in-DVs setting than for the others.



(a) 11 collocation points.



(b) 21 collocation points.

Figure 5.10: Measure of error between generated gaits and baseline for 5-link optimizations using randomized initial guess.

5.3 Determination of "Best" Settings

5.3.1 Implementation

Before discussing the experimental results, it is useful to discuss the variations between different settings in terms of implementation time and complexity. With regards to the collocation method used, the implementation time was not too different. The settings using H-S had a greater number of constraints and points, which could potentially have made the NLPs harder to solve, but otherwise, there were no major differences. In summary, TPZD was slightly easier to implement.

The greatest differences in implementation were between numerical and symbolic differentiation. As mentioned in section 3.3.1.1, for ND, the same function, *get_numerical_diff()*, was called repeatedly to differentiate all the constraints, as well as the objective function. Therefore, very little new code needed to be written for the implementation of ND. Debugging was also fairly straightforward, as any changes to the constraints or objective themselves were automatically reflected in the Jacobian or gradient, respectively.

This differed significantly from the SD case. As mentioned previously, new MATLAB functions had to be created for the Jacobian and gradient. This resulted in three main implementation issues. First, for SD, there were more files to keep track of. Secondly, it could take time to derive the symbolic expressions needed, and the resulting new files could be quite large. This was particularly a problem for the five-link accelerations and their Jacobian, as shown in Figure 5.11. These problems would only increase for higher-order systems, so the scalability of SD is questionable at best. Lastly, the existence of multiple files with SD made debugging more difficult. Any time one of the constraints was changed, the *.mat* and *.m* files associated with its Jacobian had to be rederived and resaved. Overall, the implementation of ND is preferable to that of SD.

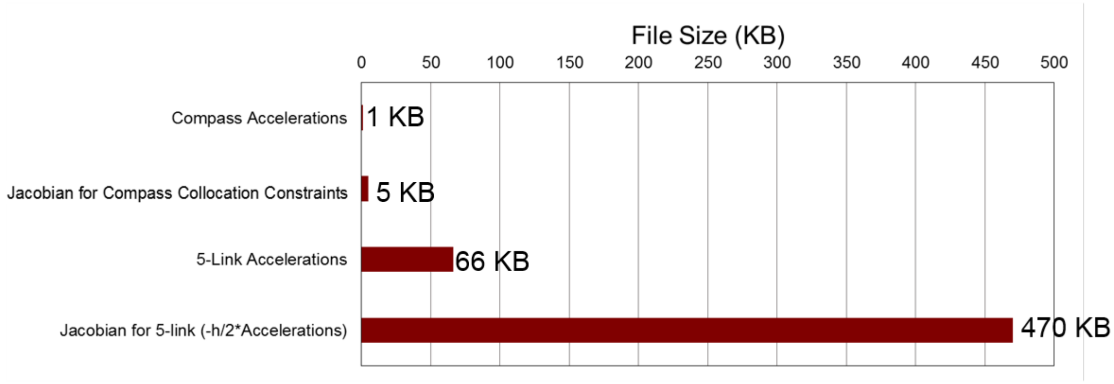


Figure 5.11: Comparison of file sizes for 2-DOF and 5-DOF models.

The last type of modification in this study was the addition of joint accelerations to the vector of DVs. This required different constraints and a higher number of decision variables, but overall, it was not significantly harder to implement.

5.3.2 Performance

5.3.2.1 Compass Walker

The figures in section 5.1 are fairly indicative of the actual performance comparisons between factors. To begin, in reference to Figures 5.3 and 5.4, Tables 5.1 and 5.2 below contain the p-values corresponding to the time-related factors. Table 5.1 uses all the data found with a randomized IG, while Table 5.2 leaves out the runs similar to the "alternative" gait. The average CPU time per iteration is strongly dependent on the collocation method and differentiation method employed. In fact, the collocation method makes a significant difference for all time-related measures. From the plots of the related data, it is clear that TPZD is significantly faster than H-S. This is likely due to the decreased number of constraints and decision variables, as well as the relative simplicity of the collocation constraints and cost function. The block of ones in the Jacobian also could have been a factor. Additionally, ND is faster than SD when the overall CPU time is considered. Figure 5.12 shows this clearly. This is a bit unexpected, as each numerical differentiation requires two evaluations of the constraints. However, with the simplicity of the compass walker, it is possible that these two evaluations together are still less complex than the evaluation of the symbolic Jacobian expressions.

Factor	CPU Time	Iterations	CPU/Iter
Collocation Method	0.000 [*]	0.000 [*]	0.000 [*]
Jacobian Differentiation Method	0.000 [*]	0.050 [*]	0.000 [*]
Interaction	0.048 [*]	0.017 [*]	0.000 [*]

Table 5.1: p-values of time-related data, all compass randomized IG data. [*] means $p \leq 0.05$.

Factor	CPU Time	Iterations	CPU/Iter
Collocation Method	0.000 [*]	0.000 [*]	0.000 [*]
Jacobian Differentiation Method	0.000 [*]	0.073	0.000 [*]
Interaction	0.063	0.054	0.000 [*]

Table 5.2: p-values of compass time-related data, excluding "alternative" gait data. [*] means $p \leq 0.05$.

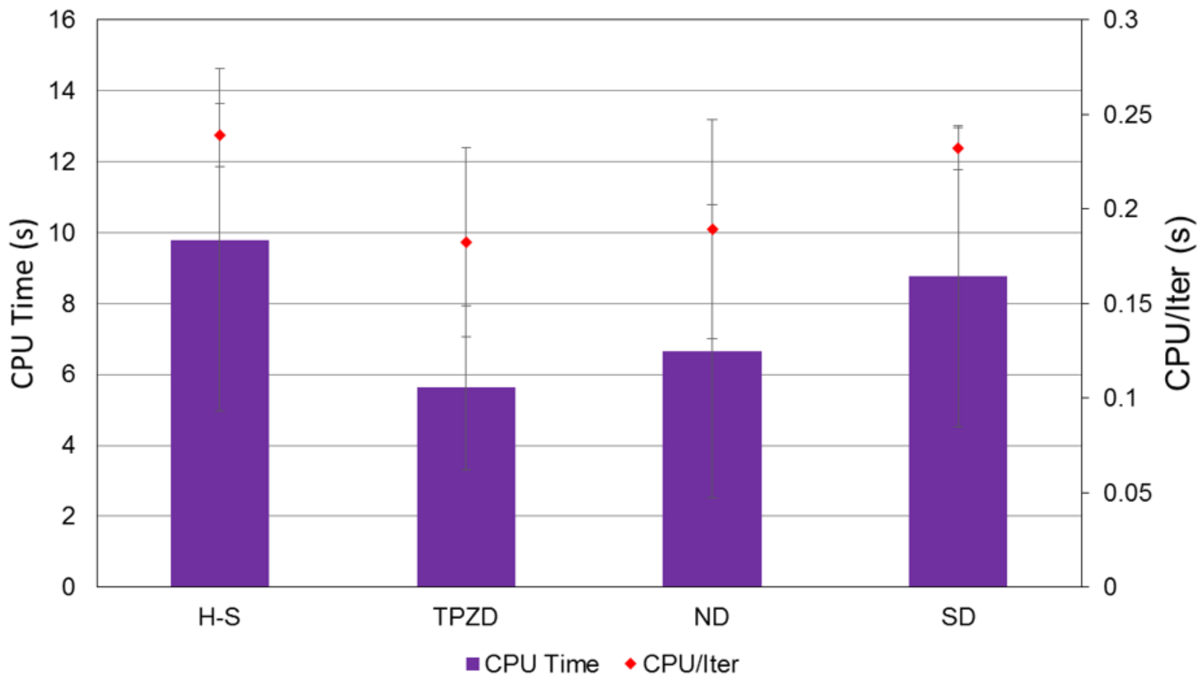


Figure 5.12: Compass gait time-related results by factor.

The data in the tables above also reveal the existence of interactions between factors, particularly for the average CPU time per iteration. Figure 5.13 below was generated by Minitab, and it shows interesting trends. When H-S is used, it is slightly more computationally efficient to calculate the Jacobian symbolically than numerically. However, when TPZD is used, it is significantly more efficient to use numerical differentiation than symbolic differentiation.

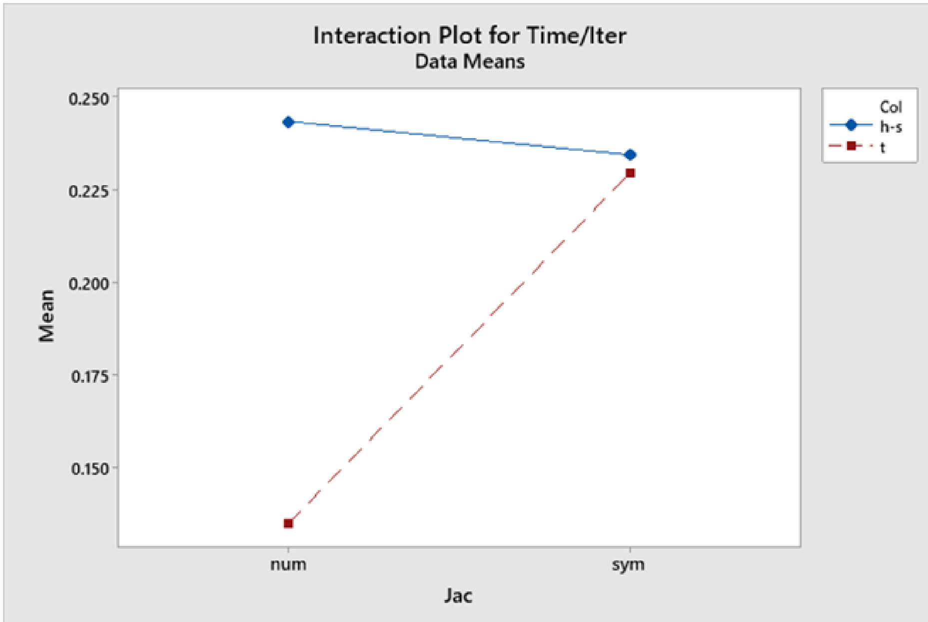


Figure 5.13: Minitab plot[2] of interaction between collocation method and Jacobian calculation method for CPU/Iter.

With regards to the objective value and period of the gait, the only significant factor was the collocation method. The results for the comparisons to the baseline gait were a bit more interesting, although they still did not yield too many conclusive results. The p-values for this data are given in Tables 5.3 and 5.4. These data, in combination with Figure 5.5, suggest that when all runs are going towards an expected baseline gait, H-S is more accurate than TPZD. This is reasonable when compared to Pardo et al.'s results [23], and it makes sense theoretically when considering that H-S uses a higher-order approximation.

Factor	q1 Error	q1d Error	q2 Error	q2d Error
Collocation Method	0.798	0.921	0.738	0.679
Jacobian Method	0.33	0.334	0.329	0.329
Interaction	0.862	0.899	0.848	0.841

Table 5.3: p-values of comparison to baseline, all compass randomized IG data. [*] means $p \leq 0.05$.

Factor	q1 Error	q1d Error	q2 Error	q2d Error
Collocation Method	0.000 [*]	0.000 [*]	0.000 [*]	0.000 [*]
Jacobian Method	0.309	0.623	0.457	0.990
Interaction	0.307	0.607	0.425	0.990

Table 5.4: p-values of comparison to baseline, excluding "alternative" compass gait data. [*] means $p \leq 0.05$.

5.3.2.2 Five-Link Biped

The different settings used for the five-link walker also had a significant effect on the computational measures. These are outlined in Tables 5.5 and 5.6, using the results for both eleven and twenty-one collocation points, which are separated with commas. It should be noted that a completely separate ANOVA was performed to look at the inclusion of accelerations in the DVs. From these tables, with twenty-one points, the Jacobian differentiation method did not have a significant effect on the CPU time, and the presence of accelerations in the decision variables was also not critical.

Factor	CPU Time	Iterations	CPU/Iter
Acceleration Calculation	0.000 [*], 0.000 [*]	0.000 [*], 0.050 [*]	0.000 [*], 0.000 [*]
Jacobian Differentiation	0.000 [*], 0.778	0.000 [*], 0.025 [*]	0.000 [*], 0.000 [*]
DVs	0.000 [*], 0.005 [*]	0.000 [*], 0.000 [*]	0.000 [*], 0.000 [*]

Table 5.5: p-values of time-related data, all 5-link randomized IG data with (11 collocation points, 21 collocation points). [*] means $p \leq 0.05$.

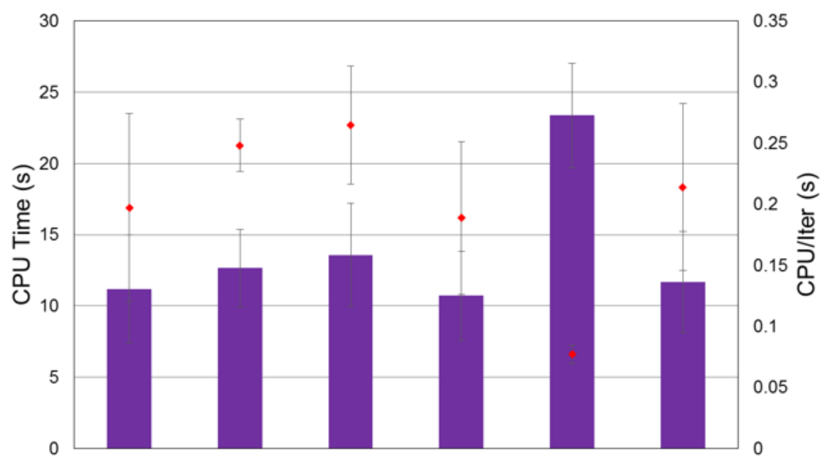
Factor	CPU Time	Iterations	CPU/Iter
Acceleration Calculation	0.000 [*], 0.000 [*]	0.000 [*], 0.149	0.000 [*], 0.000 [*]
Jacobian Differentiation	0.000 [*], 0.556	0.000 [*], 0.049 [*]	0.000 [*], 0.000 [*]
DVs	0.000 [*], 0.054	0.000 [*], 0.000 [*]	0.000 [*], 0.000 [*]

Table 5.6: p-values of time-related data, excluding "alternative" 5-link gait data with (11 collocation points, 21 collocation points). [*] means $p \leq 0.05$.

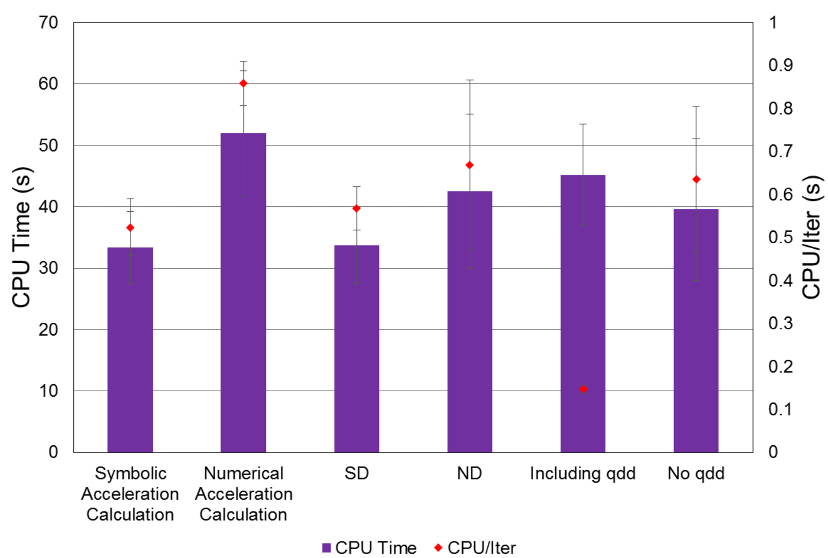
Figure 5.14 shows the factor-by-factor time data graphically. According to this figure, symbolically evaluating the joint accelerations was quicker than repeatedly calculating them numerically. Even though the symbolic acceleration expression is quite complicated, it is not too unreasonable to consider that the repeated solution of a linear system of equations is still more costly.

The relative computational times due to the Jacobian differentiation method are less clear. When twenty-one collocation points were used, numerical differentiation was slower, which, as mentioned in the previous section, aligns with intuition. However, ND was faster when only eleven collocation points were used, suggesting that there may be some threshold for the number of constraint evaluations performed before SD becomes more efficient. All this being said, based on the p-values, it is more likely that the slower ND time had more to do with the acceleration computation method used than ND itself.

The results related to the presence of acceleration in the DVs were quite interesting as well. The time required per iteration was significantly faster when they were included, due to the increased sparsity of the problem and the elimination of the need to use a separate function to evaluate the accelerations before including them in constraints. However, the number of iterations required, and subsequently the overall CPU time, were higher when they were included. This could be due to the fact that there were more constraints, as the satisfaction of dynamics and the collocation constraints were separated.



(a) 11 collocation points.



(b) 21 collocation points.

Figure 5.14: 5-link time-related results by factor.

There were very few significant differences for the data comparing the individual generated gaits to the baseline gait. In fact, several ANOVA results were unable to be calculated once the runs similar to the "alternative" gaits were eliminated; this was because the data points for all versions of a factor were the same to Minitab's precision. For eleven collocation points, this issue was the case for all objective, period, q1, and q2 analyses, as well as the acceleration calculation and Jacobian differentiation analyses for q5. This event occurred for slightly different analyses when twenty-one collocation points were used, but the overall phenomenon persisted. For both eleven and twenty-one collocation points, some DV-related analyses were found to be significant when "alternative" gaits were removed. No small part of this was the spread of the results when accelerations were included in the DVs, as evidenced by Figure 5.15.

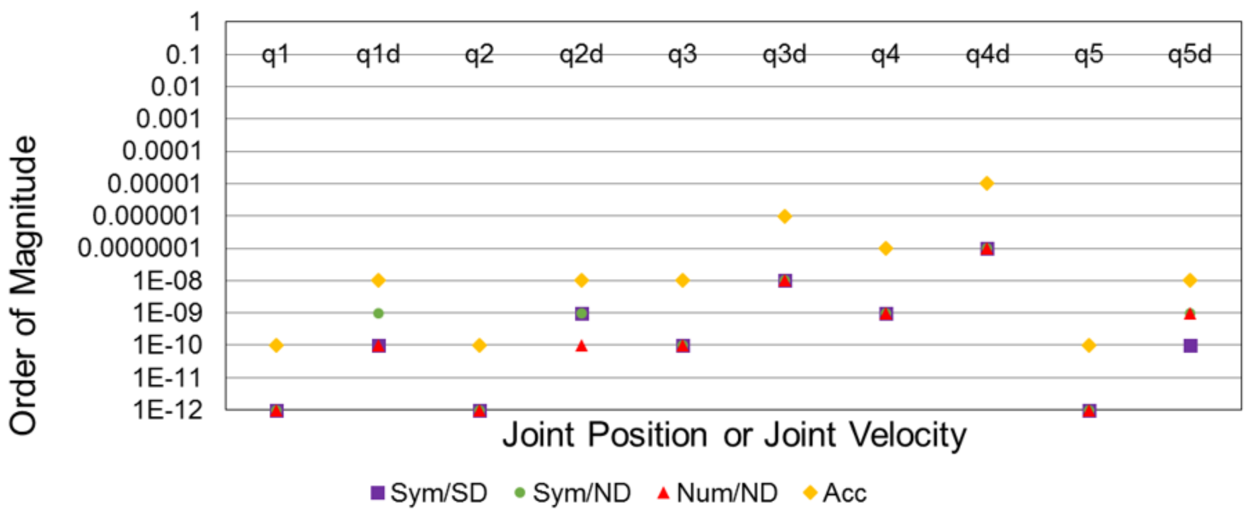


Figure 5.15: Order of magnitude of standard deviation for error with respect to baseline for gaits similar to baseline with 11 collocation points.

5.3.3 Analysis of Tradeoffs

The above discussion reveals that some tradeoffs might have to be made in order to recommend the "best" optimization settings. For the compass walker, the primary tradeoff to consider is the increased accuracy of the H-S collocation method (as evidenced by runs similar to the baseline gait) and its far slower speed. Also of importance is the interaction between the collocation method and Jacobian calculation method, which was given in Figure 5.13. If ND is used, as is recommended based on the CPU time, time per iteration, and ease of implementation, then based on the interactions, TPZD should be used. This is the author's recommendation for the compass walker optimization, particularly given the still reasonable "accuracy" of TPZD. However, if an extremely high level of "accuracy" is desired, then H-S should still be used.

The five-link situation is made more complicated by the fact that two completely different recommendations must be made. On one hand, it must be determined whether the joint accelerations should be included in the decision variables. Secondly, in the event that they are not included, the methods of calculating the joint accelerations and Jacobian must be determined.

The former is the more complex situation, as including the decision variables results in a quicker CPU time per iteration but a slower overall CPU time. Additionally, including the decision variables seems to result in a higher "robustness," as no "alternative" gaits were calculated with that method; however, when the alternative gaits are removed, this method has the least consistency in deviation from the baseline gait. All things considered, it seems prudent to include the accelerations in the decision variables due to the likely "robustness" and efficiency of the method.

The latter case is a bit more straightforward. The joint accelerations should be evaluated symbolically due to the faster CPU time. The collocation constraints should be evaluated numerically for three main reasons. First, the implementation of ND is significantly easier; secondly, ND is highly likely to scale to higher dimensions more easily than SD. Lastly, ND was shown to be faster in the case of eleven collocation points; however, since this was not the case with the more precise (yet statistically less significant) twenty-one point runs, this is might not be as strong a reason.

5.4 Limitations of Study

There were a few potential limitations to this study. First, the compass walker was only studied on a downslope. This biped is able to passively walk down a slope, so determining the optimal control on flat ground, where control is actually needed, may have been more illustrative. Additionally, The calculation of Jacobian elements corresponding to the midpoint torque constraints, as well as the construction of their section of the Jacobian structure, could have been more precise.

With regards to the five-link walker, it is possible that there could be a more "optimal" gait than the baseline used, as it was generated with the same NLP as the other gaits. Regardless, though, using this particular baseline was useful for looking at the consistency of the gait generation. Also, completing more runs or trying a greater number of collocation points could have affected the statistical results. There were a few differences between the results found for eleven and twenty-one collocation points. Lastly, it could have been beneficial to also test a setting with numerical acceleration calculation and symbolic Jacobian differentiation.

6. CONCLUSIONS

6.1 Conclusion

Nonlinear programs were developed in order to find optimal trajectories and control signals for two different bipedal walking models, the 2-DOF compass walker and the 5-DOF kneed biped with a torso. Through the repeated solution of these NLPs with randomized initial guesses, some factors were shown to have a larger effect on results such as CPU time and deviation from a baseline gait than others.

For the compass walker, it was recommended to use numerical differentiation due to its faster speed. It was also recommended to use trapezoidal collocation for this reason, although it is at the (slight) expense of accuracy.

For the five link biped, it was recommended to include joint accelerations in the NLP's decision variables due to a potential "robustness" in gait convergence and an extremely rapid computational time required per iteration. However, this does result in a longer overall computational time, as more iterations are required. In the event that the accelerations are not included in the DVs, generating a symbolic function which directly determines the joint accelerations is computationally more efficient than numerically calculating the accelerations each iteration. Also, although the relative computational speeds of numerical and symbolic differentiation of the gradient and Jacobian are somewhat inconclusive, it is still recommended to use numerical differentiation due to its ease of implementation for problems of many sizes.

6.2 Future Work

There is significant room for future work related to this topic. In this study, only two forms each of differentiation and collocation were tested. It would be valuable to see how the performance of automatic differentiation–which has the accuracy of symbolic differentiation without the need to store such large expressions [24]–would compare to symbolic and numerical differentiation. Likewise, there are different ways to discretize these continuous-time systems which the author has not yet tested and compared. One last comparison which would also prove useful in the future would be to compare the efficiency of different programming languages in these optimizations. Only MATLAB was used here, but languages like Julia or Python might prove to be more successful.

REFERENCES

[1] Y. C. Lin and M. G. Pandy, “Three-dimensional data-tracking dynamic optimization simulations of human locomotion generated by direct collocation,” *Journal of Biomechanics*, vol. 59, pp. 1–8, 2017.

[2] A. Falisse, G. Serrancolí, C. L. Dembia, J. Gillis, and F. DeGroote, “Algorithmic differentiation improves the computational efficiency of OpenSim-based trajectory optimization of human movement,” *PLoS ONE*, vol. 14, no. 10, pp. 1–19, 2019.

[3] B. Kaddar, Y. Aoustin, and C. Chevallereau, “Arm swing effects on walking bipedal gaits composed of impact, single and double support phases,” *Robotics and Autonomous Systems*, vol. 66, pp. 104–115, 2015.

[4] X. Mu and Q. Wu, “A Complete Dynamic Model of Five-Link Bipedal Walking,” *Proceedings of the American Control Conference*, vol. 6, pp. 4926–4931, 2003.

[5] M. Kelly, “An introduction to trajectory optimization: How to do your own direct collocation,” *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[6] L. B. Freidovich, U. Mettin, A. S. Shiriaev, and M. W. Spong, “A passive 2-DOF walker: Hunting for gaits using virtual holonomic constraints,” in *Proceedings of the 47th IEEE Conference on Decision and Control*, (Cancun, Mexico), pp. 1202–1208, 2008.

[7] Z. Liu and Y. Tian, “Some Control Strategy on the Compass Gait Biped,” in *IMACS Multiconference on "Computational Engineering in Systems Applications"*, (Beijing, China), pp. 1712–1718, 2006.

[8] S. Krafes, Z. Chalh, and A. Saka, “A Review on the Control of Second Order Underactuated Mechanical Systems,” *Complexity*, vol. 2018, 2018.

[9] S. Gupta and A. Kumar, “A brief review of dynamics and control of underactuated biped robots,” *Advanced Robotics*, vol. 31, no. 12, pp. 607–623, 2017.

[10] R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, H. B. Brown, D. Mcmordie, U. Saranli, R. Full, and D. E. Koditschek, "RHex: A biologically inspired hexapod runner," *Autonomous Robots*, vol. 11, no. 3, pp. 207–213, 2001.

[11] K. Chao, *Bipedal Walking Analysis, Control, and Applications Towards Human-Like Behavior*. PhD thesis, Texas A&M University, 2019.

[12] H. Dai and R. Tedrake, "Optimizing robust limit cycles for legged locomotion on unknown terrain," in *Proceedings of the IEEE Conference on Decision and Control*, (Maui, USA), pp. 1207–1213, 2012.

[13] V. Paredes, W. Hong, S. Patrick, and P. Hur, "Upslope walking with transfemoral prosthesis using optimization based spline generation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Daejeon, Korea), pp. 3204–3211, IEEE, 2016.

[14] J. Kumar, N. Kumar, D. Pankaj, and A. Kumar, "Implementation of real time control algorithm for gait assistive exoskeleton devices for stroke survivors," in *Proceedings- 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*, (Nagpur, India), pp. 271–275, IEEE, 2014.

[15] A. C. Hildebrandt, R. Wittmann, F. Sygulla, D. Wahrmann, D. Rixen, and T. Buschmann, "Versatile and robust bipedal walking in unknown environments: real-time collision avoidance and disturbance rejection," *Autonomous Robots*, vol. 43, no. 8, pp. 1957–1976, 2019.

[16] K. Chao and P. Hur, "A step towards generating human-like walking gait via trajectory optimization through contact for a bipedal robot with one-sided springs on toes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Vancouver, Canada), pp. 4848–4853, 2017.

[17] R. W. Sargent, "Optimal control," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 361–371, 2000.

[18] M. P. Kelly, "Transcription Methods for Trajectory Optimization: A beginners tutorial," tech. rep., 2017. arXiv:1707.00284.

[19] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, “Dynamic Humanoid Locomotion: A Scalable Formulation for HZD Gait Optimization,” *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 370–387, 2018.

[20] E. Bertolazzi, “mexIPOPT,” 2018. Available: https://github.com/ebertolazzi/mexIPOPT.

[21] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, pp. 25–57, 2006.

[22] Y. Nie and E. C. Kerrigan, “How Should Rate Constraints be Implemented in Nonlinear Optimal Control Solvers?,” *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 362–367, 2018.

[23] D. Pardo, L. Moller, M. Neunert, A. W. Winkler, and J. Buchli, “Evaluating Direct Transcription and Nonlinear Optimization Methods for Robot Motion Planning,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 946–953, 2016.

[24] A. Güne, G. Baydin, B. A. Pearlmutter, and J. M. Siskind, “Automatic Differentiation in Machine Learning: a Survey,” *Journal of Machine Learning Research*, vol. 18, pp. 1–43, 2018.

[25] A. Dürrbaum, W. Klier, and H. Hahn, “Comparison of automatic and symbolic differentiation in mathematical modeling and computer simulation of rigid-body systems,” *Multibody System Dynamics*, vol. 7, no. 4, pp. 331–355, 2002.

[26] M. Giftthaler, M. Neunert, M. Stäuble, M. Frigerio, C. Semini, and J. Buchli, “Automatic differentiation of rigid body dynamics for optimal control and estimation,” *Advanced Robotics*, vol. 31, no. 22, pp. 1225–1237, 2017.

[27] J. Ismail and S. Liu, “Efficient Planning of Optimal Trajectory for a Furuta Double Pendulum Using Discrete Mechanics and Optimal Control,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10456–10461, 2017.

[28] Z. Manchester, N. Doshi, R. J. Wood, and S. Kuindersma, "Contact-implicit trajectory optimization using variational integrators," *International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1463–1476, 2019.

[29] E. R. Westervelt, J. W. Grizzle, C. Chevellereau, J. H. Choi, and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, 2007.

[30] A. Goswami, B. Thuilot, and B. Espiau, "Compass-Like Biped Robot Part I : Stability and Bifurcation of Passive Gaits," tech. rep., INRIA, RR-2996, 1996.

[31] R. F. Chandler, C. E. Clauser, J. T. McConville, H. M. Reynolds, and J. W. Young, "Investigation of inertial properties of the human body," tech. rep., Aerospace Medical Research Laboratory, 1975.

[32] Y. Kawajir, C. D. Laird, S. Vigerske, and A. Waechter, "Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT," 2015.