DEVELOPMENT OF A MODEL TO SIMULATE THE SINGLE-LIMB STANCE PHASE OF WALKING TO DETERMINE THE LIMITS OF STABILITY

by

CLIFFORD LEE HANCOCK

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN BIOMEDICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2013

Acknowledgements

I would like to express my sincerest gratitude to the three members of my thesis committee: Dr. Ricard, Dr. Chuong, and Dr. Bowling. They have all been very supportive of my research interests and goals. In particular, I would like to thank Dr. Ricard for helping me decide on a topic and sticking with me through the entire process. Without his guidance, none of my research would have ever come to fruition or have even been possible. In fact, not only his thesis advice, but also his career and life guidance has shaped my future, and I cannot thank him enough for everything that he has provided. I certainly look forward to the next step in my career, and am tremendously grateful for his guidance.

I also must genuinely thank the bioengineering department faculty for allowing me the opportunity to become a part of their terrific program which allowed me to continue to study my interests in the field of biomechanics.

April 15, 2013

Abstract

DEVELOPMENT OF A MODEL TO SIMULATE THE SINGLE-LIMB STANCE PHASE OF WALKING TO DETERMINE THE LIMITS OF STABILITY

Clifford Lee Hancock, M.S.

The University of Texas at Arlington, 2013

Supervising Professor: Mark D. Ricard

Falls during walking are the primary cause of injuries for older persons and present a high associated cost to society due to the resulting treatment. Hence, the control of balance while moving is an essential issue especially amongst the elderly population. Better understanding of why and how these falls occur could lead to better fall prevention methods, therapies, or devices. These changes could effectively decrease overall emergency room visits, create more room in hospitals for other demands, and decrease medical costs for fall-related disabilities and deaths amongst patients overall. This study attempted to develop an inverted pendulum model of the single-leg support phase of walking within the sagittal plane. Upon validating the model's use, the model was utilized to predict the limits of stability during this walking phase for various conditions including: eyes open with no obstacle, eyes open with an obstacle, and eyes closed without an obstacle. Additionally, the region between these limits of stability, coined the area

of stability, was compared between elderly and young subjects and across the three conditions to determine whether their respective values significantly differed.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Background

Various types of falls account for many trips to the emergency room across the world, but falls can be especially damaging within the elderly population. The elderly tend to have weaker, less dense and more brittle bones that lead to easier fractures. Among a test group of elderly community dwellers, falls have been shown to occur to 30% of community dwellers (≥65 years of age) at least once per year, and to 15% two or more times per year. This elderly test group "reported physical injury (68.1%), major injury (5.9%), health service use (23.5%), treatment (17.2%), decline in functional status (35.3%), and social (16.7%) and physical activities (15.2%)" (Stel et al., 2004). Hence, the control of balance while moving is an essential issue especially amongst the elderly population. Better understanding of why and how these falls occur could lead to better fall prevention methods, therapies, or devices. These changes could effectively decrease overall emergency room visits, create more room in hospitals for other demands, and decrease medical costs for fall-related disabilities and deaths amongst patients overall.

The elderly not only typically have decreased bone density, but they also sometimes exhibit muscle, nervous, sensory, and cognitive degeneration (Chiu et al., In Press). These extra complications can help lead to imbalance especially

following an unexpected perturbation. Nerve damage, such as myelin sheath degeneration in various disorders, leads to slower electrical conduction velocities in both afferent sensation and efferent motor control signals. If a person is forced to shift his body unexpectedly due to a trip or a push, the nerve conduction speeds become very important so that the person can respond in a timely manner. In addition, these nerves must communicate with either the spinal column or the brain. If the person exhibits cognitive degeneration of some kind (especially degeneration of motor cortical regions), the brain may take too long to issue a response to the perturbation or simply fail to respond at all leading to instability. Assuming the brain responds and the signal is sent via the nerves in time, various muscles throughout the body will attempt to regain bodily balance. If these muscles fail to produce enough force due to weakened muscle tone or the rate of force generation is too slow overall, a fall will likely ensue. Due to all of these issues, many elderly typically exhibit "slower walking speeds, shorter steps, increased step width, decreased muscle strength and reduced range of motion in the lower extremities" (Chiu et al., In Press).

Assuming these elderly subjects possess no lower extremity or back pain and lack cognitive, muscular, or nervous disorders, this study attempted to develop a model to predict when instability would ensue. Upon validating the model's use, the model was then utilized to analyze differences in young and elderly subjects while traversing a walkway. Since many falls occur during the

night or morning hours, when vision is obscured due to low light levels, the subjects were also asked to traverse the walkway with their eyes closed. Subsequently, the model was then utilized to analyze the same subjects while crossing over a small obstacle. Assuming the model was valid, the model should have been able to predict when instability would occur in each of these situations given certain initial and boundary conditions.

1.2 Purpose

The purpose of this study was to first develop a model to predict under what circumstances instability would ensue. Subsequently, kinematic data was collected from both elderly and young subjects traversing a pathway with eyes open, closed, and while stepping over an obstacle. This data, along with force plate readings for each trial, helped validate the model which was then used to further analyze the differences in stability regions between young and elderly participants.

1.3 Definition of Terms

1) Fall: This study defined a fall as anything that caused a change in base of support (Patton et al., 1999). This definition includes actions such as side-stepping, leaping, and stumbling.

2) Base of Support: The base of support was defined as the area under the subjects' feet that was used to maintain balance. In various other studies, this area extends between both feet. However, this study only dealt with the single-leg

support phase of gait so this larger area was not considered. Assuming toe and heel markers were situated along the center line of the foot from overhead, this area was approximated as a rectangle with corners that extended half the ankle width medial-laterally in both directions from each marker [Figure 1-1].

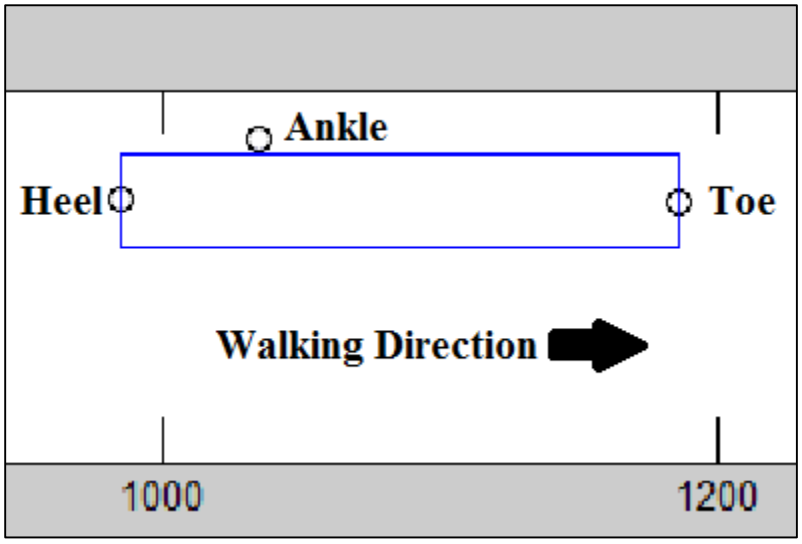

Figure 1-1 The Base of Support (BoS) is outlined in blue and displayed from overhead. The heel, ankle, and toe markers are shown as black circles from left to right respectively.

However, since this study did not attempt to model in the transverse plane and only in the sagittal plane, the heel and toe marker x-direction (walking direction) positions were the only points of interest as far as the base of support was concerned.

3) Center of Pressure: The center of pressure is the location within or near the base of support where the ground reaction force vector is located. This location can be tracked over time by the use of a force plate [Figure 1-2].

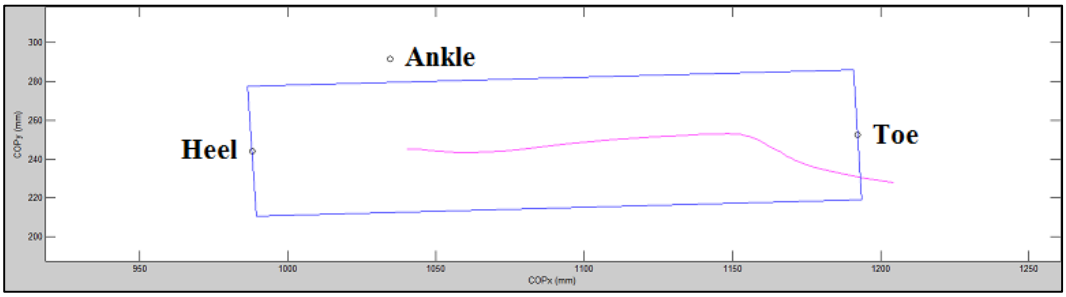

Figure 1-2 The Center of Pressure (CoP) is shown in pink during this single-leg support phase example. As the subject lifts his trailing limb off the ground, the CoP tracks from left to right until the trailing limb turns into the leading limb and makes contact with the floor. The BoS is outlined in blue and the heel, ankle, and toe markers are shown as black circles from left to right respectively.

4) Center of Mass: The center of mass is the location within the human body where the mass from all the individual segments balance. This location is typically in proximity to the navel but can vary from person to person depending upon individual subject segment shapes, masses, and lengths.

5) Force Plate: There are multiple types of force plates that are in use around the world, but within this paper, the use of the words “force plate” will refer to the strain gauge variety. These force plates contain a series of strain gauges which deform when an external force is applied to their exteriors. The

deformation of the individual strain gauges causes changes in electrical resistance within each gauge. These changes in electrical resistance can be measured, turned into a voltage, and ultimately outputted as a force value. By combining a series of strain gauges, exact force locations and directions can be determined by their combined deformations over time.

6) Kinematics: The field of dynamics is typically split into two parts: kinematics and kinetics. Kinematics refers to the "study of motion without reference to the forces which cause motion" (Meriam and Kraige, 2009). Hence, motion of an object, such as a human, can be described simply by discussion of its position, velocity, and acceleration.

7) Kinetics: Kinetics is the other section of dynamics "which relates the action of forces on bodies to their resulting motions" (Meriam and Kraige, 2009). Kinetics in human locomotion may take into account internally applied forces, such as muscle force generation, or externally applied forces, such as a push, which both result in bodily movement.

1.4 Delimitations

The delimitations of the study include the analysis of 7 young subjects (21±2.8 years) and 7 elderly subjects (71±6.4 years). The two age groups included both male and female participants to more accurately assess the general population. In order to focus solely on the bodily segment kinematics, all subjects were healthy and free of musculoskeletal, neurodegenerative, and cognitive

disorders. In addition, they self-reported that they were free of hip, knee, or back pain that required treatment within two months immediately preceding testing. No subjects had a history of back or lower limb surgery. The subjects were asked to walk barefoot to help ensure differences in footwear did not alter the force plate data or gait in any way. All subjects were asked to complete the same number and types of trials and were given practice sessions in order to hit both force plates with a natural stride-length and speed. Participants were required to repeat trials if they decelerated just prior to hitting the force plates or if they simply missed the force plate in order to achieve the most accurate data possible.

1.5 Assumptions

Since this study involved human subjects, several important assumptions were made. The first assumption was that the subjects were truthful about being free of any problematic disorders, and that they were free of back, hip, and lower extremity issues. Secondly, despite giving the subjects multiple practice sessions, the subjects may have altered their gait from normal while under examination. However, we had to assume that the subjects were walking as normally as possible. In addition, as with any test trial involving reflective markers, the assumption was made that the markers stayed firmly attached and static on each subject for each trial. However, skin is an elastic organ and even the slightest forces can cause skin to shift. Hence, walking likely caused the skin and markers to move; but since walking is a relatively calm motion, it can be assumed that the

skin moved very little and the reflective markers stayed relatively still. As a third assumption, the inverted pendulum model utilized by this study was very simplistic. When falling, subjects sometimes attempt to restore balance by bending their hips or shifting their arms. These motions shift the CoM (which was included within the model) so it was assumed that the model was able to account for such complexities. Lastly, the equation that defined stability, which was utilized to determine the limits of stability, was borrowed from a study that made several important assumptions. The stability equation was derived assuming that the CoM pendulum always stayed near vertical, that the length of the pendulum did not change, and that the CoP location remained constant. While standing still, the CoM pendulum likely remains near vertical but during dynamic situations such as walking, the pendulum rotates and does not remain vertical. Since this study's model utilized this vertical assumption, the results may have been somewhat distorted. In addition, the stability equation assumed the CoM pendulum length remained constant. This assumption was not as detrimental to the results since the pendulum length did not change very drastically throughout the motion. The last stability equation assumption was that the CoP location remained constant near the balls of the feet. In this study's model, the CoP was allowed to change location based upon actual subject force plate data. However, the CoP remained near the ball of the feet for much of the single-leg support phase so this assumption seemed reasonable.

1.6 Limitations

The model and the study in general had some limitations which had to be addressed. The human body is a very complex system and during locomotion there are many moving bodily segments. The model attempted to simplify the body down into a single segment with a single point mass at one end rotating about an ankle joint. During attempted balance recovery, many muscles may be activated, including ones in the arms to attempt to shift the overall body center of mass. Thus, this model greatly simplified the actual human body. However, the model only attempted to depict the single-leg stance phase of walking and thus avoided many of the complexities of gait.

Along with the model, the subject trials also had various limitations. In each of the subject trials, the walking speed was not well controlled across participants and some of the elderly subjects, in particular, slowed their gait during obstacle crossing in order to ensure the maintenance of balance. In addition, all of the subjects were from the Arlington, TX community and seemed fairly representative of the general population, but people from different regions would have possibly generated a better representation. However, for the purposes of developing a model, the group of subjects chosen was perfectly adequate.

In addition, while the force plate and camera data were filtered by intelligent means, no filter is perfect. Filters nearly always eliminate some of the relevant data while attempting to eliminate extraneous noise. Therefore, the data

may have been slightly altered due to the filtering methods, but this error was hopefully minimized by the respective types of filters chosen for both applications.

Chapter 2

Review of Literature

2.1 Gait in the Elderly

Falls during walking are the primary cause of injuries for older persons (Berg et al., 1997; Ihlen et al., 2012; Luukinen et al., 1994; Niino et al., 2000; Sterling et al., 2001). Due to their high rate of incidence, falls present a high associated cost to society due to the resulting treatment. Much research has been conducted to better understand gait stability to help identify people at risk of falling. For instance, research has shown that older subjects tend to exhibit a decreased recovery performance compared to their younger counterparts when met with an unexpected perturbation such as tripping (Berg et al., 1997; Grabiner et al., 2005; Karamanidis and Arampatzis, 2007; Pijnappels et al., 2005). This decrease in responsiveness and effectiveness can be attributed to degeneration of the motor cortical regions and neurotransmitter systems (Seidler et al., 2010), a slower force generation within muscles (Vandervoort and Mccomas, 1986), tendon stiffness, and decreased muscle strength in general (Fukagawa and Schultz, 1995; Grabiner et al., 2005; Karamanidis et al., 2008). Aging may also cause different plantar loadings due to changes in foot characteristics such as flatter and more pronated feet. These changes could lead to decreased forces on the heel and lateral forefoot, but also lead to longer contact times at the heel, midfoot, and metatarsophalangeal joints (Scott et al., 2007). Additionally, spatial

characteristics, such as the CoP distribution across the foot during gait, and temporal characteristics, such as the CoP velocity at various gait intervals, will also change due to changes in foot shape. In particular, the elderly have been shown to exhibit faster CoP velocity during heel strike, slower CoP velocity during mid-stance, and a more significant medial CoP progression curve than their younger counterparts (Figure 2-1, (Chiu et al., In Press)).

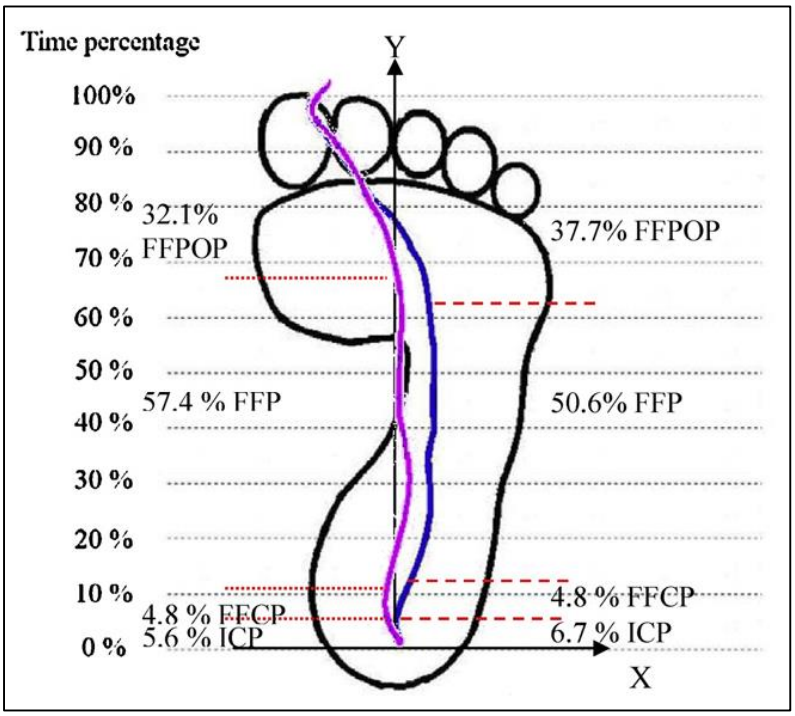

Figure 2-1 CoP progression from heel strike to toe off with time percentages for old (left) and young (right) adults. ICP, initial contact phase, FFCP, forefoot contact phase, FFP, foot flat phase, FFPOP, forefoot push off phase, (Chiu et al., In Press).

The elderly also typically exhibit slower preferred walking speeds with shorter step lengths which may also contribute to an increased risk of falls (Laufer, 2005). Various authors have shown that a slower gait is directly linked to an increased risk of falls (Abella et al., 2005; Cromwell and Newton, 2004). However, slower gait speeds have also been shown to increase local stability (Dingwell and Marin, 2006; England and Granata, 2007; Li et al., 2005), and shorter step lengths have been shown to be beneficial when on slippery surfaces (Moyer et al., 2006). Hence, these findings are very controversial and it is rather unclear whether or not decreased walking speeds or decreased step lengths actually lead to falls. "Shorter steps may in fact provide greater stability, but this benefit may be offset by the accompanying slowness in gait speed" (Bhatt et al., 2011).

While older persons may exhibit some or all of these drawbacks, they have likely experienced a large number of unexpected perturbations throughout their lifetime simply due to their age. Since people possess the ability to learn from prior experiences, the elderly often can improve their dynamic stability and counteract these drawbacks associated with old age via predictive and adaptive adjustments (Marigold and Patla, 2002; Pai et al., 2003; Van Der Linden et al., 2007). In fact, repetitive simulations of perturbations while walking have been shown to improve the reactive behavior in both young and elderly subjects (Bierbaum et al., 2011). However, these predictive adjustments assume that the perturbation is somewhat expected. If the perturbation is truly unexpected, as is

often the case in real life scenarios, the person must utilize purely reactive adjustments which is based upon information received during the movement. Since age often diminishes the capacity of the proprioceptive system and affects the afferent sensory system, less information is ultimately transmitted to the spinal column about the disturbance (Bierbaum et al., 2011; Hurley et al., 1998; Lord et al., 1996; Patel et al., 2009).

In addition to physical changes within the human body, people that have experienced falls may actually alter their normal gait due to mental fears of future falls. This fear may cause them to have a slower gait or to have abnormal step width variability. Extreme step width variability can be another cause for imbalance and in one study, subjects with "either low or high step width variability were 4.38 times more likely to have fallen in the past year" (Barry et al., 2005). Thus, there are many reasons that falls take place and many researchers have attempted to find commonalities between each type of fall. The ultimate goal is to better anticipate under what circumstances a fall will occur to hopefully prevent future falls from transpiring.

2.2 Balance

While standing still, the ability to maintain balance depends upon the vertical projection of the body's center of mass (CoM) with respect to the base of support and the CoM movement speed (Patton et al., 1999). The base of support is defined as the region in which the center of pressure can be generated (Hof et al.,

2005). During standing, if the CoM is too far outside this base of support, the person will fall or be forced to take a step. Similarly, if the CoM is moving too quickly in any given direction, such as when someone is pushed from behind while standing, the person will also fall. While maintaining stability during standing is an area of great concern, many falls occur during locomotion. While moving, the CoM repeatedly shifts beyond and behind the base of support. Therefore, unlike during standing, a person can still maintain balance while moving when the CoM greatly exceeds the base of support. Thus, the control of CoM velocity and momentum plays a much larger role in maintaining balance during locomotion.

In particular, imbalance and tripping, due to the failure to maintain adequate CoM control and velocity during obstacle crossing, leads to "two of the most common causes of falls in the elderly" (Chou et al., 2003). Successful obstacle-clearing attempts require adequate foot clearance of both the leading and trailing legs, the ability to exhibit a longer duration single-leg stance while the other leg is crossing the obstacle, and correct body segment coordination. If a subject fails to complete any of these steps, their CoM will move too quickly in one direction (such as a sideways) and the subject will likely fall.

2.3 Modeling

In several past papers, inverted pendulums have been utilized to model the stance leg in balance dynamics (Gurfinkel, 1973; Hemami and Golliday Jr, 1977;

Lee and Patton, 1997; Winter, 1995). In these situations, the body mass is assumed to be a point mass which resides at the center of mass of the entire body. The pendulum length extends from this point mass down to a pivot location at the ankle. This type of two-dimensional model greatly simplifies the overall motion of the body which consists of many segments with many individual segment masses. Each of these segments rotates at different angular velocities and accelerations during locomotion which makes the physics and mathematics fairly complex. In order to avoid such complexities, these two-dimensional models only have one artificial segment, one point mass, one angular velocity and one angular acceleration.

Simple inverted pendulums obey Euler's equation:

$$\sum \boldsymbol{M} = I\ddot{\theta} \tag{1}$$

where **M** includes the moments about the ankle rotation point, $I = ml^2$ if l is assumed to be a fixed length, and $\ddot{\theta}$ is the angular acceleration of the pendulum arm. If one wanted to convert from polar coordinates (r, θ) into a Cartesian frame (x, y), the conversion equations would need to be utilized:

$$x = r\cos\theta \tag{2}$$

$$\dot{x} = \dot{r}\cos\theta - \dot{\theta}r\sin\theta \tag{3}$$

$$\ddot{x} = \ddot{r}\cos\theta - 2\dot{r}\dot{\theta}\sin\theta - \ddot{\theta}r\sin\theta - \dot{\theta}^2 r\cos\theta \tag{4}$$

where if $r = l = constant$,

$$\ddot{x} = -\ddot{\theta}r\sin\theta - \dot{\theta}^2 r\cos\theta \tag{5}$$

or through simplification and rearrangement,

$$\ddot{\theta} = -\frac{\ddot{x}}{r\sin\theta} - \frac{\dot{\theta}^2\cos\theta}{\sin\theta}. \quad (6)$$

When $\theta \approx 90°$, since the CoM pendulum is typically close to vertical, $\cos\theta \approx 0$ and $\sin\theta \approx 1$ so,

$$\ddot{\theta} \approx -\frac{\ddot{x}}{r} \approx -\frac{\ddot{x}}{l}. \quad (7)$$

These equations were utilized by Hof et al. to examine an inverted pendulum in a very similar manner [Figure 2-2].

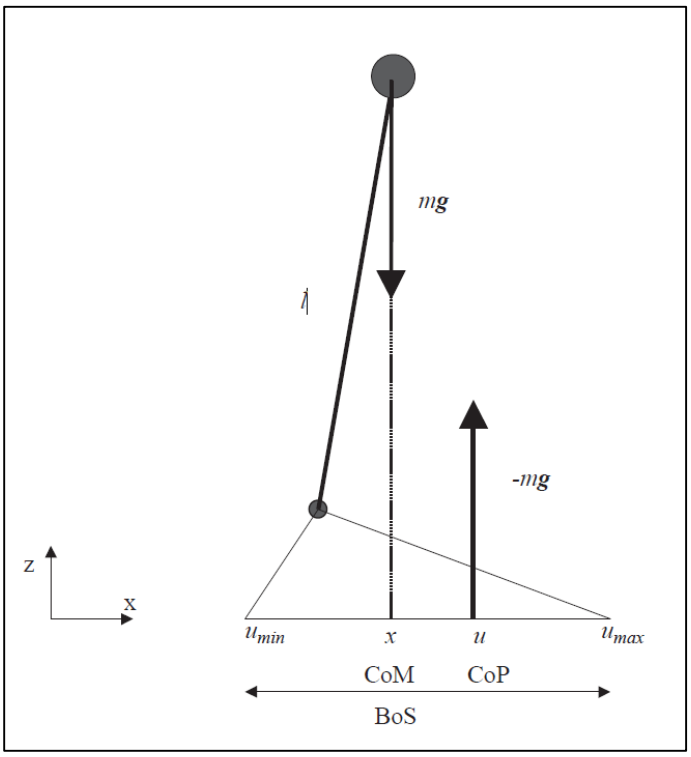

Figure 2-2 Inverted pendulum model. The body is modeled as a single mass m balancing on top of a stick with length l. Indicated are the Centre of Pressure (CoP) u, the location of the effective ground reaction force, and the vertical

projection of the Centre of Mass (CoM) x. The Base of Support (BoS) is the area to which the CoP is confined, and roughly equals the area of the footsole (Hof et al., 2005).

By applying equations (1) and (7), he found that

$$(u - x)mg = I\ddot{\theta} \approx \text{-m}l^2 \frac{\ddot{x}}{l} \tag{8}$$

or

$$u - x = -\frac{l}{g}\ddot{x} = -\frac{\ddot{x}}{\omega_0^2} \tag{9}$$

where he defined a new parameter, $\omega_0 = \sqrt{\frac{g}{l}}$, named the angular eigenfrequency. If this linear second order differential equation (9) is solved assuming the CoP position, u, remains constant, the solution is, cf. (Townsend, 1985):

$$x(t) = u + (x_0 - u)\cosh(\omega_0 t) + \frac{v_0}{\omega_0}\sinh(\omega_0 t) \tag{10}$$

where v_0 and x_0 are the initial velocity and position of the CoM respectively. Instability may ensue if the deceleration of the body fails to prevent the CoM from crossing the CoP. In order to prevent such a situation, the position of the CoM must be less than or equal to the position of the CoP for all time ($x(t) \leq u$). Thus, equation 10 can be rearranged to obey this new constraint as

$$(x_0 - u)\cosh(\omega_0 t) + \frac{v_0}{\omega_0}\sinh(\omega_0 t) \leq 0 \tag{11}$$

or $(u - x_0) \geq \frac{v_0}{\omega_0}\tanh(\omega_0 t)$. (12)

Since $-1 < \tanh(\omega_0 t) < 1$, this equation can be reduced to

$$x_0 + \frac{v_0}{\omega_0} \leq u \ . \tag{13}$$

Hence, if the initial CoM location plus a factor which accounts for its initial velocity and eigenfrequency is less than or equal to the CoP location, then the body will maintain balance. The left hand side of equation 13 was coined the extrapolated center of mass (XcoM) which was shown to be valid for both static and dynamic situations. When the XcoM becomes larger than u_{max}, which lies at the toe end of the BoS, stability will no longer be maintained. By rearranging the terms in equation 13, a new variable was named the margin of stability:

$$b = |u_{max} - \left(x + \frac{v}{\omega_0}\right)|. \tag{14}$$

This variable relates the CoM movements to the BoS area which acts as a spatial stability margin which has been utilized similarly by several other authors (Patton et al., 2000; Popovic et al., 2000; Van Wegen et al., 2002).

Chapter 3

Materials and Methods

3.1 Required Materials

Before developing a model, data was collected utilizing the following materials: a PC running Windows XP, Vicon's Plug-In-Gait Full-Body Model (Vicon, Oxford, UK), Vicon Workstation software, Matlab R2011A, Microsoft Excel 2010, IBM's SamplePower, a six camera Vicon 460 motion capture system, two AMTI OR6-7 force platforms (Advanced Mechanical Technology Inc., Watertown, MA), a 4 inch tall obstacle, and a 25 mm reflective marker set containing 35 markers to place on each subject.

3.2 Subjects

This study's data was collected from seven healthy college aged subjects (21±2.8 years) and seven healthy older adults (71±6.4 years) who all volunteered for a single testing occasion. This number of subjects appeared consistent with various other studies performing similar analyses (Chou et al., 2003; Hof et al., 2005; Huang et al., 2008).

After deciding upon an adequate number of subjects, it was verified that all participants lacked neurological, musculoskeletal, or cognitive disorders. In addition, they self-reported that they were free of hip, knee, or back pain that required treatment within two months immediately preceding testing. No subjects had a history of back or lower limb surgery. All participants provided written

informed consent before testing, and the study was approved by the University of Texas at Arlington's Human Subject's Ethics Committee [Appendix A].

3.3 Testing Procedures

Upon arriving to the testing laboratory, each subject was outfitted with a full-body marker set (35 markers for 15 bodily segments) with 25 mm reflective markers at locations consistent with Vicon's Plug-In-Gait full-body model (Vicon, Oxford, UK) [Appendix B].

Figure 3-1 A subject wearing some of the reflective markers in the locations described by Vicon's Plug-In-Gait Model (Vicon, Oxford, UK) [Appendix B].

Anthropometric measurements of each subject were taken while following the Vicon requirements for dynamic modeling. After ensuring the subjects had

enough time to understand the trials, the subjects were asked to practice walking down the pathway to ensure they hit both force plates without needing to accelerate or decelerate just prior to contact. Subsequently, each subject finished ten barefoot walking trials at self-selected speeds in each of the following situations: walking with eyes open, walking with eyes open while stepping over a 4 inch obstacle, and walking with eyes closed with no obstacle. Only trials in which the subjects contacted both force plates and had little change in speed were retained for analysis. However, some deceleration was allowed for the obstacle-clearing with eyes open condition especially for various elderly subjects.

3.4 Equipment

A six-camera Vicon 460 motion capture system (Vicon, Oxford, UK) was utilized to capture full-body gait kinematics as the subjects traversed the walkway [Figure 3-2].

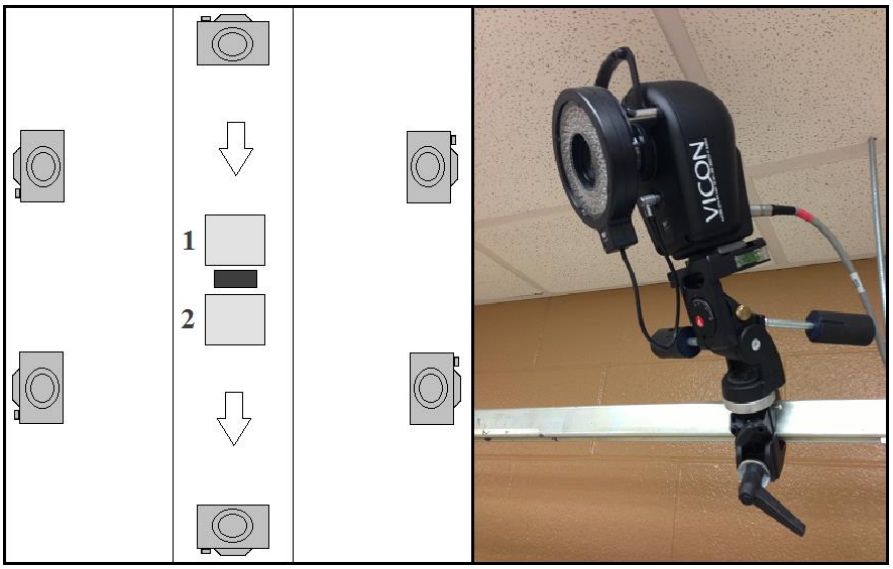

Figure 3-2 The six-camera Vicon 460 motion capture system setup is shown encircling the room. The walkway is displayed down the center with the two grey force plates and black obstacle (left). One of the six cameras is shown mounted (right).

These cameras tracked the three-dimensional locations of each reflective marker at 120 Hz for each trial. Two AMTI OR6-7 force platforms (Advanced Mechanical Technology Inc., Watertown, MA) were embedded successively in the center of the 16 m walkway and laid flush with the floor [Figure 3-3].

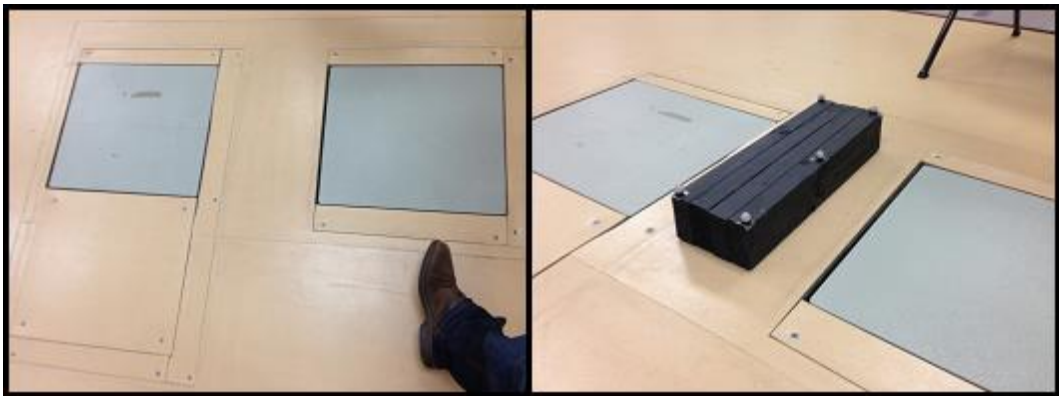

Figure 3-3 The two embedded AMTI OR6-7 force platforms are shown compared to the size of a shoe (left). The 4-inch black obstacle with five reflective markers is planted in between the plates (right).

The force plates sampled at 1080 Hz in synchrony with the cameras.

In the trials that involved clearing an obstacle, a 4 inch tall wooden rectangular block was situated between the two force plates. The obstacle had five reflective markers so that the cameras could note its location while tracking the subjects. The force plates did not require markers since their locations could be determined by the pressure readings.

3.5 Data Analysis

After gathering the force plate and camera data for each subject at their respective sampling rates, the data was processed within the Vicon Workstation software to output the calculated CoM and CoP locations to Microsoft Excel. The CoM locations were calculated by the Vicon software based upon each subject's reflective marker locations, weight, height, and general anthropometric data.

Within Excel, the relevant CoM positions, CoP positions, lower extremity markers, and force plate readings were copied and placed into their own separate Excel document. Subsequently, this data was exported into MATLAB R2011a (MathWorks, Natick, MA) to complete numerical computations which was selected due to its relative ease at handling large matrices. Upon loading the data into MATLAB, the force plate data was filtered utilizing a critically damped Butterworth low pass filter which could be described as a fourth-order recursive Butterworth zero phase shift filter [Appendix D-4]. The filter coefficients were corrected for the number of passes (two) through the filter (Robertson and Dowling, 2003; Winter, 2009). The filter effectively removed any extraneous noise in the data that was generated due to skin movement, electrical interference, and other sources of error. In order to filter, sampling and cutoff frequencies were chosen for both the force plate and marker data in order to eliminate the noise while attempting to retain as much of the important information as possible. At this stage, residual analyses could have been conducted on each of the individual data sets to determine the ideal cutoff frequencies. However, a standardized value for all the force plate data sets was selected as 30 Hz based upon values chosen in previous gait experiments. In addition to the cutoff frequency, the chosen sampling frequency must have been at least twice the highest frequency seen in the raw data, and thus must have obeyed the Nyquist sampling theorem. Choosing a sampling frequency lower than twice the highest frequency would have resulted

in aliasing of critical information. Hence, a standardized value of 1080 Hz was chosen for the sampling frequency of the force plate data. After deciding upon those values, a critically damped low-pass filter was chosen over a normal low-pass filter to better match the force plate readings [Figure 3-4].

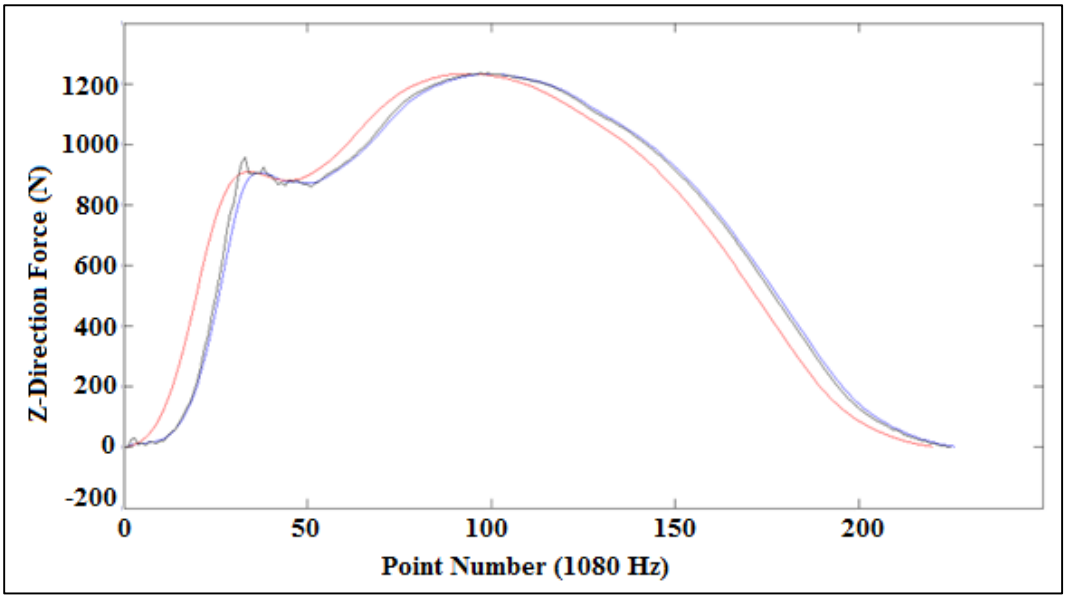

Figure 3-4 Different filtering methods are displayed for this force plate z-direction force data. Original unfiltered data = black. Low-pass Butterworth filter = red. Critically damped low-pass Butterworth filter= blue.

As shown in the figure above, the actual z-direction force data (black) is shown to be very jagged near the first peak. This is a sign that the data definitely contains noise and requires filtering. When utilizing a low-pass Butterworth filter (red) on the data, the result overshoots and undershoots the actual data continuously and appears to have a phase shift to the left. The critically damped low-pass

Butterworth filter's result (blue) appears to align much more appropriately with the data and thus was a better option.

While the critically damped filter worked well for the force plate, the low-pass Butterworth filter worked much better with the individual marker data [Appendix D-3]. This was due to the fact that the critically damped filter did not work well when derivatives needed to be applied to the resulting smoothed data. Since the marker's individual positions were ultimately utilized to determine segment velocities and accelerations, a regular low-pass filter was utilized for the marker data. Ideally, different cutoff frequencies should have been selected for each marker depending upon their individual residual analyses. However, a standardized cutoff frequency of 6 Hz was chosen across all the markers. In addition, the filter's sampling frequency was chosen at 120 Hz which was definitely sufficient since most human movements occur at relatively low frequencies, so "displacement-time data seldom needs to be sampled at rates in excess of 100 Hz, and indeed, that rates as low as 25-30 Hz often suffice" (Wood, 1982).

After filtering the force plate and marker data, the data needed to be trimmed down to only contain values associated with the single-leg support phase of gait. In this phase, the leading foot was firmly planted on the second force plate. The minimum value of the range was found by determining when the trailing foot completed lifted off the first force plate by noting when the force in

the z-direction went to zero. The maximum of the range was determined by finding the locations when the absolute value of the leading limb's heel vertical velocity fell under 200 $\frac{mm}{s}$, when the heel's acceleration remained positive, and when the heel's vertical position fell under 100 mm. These values were determined arbitrarily based upon several subjects' values when their respective leading limb heel strikes occurred. Upon deciding the range and removing extraneous data, the initial position coordinates and the initial angular velocity were determined for the CoM pendulum. In order to perform this velocity calculation, a two-point finite difference formula was utilized which computed a secant line through the points after (x+1) and before (x-1) the value of interest (x).

$$f'(x) \approx \frac{f(x+1)-f(x-1)}{2*dt} \tag{15}$$

The advantage of utilizing this method over other finite difference methods was that the error was proportional to dt^2 which was very miniscule since $dt = \frac{1}{120}$ seconds. This value of dt was chosen since the camera system sampled at 120 Hz. Since the force plates sampled at 1080 Hz, either the camera system had to be upsampled or the force plates had to be downsampled in order for the two systems to contain the same number of data points. Since upsampling requires unreliable interpolation, the force plate data was downsampled to 120 Hz and consequently, only one of every nine force plate data points was utilized in order to ensure synchrony.

After supplying the model's input conditions of position and velocity, the subsequent positions in time had to be determined. Before proceeding any further, a free body diagram of the pendulum was drawn in order to develop the equation of motion [Figure 3-5].

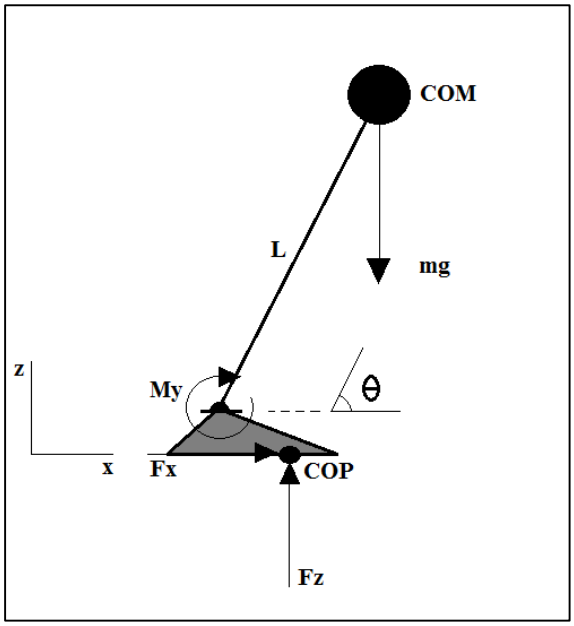

Figure 3-5 Free body diagram of the inverted pendulum model with a base of rotation about the ankle and a point mass at the location of the CoM. Input conditions include the mass at the CoM, the y-direction moment at the ankle at each instance in time, the x- and z-direction forces applied at the CoP at each instance in time, the locations of the CoP and ankle at each instance in time, and the initial angle θ and angular velocity $\dot{\theta}$.

The model's differential equation of motion was found to be:

$$\ddot{\theta} = \frac{-mgLcos(\theta)}{I} + \frac{AnkleTorque}{I} + \frac{Fx*Ha}{I} + \frac{Fz*La}{I} \quad (16)$$

where θ was the angle from the ground up to the pendulum, $\ddot{\theta}$ was the angular acceleration of the pendulum, I was the moment of inertia (mL^2), F_y was the y-direction Force, F_z was the z-direction force, H_a was the height to the ankle from the base of support, and L_a was the length from the center of pressure to the ankle. This equation of motion differed from Hof's equation (Equation 7) since the variables were kept in the polar coordinate system and did not assume the pendulum was always near vertical (Hof et al., 2005). In addition, this equation of motion allowed the length of the pendulum, L, to change which was vital for the model's validity since the length from the ankle to the CoM constantly fluctuates while walking.

Since equation 16 merely solves for the accelerations, the velocity and position values at each instance in time had to be solved utilizing a fourth-order Runge-Kutta numerical integration technique:

K1 = penddyn (time , θ , …)

$$\theta_{mid} = K1 * \frac{dt}{2} + \theta(i)$$

K2 = penddyn (time + $\frac{dt}{2}$, θ_{mid} ,…)

$$\theta_{mid} = K2 * \frac{dt}{2} + \theta(i)$$

K3 = penddyn (time + $\frac{dt}{2}$, θ_{mid} ,…)

$$\theta_{end} = K3 * dt + \theta(i)$$

K4 = penddyn (time + dt , θ_{end} ,…)

$$\theta(i+1) = \theta(i) + \frac{dt}{6} * (K1 + 2 * K2 + 2 * K3 + K4) \quad (17)$$

This type of process created accumulated error with a magnitude of dt^4 but avoided the need to preform two analytical integrations. Provided various inputs, the constants K1 through K4 called a separate function named "penddyn" (Appendix D-7) which utilized the equation of motion (Equation 16). This "penddyn" function ran four times for each constant until the next angle in time, $\theta(i+1)$, could ultimately be determined. This process was repeated until there was enough angular position and velocity data to fill the single-leg stance phase requirements.

In order to validate this model's use, the model's CoM position, at each instance in time, had to remain reasonably close to the actual motion of the subject CoMs. Hence, the percent error differences for each consecutive CoM position were determined by calculating the percent error of the pendulum angle, θ (Equation 18).

$$Percent\ Error\ Theta\ (\%) = \frac{Calculated\ \theta - Actual\ \theta}{Actual\ \theta} * 100 \quad (18)$$

The model was arbitrarily deemed reasonable if these percent error values were consistently fewer than 10% [Appendix D-8].

After validating the model, the limits of stability had to be determined for each subject and each condition. In order to define the boundary of stability,

Hof's XcoM equation (Equation 13) was used to test a range of angular position and velocity values (Equation 19).

$$XcoM = x_0 + \frac{v_0}{\omega_0} \tag{19}$$

By applying an iterative testing technique, a series of initial angles and angular velocities were inputted into the model. The model then calculated the resulting CoM locations at each instance in time for the single-leg stance phase for each inputted pair of values. Subsequently, the XcoM equation determined whether instability occurred at any point during this gait phase. If the XcoM became greater than the toe or less than the heel positions at any time during the model's simulation for a given set of input conditions, those initial values were recorded as instable conditions. However, if the XcoM did not pass the toe or heel markers at any time during the model's simulation for a given pair of inputs, the pair was recorded as stable conditions. The remaining stability conditions were determined by scanning a large set of input conditions by either adding 1 degree or 1 degree/sec to the original input angle or velocity respectively for a number of iterations. For instance, if the initial angle was set to 100 degrees, the model ran via the MATLAB code for each initial velocity value (intervals of 1 degree/sec) ranging between -40 and 40 degrees/sec. For each of these initial velocity values, the XcoM was determined from each subsequent CoM location determined by the model. If any of these XcoM values crossed either the heel or toe markers, the corresponding initial values were recorded as instable conditions. Subsequently,

the angle was changed to 101 degrees and the process was repeated. Consequently, due to the extremely high number of calculations, many large matrices of instability testing results were generated. After determining which pairs of initial conditions did not result in instability throughout their respective model simulations, the boundary of stability and its area was determined utilizing Simpson's rule [Appendix D-5]. Subsequently, this area, coined the area of stability (AoS), was compared across different age groups and conditions [Appendix D-9]. Larger AoS values indicated that instability would less likely occur for that particular condition or subject.

3.6 Statistical Analysis

A 2 x 3 repeated measures ANOVA was completed with each row representing the two age groups (young and elderly) and each column representing each of the three conditions (eyes open, obstacle, eyes closed) to determine the effects of each condition and age group on the area of stability. Alpha was set at 0.05 for all the comparisons.

Chapter 4

Manuscript

4.1 Abstract

Falls during walking are the primary cause of injuries for older persons and present a high associated cost to society due to the resulting treatment. Hence, the control of balance while moving is an essential issue especially amongst the elderly population. Better understanding of why and how these falls occur could lead to better fall prevention methods, therapies, or devices. These changes could effectively decrease overall emergency room visits, create more room in hospitals for other demands, and decrease medical costs for fall-related disabilities and deaths amongst patients overall.

This study will attempt to develop an inverted pendulum model of the single-leg support phase of walking for the sagittal plane. Upon validating the model's use, the model will then be utilized to predict the limits of stability during this walking phase for various conditions including: eyes open with no obstacle, eyes closed with no obstacle, and eyes open with an obstacle. Additionally, these stability regions will then be compared across elderly and young subjects to analyze whether their areas of stability significantly differ.

4.2 Introduction

Falls during walking are the primary cause of injuries for older persons (Berg et al., 1997; Ihlen et al., 2012; Luukinen et al., 1994; Niino et al., 2000;

Sterling et al., 2001). Due to their high rate of incidence, falls present a high associated cost to society due to the resulting treatment. Therefore, much research has been conducted to better understand gait stability to help identify people at risk of falling. For instance, research has shown that older subjects tend to exhibit a decreased recovery performance compared to their younger counterparts when met with an unexpected perturbation such as tripping (Berg et al., 1997; Grabiner et al., 2005; Karamanidis and Arampatzis, 2007; Pijnappels et al., 2005). This decrease in responsiveness and effectiveness can be attributed to degeneration of the motor cortical regions and neurotransmitter systems (Seidler et al., 2010), a slower force generation within muscles (Vandervoort and Mccomas, 1986), tendon stiffness and decreased muscle strength in general (Fukagawa and Schultz, 1995; Grabiner et al., 2005; Karamanidis et al., 2008). Aging may also cause different plantar loadings due to changes in foot characteristics such as flatter and more pronated feet. These changes could lead to decreased forces on the heel and lateral forefoot, but also longer contact times at the heel, midfoot, and metatarsophalangeal joints (Scott et al., 2007). Additionally, spatial characteristics, such as the center of pressure (CoP) distribution across the foot during gait, and temporal characteristics, such as the CoP velocity at various gait intervals, will also change due to changes in foot shape. In particular, the elderly have been shown to exhibit faster CoP velocity during heel strike, slower CoP

velocity during mid-stance, and a more significant medial CoP progression curve than their young counterparts (Chiu et al., In Press).

The elderly also typically exhibit slower preferred walking speeds with shorter step lengths which may also contribute to an increased risk of falls (Laufer, 2005). Various authors have shown that a slower gait is directly linked with an increased risk of falls (Abella et al., 2005; Cromwell and Newton, 2004). However, slower gait speeds have also been shown to increase local stability (Dingwell and Marin, 2006; England and Granata, 2007; Li et al., 2005), and shorter step lengths have been shown to be beneficial when on slippery surfaces (Moyer et al., 2006). Thus, these findings are very controversial, and it is rather unclear whether decreased walking speeds or decreased step lengths actually contribute to a higher incidence of falls. "Shorter steps may in fact provide greater stability, but this benefit may be offset by the accompanying slowness in gait speed" (Bhatt et al., 2011).

While older persons may exhibit some or all of these drawbacks, they have likely experienced a large number of unexpected perturbations throughout their lifetime simply due to their age. Since people possess the ability to learn from prior experiences, the elderly often can improve their dynamic stability and counteract these drawbacks associated with old age via predictive and adaptive adjustments (Marigold and Patla, 2002; Pai et al., 2003; Van Der Linden et al., 2007). In fact, repetitive simulations of perturbations while walking have been

shown to improve the reactive behavior in both young and elderly subjects (Bierbaum et al., 2011). However, these predictive adjustments assume that the perturbation is somewhat expected. If the perturbation is truly unexpected, as is often the case in real life scenarios, the person must utilize purely reactive adjustments which are based upon information received during the movement. Since age often diminishes the capacity of the proprioceptive system and affects the afferent sensory system, less information is ultimately transmitted to the spinal column about the disturbance (Bierbaum et al., 2011; Hurley et al., 1998; Lord et al., 1996; Patel et al., 2009).

In addition to physical changes within the human body, people that have experienced falls may actually alter their normal gait due to mental fears of future falls. This fear may cause them to have slower gait or to have abnormal step width variability. Extreme step width variability can be another cause for imbalance and in one study, subjects with "either low or high step width variability were 4.38 times more likely to have fallen in the past year (Barry et al., 2005). Hence, there are many reasons that falls take place and many researchers have attempted to find commonalities between each type of fall. The ultimate goal is to better anticipate under what circumstances a fall will occur to hopefully prevent future falls from transpiring.

While standing still, the ability to maintain balance depends upon the vertical projection of the body's center of mass (CoM) with respect to the base of

support and the CoM movement velocity (Patton et al., 1999). The base of support is defined as the region in which the center of pressure can be generated (Hof et al., 2005). During standing, if the CoM is too far outside the base of support, the person will fall or be forced to take a step. Similarly, if the CoM is moving too quickly in any given direction, such as when someone is pushed from behind while standing, the person will also fall. While maintaining stability during standing is an area of great concern, many falls occur during locomotion. While moving, the CoM repeatedly shifts beyond and behind the base of support. Unlike during standing, a person can still maintain balance while moving when the CoM greatly exceeds the base of support. Thus, the control of CoM velocity and momentum plays a much larger role in maintaining balance during locomotion.

In particular, imbalance and tripping, due to the failure to maintain adequate CoM control and velocity during obstacle crossing, leads to "two of the most common causes of falls in the elderly" (Chou et al., 2003). Successful obstacle-clearing attempts require adequate foot clearance of both the leading and trailing legs, the ability to exhibit a longer duration single-leg stance while the other leg is crossing the obstacle, and correct body segment coordination. If a subject fails to complete any of these steps, their CoM will move too quickly in one direction (such as a sideways) and the subject will likely fall.

4.3 Theory

In several past papers, inverted pendulums have been utilized to model the stance leg in balance dynamics (Gurfinkel, 1973; Hemami and Golliday Jr, 1977; Lee and Patton, 1997; Winter, 1995). In these situations, the body mass is assumed to be a point mass which resides at the center of mass of the entire body. The pendulum length extends from this point mass down to a pivot location at the ankle. This type of two-dimensional model greatly simplifies the overall motion of the body which consists of many segments with many individual segment masses. Each of these segments rotates at different angular velocities and accelerations during locomotion which makes the physics and mathematics fairly complex. In order to avoid such complexities, these two-dimensional models only have one artificial segment, one point mass, one angular velocity and one angular acceleration.

Simple inverted pendulums obey Euler's equation:

$$\sum \boldsymbol{M} = I\ddot{\theta} \tag{1}$$

where **M** includes the moments about the ankle rotation point, $I = ml^2$ if l is assumed to be a fixed length, and $\ddot{\theta}$ is the angular acceleration of the pendulum arm. If one wanted to convert from polar coordinates (r, θ) into a Cartesian frame (x, y), the conversion equations would need to be utilized:

$$x = r\cos\theta \tag{2}$$

$$\dot{x} = \dot{r}\cos\theta - \dot{\theta}r\sin\theta \tag{3}$$

$$\ddot{x} = \ddot{r}\cos\theta - 2\dot{r}\dot{\theta}\sin\theta - \ddot{\theta}r\sin\theta - \dot{\theta}^2 r\cos\theta \tag{4}$$

where if $r = l = constant$,

$$\ddot{x} = -\ddot{\theta}r\sin\theta - \dot{\theta}^2 r\cos\theta \tag{5}$$

or through simplification and rearrangement,

$$\ddot{\theta} = -\frac{\ddot{x}}{r\sin\theta} - \frac{\dot{\theta}^2\cos\theta}{\sin\theta} \tag{6}$$

and when $\theta \approx 90°$, since the CoM pendulum is typically close to vertical, $\cos\theta \approx 0$ and $\sin\theta \approx 1$ so,

$$\ddot{\theta} \approx -\frac{\ddot{x}}{r} \approx -\frac{\ddot{x}}{l}. \tag{7}$$

These equations were utilized by Hof et al. to examine an inverted pendulum in a very similar manner [Figure 4-1].

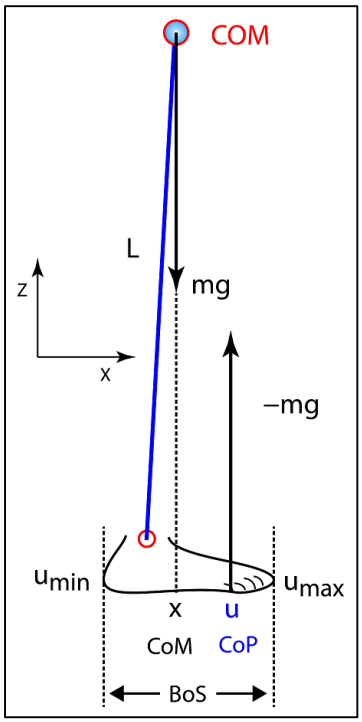

Figure 4-1 Inverted pendulum model displaying the pendulum arm extending from the ankle rotation point to the COM point mass. The variables x and u mark the x-direction locations of the CoM and CoP respectively. The BoS is defined between the boundaries u_{min} and u_{max} as established by Hof et al.

By applying equations (1) and (7), he found that

$$(u - x)mg = I\ddot{\theta} \approx \text{-m}l^2 \frac{\ddot{x}}{l} \tag{8}$$

or

$$u - x = -\frac{l}{g}\ddot{x} = -\frac{\ddot{x}}{\omega_0^2} \tag{9}$$

where he defined a new parameter, $\omega_0 = \sqrt{\frac{g}{l}}$, named the angular eigenfrequency. If the linear second order differential equation (9) is solved assuming the CoP position, u, remains constant, the solutioin is, cf. (Townsend, 1985):

$$x(t) = u + (x_0 - u)\cosh(\omega_0 t) + \frac{v_0}{\omega_0}\sinh(\omega_0 t) \tag{10}$$

where v_0 and x_0 are the initial velocity and position of the CoM respectively. Instability may ensue if the deceleration of the body fails to prevent the CoM from crossing the CoP. In order to prevent such a situation, the position of the CoM must be less than or equal to the position of the CoP for all time ($x(t) \leq u$). Thus, equation 10 can be rearranged to obey this new constraint as

$$(x_0 - u)\cosh(\omega_0 t) + \frac{v_0}{\omega_0}\sinh(\omega_0 t) \leq 0 \tag{11}$$

or $$(u - x_0) \geq \frac{v_0}{\omega_0}\tanh(\omega_0 t). \tag{12}$$

Since $-1 < \tanh(\omega_0 t) < 1$, this equation can be reduced to

$$x_0 + \frac{v_0}{\omega_0} \leq u\ . \tag{13}$$

Hence, if the initial CoM location plus a factor which accounts for its initial velocity and eigenfrequency is less than or equal to the CoP location, then the body will maintain balance. The left hand side of equation 13 was coined the extrapolated center of mass (XcoM) which was shown to be valid for both static and dynamic situations. When the XcoM becomes larger than u_{max}, which lies at the toe end of the BoS, stability will no longer be maintained. Similarly, when the

XcoM becomes smaller than u_{min}, which lies at the heel end of the BoS, stability will no longer be maintained. Therefore, as long as the XcoM stays within the BoS, falls will not occur in the anterior-posterior directions.

This study expanded Hof's work and utilized a variant of his free body diagram [Figure 4-2].

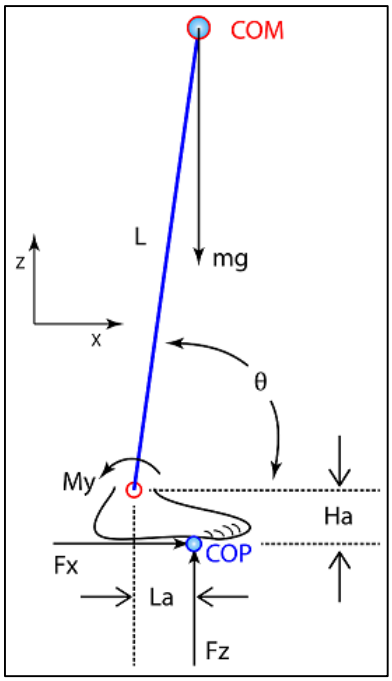

Figure 4-2 Free body diagram of the inverted pendulum model with a base of rotation about the ankle and a point mass at the location of the COM. Input conditions include the mass at the COM, the y-direction moment at the ankle at each instance in time, the x- and z-direction forces applied at the CoP at each

instance in time, the locations of the CoP and ankle at each instance in time, and the initial angular position θ and angular velocity $\dot{\theta}$.

By utilizing equation 1, this model's differential equation of motion was found to be:

$$\ddot{\theta} = \frac{-mgLcos(\theta)}{I} + \frac{AnkleTorque}{I} + \frac{Fx*Ha}{I} + \frac{Fz*La}{I} \quad (14)$$

where θ is the angle between the ground and the pendulum, $\ddot{\theta}$ is the angular acceleration of the pendulum, I is the moment of inertia (mL^2) for a rotating point mass with a massless segment, F_y is the y-direction force, F_z is the z-direction force, H_a is the height to the ankle from the base of support, and L_a is the length from the center of pressure to the ankle. This equation of motion differs from Hof's equation (Equation 7) since the variables were kept in the polar coordinate system and did not assume the pendulum was always near vertical. In addition, this equation of motion allowed the length of the pendulum, L, to change which was vital for the model's validity since the length from the ankle to the CoM constantly fluctuated.

Since equation 14 merely solved for the angular accelerations, the angular velocity and position values at each instance in time were solved utilizing a fourth-order Runge-Kutta numerical integration technique in MATLAB:

K1 = penddyn (time , θ , …)

$$\theta_{mid} = K1 * \frac{dt}{2} + \theta(i)$$

K2 = penddyn (time + $\frac{dt}{2}$, θ_{mid} ,…)

$$\theta_{mid} = K2 * \frac{dt}{2} + \theta(i)$$

K3 = penddyn (time + $\frac{dt}{2}$, θ_{mid} ,…)

$$\theta_{end} = K3 * dt + \theta(i)$$

K4 = penddyn (time + dt , θ_{end} ,…)

$$\theta(i+1) = \theta(i) + \frac{dt}{6} * (K1 + 2 * K2 + 2 * K3 + K4) \quad (15)$$

This type of process created accumulated error with a magnitude of dt^4 but avoided the need to preform two analytical integrations. The constants K1 through K4 called a separate function within MATLAB named "penddyn" (Appendix D-7) which utilized the equation of motion (Equation 14). This "penddyn" function ran four times for each constant until the next angle in time, $\theta(i+1)$, could ultimately be determined. This entire process was repeated until there was enough angular position and velocity data to fill the single stance phase requirements.

In order to validate this model's use, the model's CoM position at each instance in time had to remain reasonably close to the actual motion of the subject CoMs. Hence, the percent error differences for each consecutive CoM location were determined by calculating the percent error of the pendulum angle θ (Equation 16).

$$Percent\ Error\ Theta\ (\%) = \frac{Calculated\ \theta - Actual\ \theta}{Actual\ \theta} * 100 \quad (16)$$

The model was arbitrarily deemed reasonable if these percent errors were consistently fewer than 10%.

After validating the model, the limits to stability had to be determined for each subject and each condition. In order to define the boundary of stability, Hof's XcoM equation (Equation 13) was utilized to test a range of initial angular position and velocity values (Equation 17).

$$XcoM = x_0 + \frac{v_0}{\omega_0} \tag{17}$$

By applying an iterative testing technique, a series of initial angles and angular velocities were inputted into the model. The model then calculated the resulting CoM locations at each instance in time for the single-leg stance phase for each inputted pair of values. Subsequently, the XcoM equation determined whether instability occurred at any point during this gait phase. If the XcoM became greater than the toe or less than the heel positions at any time during the model's simulation for a given set of input conditions, those initial values were recorded as instable conditions. However, if the XcoM did not pass the toe or heel markers at any time during the model's simulation for a given pair of inputs, the pair was recorded as stable conditions. The remaining stability conditions were determined by scanning a large set of input conditions by either adding 1 degree or 1 degree/sec to the original input angle or velocity respectively for a number of iterations. For instance, if the initial angle was set to 100 degrees, the model ran via the MATLAB code for each initial velocity value (intervals of 1 degree/sec)

ranging between -40 and 40 degrees/sec. For each of these initial velocity values, the XcoM was determined from each subsequent CoM location determined by the model. If any of these XcoM values crossed either the heel or toe markers, the corresponding initial values were recorded as instable conditions. Subsequently, the angle was changed to 101 degrees and the process was repeated. Consequently, due to the extremely high number of calculations, many large matrices of instability testing results were generated. After determining which pairs of initial conditions did not result in instability throughout their respective model simulations, the boundary of stability and its area was determined utilizing Simpson's rule. Subsequently, this area, coined the area of stability (AoS), was compared across different age groups and conditions. Larger AoS values indicated that instability would less likely occur for that particular condition or subject.

4.4 Experimental Methods

This study's data was collected from seven healthy college aged subjects (21±2.8 years) and seven healthy older adults (71±6.4 years) who all volunteered for a single testing occasion. This number of subjects appeared consistent with various other studies performing similar analyses (Chou et al., 2003; Hof et al., 2005; Huang et al., 2008). All participants were required to lack neurological, musculoskeletal, or cognitive disorders. In addition, they self-reported that they were free of hip, knee, or back pain that required treatment within two months

immediately preceding testing. No subjects had a history of back or lower limb surgery. All participants provided written informed consent before testing, and the study was approved by the University of Texas at Arlington's Human Subject's Ethics Committee [Appendix A].

Upon arriving to the testing laboratory, each subject was outfitted with a full-body marker set (35 markers for 15 segments) with 25 mm reflective markers at locations consistent with Vicon's Plug-In-Gait full-body model (Vicon, Oxford, UK). Anthropometric measurements of each subject were taken while following Vicon's requirements for dynamic modeling. After ensuring the subjects had enough time to understand the trials, the subjects were asked to practice walking down the pathway to ensure they hit both force plates without needing to accelerate or decelerate. Subsequently, each subject finished ten barefoot walking trials at self-selected speeds in each of the following situations: walking with eyes open, walking with eyes open while stepping over a 4 inch obstacle, and walking with eyes closed with no obstacle. Only trials in which the subjects contacted both force plates and had little change in speed were retained for analysis. However, some deceleration was allowed for the obstacle clearing with eyes open condition especially for the elderly participants.

A six-camera Vicon 460 motion capture system (Vicon, Oxford, UK) was utilized to capture full-body gait kinematics as the subjects traversed the walkway [Figure 4-3].

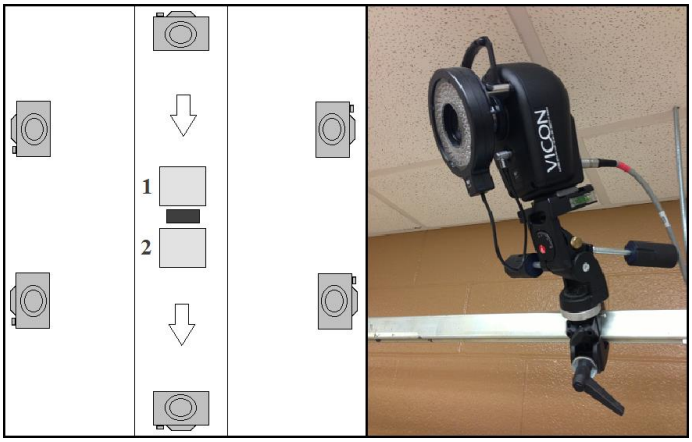

Figure 4-3 The six-camera Vicon 460 motion capture system setup is shown encircling the room. The walkway is displayed down the center with the two grey force plates and black obstacle (left). One of the six cameras is shown mounted (right).

These cameras tracked the three-dimensional locations of each reflective marker at 120 Hz for each trial. Two AMTI OR6-7 force platforms (Advanced Mechanical Technology Inc., Watertown, MA) were embedded successively in the center of the 16 m walkway and laid flush with the floor [Figure 4-4].

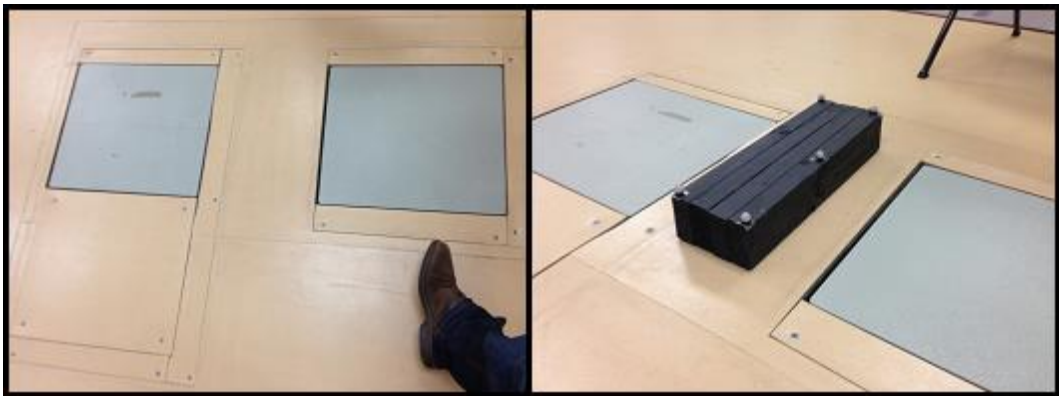

Figure 4-4 The two embedded AMTI OR6-7 force platforms are shown compared to the size of a shoe (left). The 4-inch black obstacle with five reflective markers is planted in between the plates (right).

The force plates sampled at 1080 Hz in synchrony with the cameras.

In the trials that involved clearing an obstacle, a 4 inch tall wooden rectangular prism was situated between the two force plates. The obstacle had five reflective markers so that the cameras could note its location while tracking the subjects. The force plates did not require markers since their locations could be determined by the pressure recordings.

4.5 Results

After compiling the force plate and reflective marker data into their respective matrices within MATLAB, a program ran, provided CoM position and velocity input conditions, which calculated each successive CoM position and velocity value until the model reached the end of the single-limb stance phase. The program repeated this process for the total number of subjects and conditions.

In order to visualize the model's approximated CoM movement, the program also outputted an animation of the stance phase [Figure 4-5].

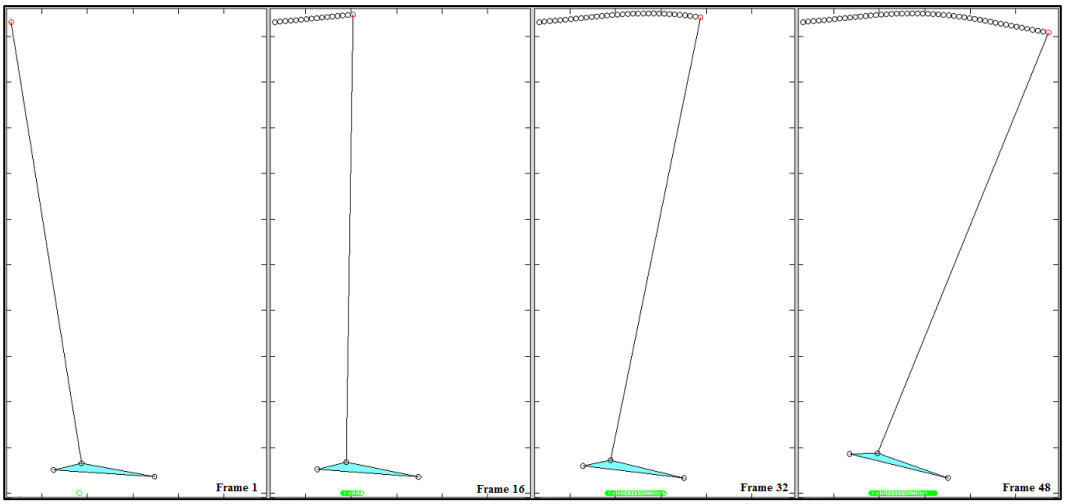

Figure 4-5 CoM pendulum model progression in MATLAB in the sagittal plane. The pendulum extends from the ankle to the CoM (red circle). The CoM and CoP locations throughout the entire single-leg support phase motion are displayed as black and green circles respectively. The foot is shown as cyan triangle with heel, ankle, and toe markers (black circles) as corners.

The figure displays the modeled CoM locations with the actual CoP, heel, ankle, and toe marker locations for four different intervals of the motion. The first interval (far left) displays the leading foot and CoM pendulum orientation when the trailing leg's toe lifts off the ground. The second and third intervals display the animation's frames 16 and 32, respectively, which show the CoM and CoP progressing forward as well as the heel lifting from the ground. Somewhere between the second and third intervals, the trailing limb becomes the leading limb

as the CoM's vertical projection approaches the balls of the feet. Note that the figure does not display either of the limbs, and that only the CoM pendulum is plotted. Lastly, in the fourth interval at frame 48, the newly leading limb's heel hits the ground, and the animation ends since the subject left the single-leg stance phase and entered the double leg support phase.

After tracing and animating the model's CoM locations, the program also compared these predicted values to the actual subject's CoM locations throughout time as determined by the Vicon Workstation software. These values were then plotted in order to establish a visual comparison [Figure 4-6].

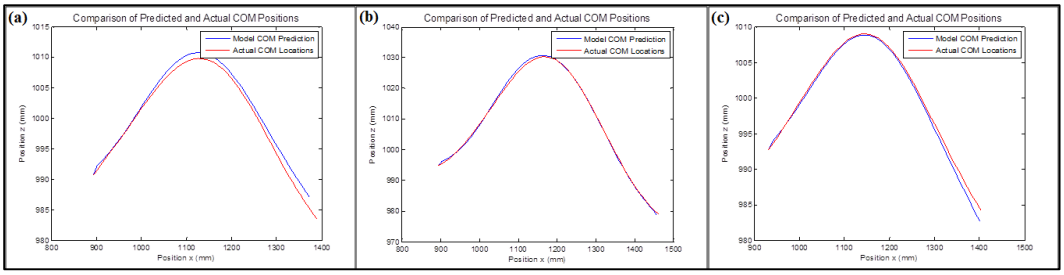

Figure 4-6 Subject 46's CoM location comparison plots for three different conditions (a) eyes open with no obstacle, (b) eyes open with an obstacle, and (c) eyes closed with no obstacle. The maximum theta percent difference for these conditions were 1.607, 1.029, and 0.592% respectively.

As displayed for this particular subject, the model predicted the CoM progression relatively accurately for all three conditions. The remaining subject's maximum theta percent error results are displayed in Appendix C-1 and vary between 0 and 9.98%. While there was a wide range of results, no subjects' data resulted in

percent error values above 10%. Hence, no data sets had to be excluded from the limits of stability calculations.

After running the program which determined the maximum theta percent error values, a separate program ran for each condition and subject to determine the limits of stability. The program utilized the model to calculate predicted CoM locations throughout the single-leg support phase for a wide range of initial angular positions and velocities. Subsequently, the XcoM was calculated utilizing each of the CoM values. If the XcoM became greater than the toe marker x-direction locations at any time for a particular set of initial conditions (exceeding the base of support), the pair of inputs was recorded as outside the area of stability. Similarly, if the XcoM became less than the heel marker x-direction locations at any point in time (also exceeding the base of support), the pair of inputs was recorded as outside the area of stability. Lastly, pairs of inputs that resulted in XcoM locations that never left the base of support were collectively recorded as the area of stability [Figure 4-7].

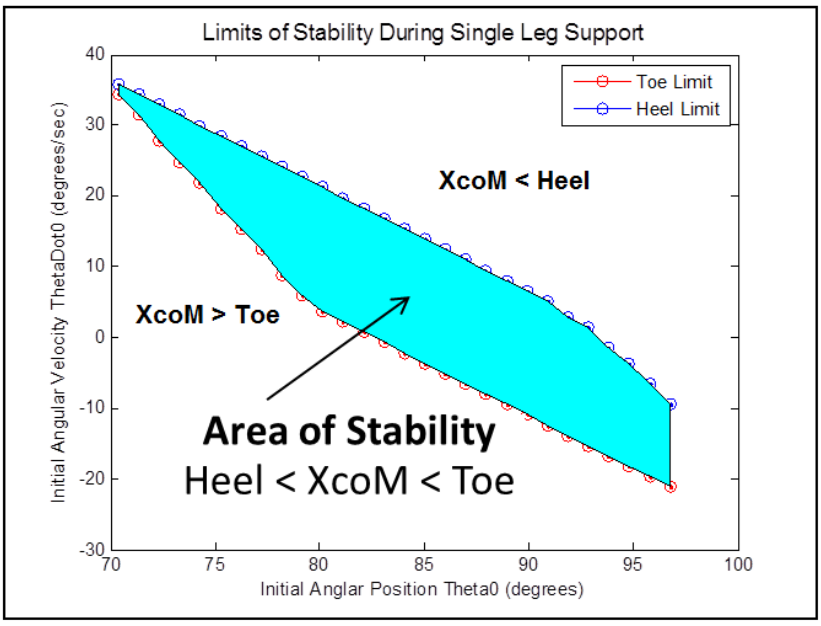

Figure 4-7 Limits of stability graphs displaying the area of stability. The region above the heel limits (blue curve (left)) signify that the XcoM < Heel. The region below the toe limits (red curve (left)) signify that the XcoM > Toe. Stability is maintained if Heel < XcoM < Toe and this region is the area of stability (cyan region (right)).

This area of stability is displayed in cyan and displays the initial value combinations which may be utilized to remain stable during the single-leg support phase. The initial angular position is shown to vary between 70 and 97 degrees where 90 degrees implies the CoM pendulum is vertical. If the initial angular position is a relatively low value, such as 70 degrees, the angular velocity must be high in order to allow the pendulum to return to near vertical. Inversely, if the initial angle is high, such as 97 degrees, the angular velocity must be negative in

order to propel the pendulum forward to remain upright. As initial angular position increases, initial angular velocity must decrease in order to ensure stability. Additionally, as the initial conditions reach their respective extreme values, the region becomes smaller indicating stability is much more difficult to maintain under these circumstances.

These areas of stability were subsequently calculated across every subject and every condition utilizing Simpson's rule. Relatively high calculated area values for a particular condition implied the subject was more stable. Smaller areas meant the subject was more likely to fall. In order to make these area size comparisons, average area values were calculated for each age group and condition [Table 4-1].

Table 4-1 Mean area of stability values (deg^2/sec) with their respective standard deviation values for all conditions and age groups. Condition 1 = Eyes open with no obstacle. Condition 2 = Eyes open with obstacle. Condition 3 = Eyes closed with no obstacle.

	Condition 1	**Condition 2**	**Condition 3**	**Condition Means**
Young	152.49 (31.76)	117.16 (12.34)	129.52 (34.71)	133.06 (30.57)
Elderly	120.48 (30.79)	81.57 (19.45)	138.12 (20.99)	113.39 (33.41)
Age Means	136.48 (34.33)	99.37 (24.21)	133.82 (27.92)	

As displayed above, the young participants generated larger areas of stability for the eyes open with no obstacle and the eyes open with an obstacle conditions. However, the elderly group had a higher average area of stability while walking

with eyes closed. Hence, the young subjects were more stable during the first two conditions but were actually more likely to fall compared to the elderly group during the eyes closed condition. In addition to the age differences, the young subjects appeared most stable during the eyes open without an obstacle condition and least stable during the eyes open with an obstacle condition as might be expected. Interestingly, the elderly group was most stable during the eyes closed condition. This result was counterintuitive since one would have expected the stability to decrease if there was no visual feedback.

By utilizing IBM's SPSS software, a 2 x 3 factorial ANOVA with two between subjects factors, age group (young and elderly) and condition (eyes open, eyes open with an obstacle, and eyes closed), was utilized to determine the effects of age group and condition on the area of stability values. The results revealed that there was no significant difference between the young and elderly age group's area of stability values ($p=0.087$). This p-value result would have likely decreased if the sample size was larger. Due to the relatively low sample size of 7 subjects per age group, the p-value could not fall below the alpha value of 0.05. However, there was a significant interaction between age group and condition overall ($p=0.019$) as determined by the ANOVA. Additionally, when comparing the two age groups for each condition, the various conditions displayed a significant main overall effect ($p=0.000$). Subsequently, Sidak post hoc tests were completed to compare significant ANOVA effects. When AoS values were

averaged across both young and elderly age groups, the obstacle condition was shown to be significantly different from both the eyes closed and eyes open with no obstacle conditions ($p=0.001$ and $p=0.001$ respectively). In order to examine these pairwise comparisons more closely, the AoS averages within the age groups were analyzed separately. First, a pairwise comparison was performed between age groups within each condition. The two age groups AoS values only significantly differed for the condition with an obstacle ($p=0.002$). Subsequently, when performing a pairwise comparison between conditions within each age group, the eyes open condition's area results were shown to be significantly different from the obstacle condition for both young and elderly age groups ($p=0.019$ and $p=0.010$ respectively). Lastly, the statistics showed that the eyes closed condition was significantly different from the obstacle condition for the elderly age group only ($p=0.000$). As briefly mentioned earlier, additional significant AoS differences may have been achieved if the sample size was increased.

In addition to calculating the areas of stability and performing the statistical analysis, the resulting areas were also displayed across each age group and for each condition. The figure below corresponds to the data generated by the young subject group during the eyes open with no obstacle condition [Figure 4-8].

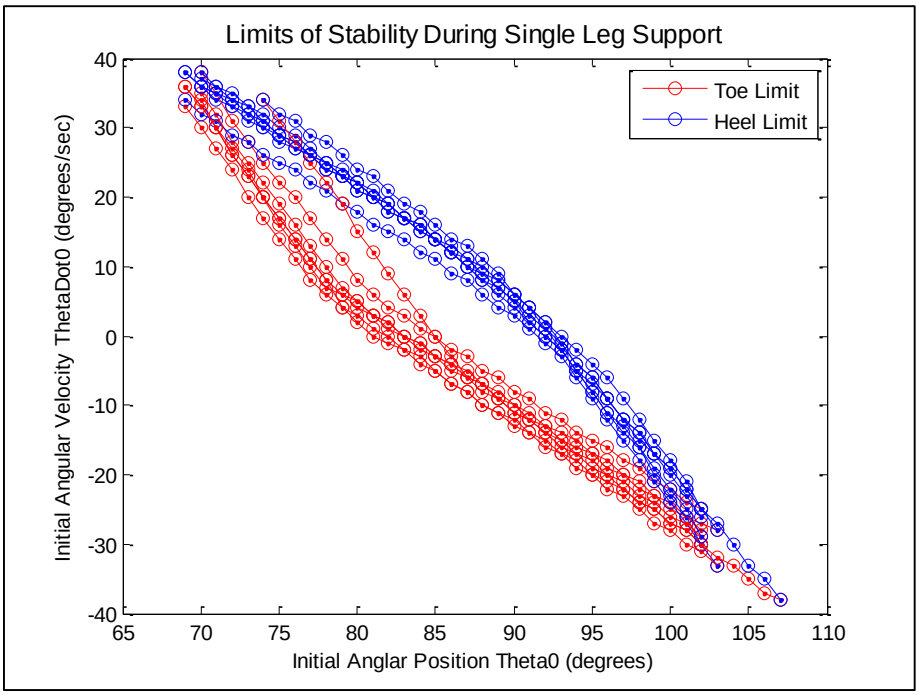

Figure 4-8 The limits and areas of stability (area between toe (red) and heel (blue) limits) for all the subjects within the young age group for the eyes open with no obstacle condition.

The remaining figures for each condition and respective age group are displayed in Appendix C-2. As shown by Figure 4-8, the area of stability results greatly varied between subjects but all stayed within an initial angular position range of approximately 70 to 107 degrees and an initial angular velocity range of -40 to 40 degrees/sec.

4.6 Discussion

This study attempted to develop a relatively simple method to determine the areas of stability (AoS) during the single-leg support phase of gait for both

young and elderly age groups under various different conditions. These areas of stability were bounded by two limits of stability curves. The limits of stability were defined by generating plots of initial angular velocity versus initial angular position input conditions. Hence, any pair of initial conditions which fell within the two limits of stability curves resulted in subject stability. Any pair of initial conditions that was located above the top curve meant that the subject would fall backwards under such circumstances. Similarly, any pair of initial conditions that was located below the bottom curve implied that the subject would fall forwards.

The individual limits of stability points, which collectively formed the limits of stability curves, were wholly dependent on the definition of the XcoM. If the XcoM's position left the base of support during the single-leg support phase for a particular set of input conditions, then these input values were recorded as instable conditions. Inversely, when the XcoM's position remained inside the base of support, those particular sets of input values were recorded as stable conditions thus defining the area of stability (deg^2/sec).

In order to test a wide range of input conditions, a two-dimensional pendulum equation of motion was first derived. The second order differential equation of motion was solved utilizing a Runge-Kutta numerical integrator which required several inputs: the subject mass, the length of the CoM pendulum, the angular position of the CoM pendulum, the torque about the ankle, the ground

reaction forces generated at the CoP, and the horizontal and vertical distances from the CoP to the ankle rotation point.

Before applying this model to find the limits of stability, the model's approximated subsequent angular positions required verification. The model was arbitrarily deemed accurate enough if the percent error differences between the approximated and actual subject CoM pendulum angles were under 10% at each instance in time. Out of the 14 total participants, no subjects had data which failed to produce relatively accurate model approximations. The subjects who were closest to the 10% limit were all within the elderly age group; and the condition that resulted in the least accurate model was the obstacle-clearing condition. Since these elderly subjects decelerated substantially while attempting to pass over the obstacle, these results were not overly surprising [Table 4-2].

Table 4-2 Mean initial CoM X-Direction Velocity values (mm/s) with their respective standard deviation values for all conditions and age groups. Condition 1 = Eyes open with no obstacle. Condition 2 = Eyes open with obstacle. Condition 3 = Eyes closed with no obstacle.

	Condition 1	**Condition 2**	**Condition 3**	**Condition Means**
Young	1457.36 (172.72)	1413.74 (100.30)	1490.85 (193.68)	1453.98 (155.78)
Elderly	1242.44 (90.75)	1182.76 (119.10)	1275.41 (146.18)	1233.54 (121.16)
Age Means	1349.90 (173.22)	1298.25 (159.86)	1383.13 (199.18)	

As shown by the initial CoM x-direction velocity table above, the velocities were consistently lower for each of the conditions at the beginning of the single-leg

stance phase for the elderly age group. These findings may help support the claim that slower walking velocities may indeed lead to decreased stability overall as displayed by previous authors (Chiu et al., In Press; Laufer, 2005; Abella et al., 2005; Cromwell & Newton, 2004).

By once again utilizing IBM's SPSS software, a 2 x 3 factorial ANOVA with two between subjects factors, age group (young and elderly) and condition (eyes open, eyes open with an obstacle, and eyes closed), was utilized to determine the effects of age group and condition on the initial COM x-direction velocity values. The results revealed that there was a significant difference between the young and elderly age group's velocity values ($p=0.008$). Additionally, there was a significant main effect for the conditions ($p=0.017$) as determined by the ANOVA. However, there was no significant interaction between age group and conditions for the velocity values overall ($p=0.946$). Subsequently, Sidak post hoc tests were completed to compare significant ANOVA effects. When velocity values were averaged across both young and elderly age groups, only the obstacle and eyes closed conditions were shown to be significantly different ($p=0.016$). In order to examine these pairwise comparisons more closely, the averages within the age groups were analyzed separately. First, a pairwise comparison was performed between age groups within each condition. For the eyes open with no obstacle condition, the eyes open with an obstacle condition, and the eyes closed with no obstacle condition, the two age groups

were shown to have significantly different velocity values ($p=0.013$, $p=0.002$, $p=0.037$ respectively). When performing a pairwise comparison between conditions within each age group, the three conditions were never shown to have significantly different velocities from one another for either age group.

As shown by the results in Table 4-1, the AoS results were higher for the young subjects in the eyes open without an obstacle (~27% higher) and with an obstacle conditions (~47% higher). Nonetheless, only the AoS values corresponding to the obstacle condition were shown to be significantly different between the two age groups ($p=0.002$). Perhaps additional statistical significance could have been achieved with an increased number of subjects. However, these obstacle-clearing condition results indicated that the young subjects were potentially able to remain balanced when beginning the single-leg support phase at more extreme values of initial angular position and velocity. In addition, the elderly group interestingly showed a larger AoS for the eyes closed with no obstacle condition. This seemed to suggest that the elderly group remained more stable with eyes closed than their younger counterparts. Perhaps their increased experience with walking during periods of limited visual feedback, due to age, taught their bodies to maintain better balance during such circumstances. These results implied that the young group was actually more likely to fall in the dark during the single-leg support phase. Even if young people do experience worse balance in the dark than the elderly, the elderly are still more likely to generate

bruises or broken bones if they fall due to lowered bone density, bone fracture strength, and other age related effects. Additionally, real life falls in the dark often occur due to an unexpected perturbation or trip while attempting to maneuver around a room. Since the elderly have been shown to have a diminished rate of adaptive feedback, the elderly do not adjust their posture as quickly as their young counterparts when such a perturbation occurs (Karamanidis et al., 2011). In this study, their adaptive feedback was not examined since there were no intentional pushes or trips. However, the elderly's greater AoS values, as compared to the young age group's values, and overall higher dynamic stability may have been a result of better feedforward motor commands. However, as mentioned earlier, the results for this condition were not actually significantly different between age groups. Thus, neither age group truly displayed that they were more stable during that particular condition.

When comparing the mean AoS values for the young age group alone, the group was shown to be most stable during the eyes open with no obstacle condition. The young age group was least stable during the eyes open with an obstacle condition as expected. Interestingly, the elderly group was most stable during the eyes closed with no obstacle condition suggesting that perhaps they walked more cautiously while their vision was obscured. Additionally, as mentioned previously, the elderly group's walking direction velocities, as a whole, were consistently lower than the young group. While the young group's

higher walking speeds may have resulted in larger AoS values for the eyes open conditions, the elderly group's overall lower walking speeds resulted in increased stability for the eyes closed condition. As noted previously, the correlation between velocity and stability is very controversial and previous studies sometimes appear to conflict with one another (Abella et al., 2005; Cromwell and Newton, 2004; Dingwell and Marin, 2006; England and Granata, 2007; Li et al., 2005). Hence, increased walking velocity does not necessarily lead to an increased AoS and there are likely more variables to consider. Lastly, the elderly group's lowest AoS value occurred during the obstacle clearing condition as expected since they were often visibly shown to struggle during this particular task.

The model and the study in general have some limitations which must be addressed. The human body is obviously a very complex system and during locomotion there are many moving bodily segments. The model attempted to simplify the body down into a single segment with a single point mass at one end rotating about an ankle joint. During attempted balance recovery, many muscles may be activated, including ones in the arms to attempt to shift the overall body center of mass. Thus, this model greatly simplified the actual human body. Nonetheless, the model only attempted to depict the single-leg stance phase and thus avoided many of the complexities associated with gait.

When determining the limits of stability, the definition of the XcoM was vital. Hof et al. made some very important assumptions when developing the XcoM equation. First, the equation of motion for their simple inverted pendulum was derived assuming the pendulum was always near vertical. During static or near static situations, this assumption is likely valid and should be deemed acceptable. However, since this study involved walking, the CoM pendulum, during the single-leg support phase, was not always near vertical and typically shifted between 100 and 70 degrees. Even so, the $\sin\theta$ values were still relatively close to 1.000 at both extremes (0.985 and 0.940 for 100 and 70 degrees respectively). Additionally, their group also assumed the length of the pendulum was a constant value. In reality, the length of the pendulum changes throughout gait, but during the single-leg support phase, the length of the pendulum remains relatively constant so this assumption was deemed acceptable. Hof et al.'s XcoM equation also included the assumption that the CoP location remained constant. During the gait phase of interest, the CoP actually shifts from under the ankle towards the balls of the feet. Nevertheless, the CoP generally resides under the balls of the feet for the majority of the phase so this assumption seemed plausible. Hence, the main questionable assumption was that the angular position of the CoM pendulum was always near vertical. For simplicity, Hof et al.'s XcoM equation was still utilized but future studies or analyses may decide to alter its definition. Altering the definition would likely generate larger AoS values. In the

definition's current form, instability ensues if a step is taken in the anterior or posterior direction. Since the subjects are walking, steps will definitely occur in the forward direction and hence, according to the current XcoM definition, instability will ensue. Forward steps should not actually be considered instable and thus, an XcoM definition modification would likely shift the limits of stability curve associated with the toe marker upwards. While a step in the anterior direction should not be considered instable while walking, a step posteriorly should still be considered instable behavior. Consequently, the limits of stability curve associated with the heel marker appeared reasonable despite the limitations. However, for the purposes of developing a tool to assess stability differences between various age groups and conditions, the definition that Hof et al. provided for the XcoM seemed to suffice. Even in the equation's current form, the AoS was still a great comparison value across the different age groups and conditions.

Along with the model and the XcoM definition, the subject trials also had various limitations. In each of the subject trials, the walking speed was not well controlled across participants. Some of the elderly subjects, in particular, slowed their gait during obstacle crossing and when eyes were closed in order to ensure the maintenance of balance. In order to resolve this issue, future studies could possibly attempt to control the speed of participants in order to maintain a consistent velocity across both age groups. However, increasing the speed of the elderly may be challenging or make them feel uncomfortable. Decreasing the

speed of the young subjects may result in unorthodox gait patterns. Therefore, these methods may not yield the ideal solution; but velocity normalization has been utilized in past literature and might present a promising solution. Several authors have demonstrated the value of normalizing the linear velocity via use of a dimensionless Froude number, $v^2 * L^{-1} * g^{-1}$(Carty et al., 2009; Hof, 1996; Pierrynowski et al., 2001). This value would certainly help account for the discrepancies seen between subjects' CoM pendulum lengths, and may also help resolve issues with the AoS values if the XcoM definition contains this normalized velocity definition. In addition to all the previous limitations, all of the subjects were from the Arlington, TX community and seemed fairly representative of the general population, but people from different regions would have possibly generated a better representation. Nonetheless, for the purposes of developing a model, the group of subjects chosen seemed perfectly adequate.

In addition, while the force plate and camera data were filtered by intelligent means, no filter is perfect. Filters nearly always eliminate some of the relevant data while attempting to eliminate extraneous noise. Thus, the resulting data may have been slightly altered due to the filtering methods, but this error was hopefully minimized by the respective types of filters chosen for both applications.

The AoS findings help illuminate some of the differences seen between these two age groups during various walking conditions. Even if it is unlikely for

a person to enter the single-leg support phase with some of the more extreme initial conditions, the AoS results help to provide a general measure of stability that may be utilized to compare various different walking situations. Applications of these methods may extend into the research of balance with prosthetics or in humanoid robotics. In addition, with slight alterations to definition of the XcoM, the equation of motion, and the code associated with those variables, the MATLAB AoS code could be expanded upon to better match different static or dynamic movements. The addition of the third axis would also generate the ability to measure stability in the medio-lateral directions but would likely require fairly extensive coding modifications. However, even without such changes, the developed model provides a relatively simple AoS comparison value that can be utilized to determine which particular subjects are more or less likely to fall for various conditions.

Appendix A

Statement of Informed Consent

APPROVED

MAR 27 2013 MAR 27 2014

Institutional Review Board

UT Arlington
Informed Consent Document

PRINCIPAL INVESTIGATOR
Mark D. Ricard, Kinesiology Department, 817-272-0764, Ricard@uta.edu.

TITLE OF PROJECT
Dynamic balance in walking.

INTRODUCTION

You are being asked to participate in a research study on the effects of walking: with eyes open, while stepping over an obstacle, and with eyes closed on dynamic balance. Your participation is voluntary. Refusal to participate or discontinuing your participation at any time will involve no penalty or loss of benefits to which you are otherwise entitled. Please ask questions if there is anything you do not understand.

PURPOSE
The purpose of this research is to compare dynamic balance in young and elderly males when walking under the following conditions: eyes open, eyes open while stepping over an obstacle and eyes closed with no obstacle.

DURATION
You will be asked to participate in 1 study visit which will last approximately 60 minutes.

NUMBER OF PARTICIPANTS
The number of anticipated participants in this research study is 60.

PROCEDURES

The procedures which will involve you as a research participant include:

1. Report to the Biomechanics and Movement Studies Laboratory, Maverick Activities Center, room 132.
2. You will then be required to change into black spandex shorts and sleeveless t-shirts provided by researchers. You will then be asked to remove your shoes and socks. One half inch reflective markers will be attached on the right and left side of your foot, heel, ankle, lower leg, knee, thigh, shoulder, upper arm, elbow, and wrist using double sided tape. In addition you will be asked to wear a spandex hat so that two reflective markers can be placed over the front and back of your head.
3. Your height, weight, ankle width, knee width and shoulder width will then be measured. You will then be filmed while standing still on the force plate for 3 seconds.
4. You will then be asked to practice walking so that your right foot lands on force plate #1 and your left foot lands on force plate #2 using your normal walking

IRB Approval Date:

IRB Expiration Date:

speed. Once you are able to walk with your normal walking speed and step on plate 1 with the right foot and plate 2 with the left foot ten trials of walking will be recorded with the video system in each of the following conditions: (1) eyes open, (2) eyes open while stepping over a 4 inch obstacle between plate 1 and 2, (3) eyes closed with no obstacle. If your right and left foot do not land on the force plate the trial will be discarded and another walking trial will be recorded.

5. After you have completed ten walking trials for each of the three conditions the reflective markers will be removed and your role in the study will be completed.
6. The video recordings and force recordings of your walking motion will then be used to compute your dynamic balance during each of the three walking conditions.

POSSIBLE BENEFITS

There are no direct benefits to you for being in the study. We hope that we will learn how dynamic balance changes in young and elderly subjects when walking over an obstacle or while walking with eyes closed.

POSSIBLE RISKS/DISCOMFORTS

The potential risks of participating in this research study are that you could sustain an injury such as a fall or strain while walking. Should you experience any discomfort please inform the researcher or his staff and they will help you to get necessary medical care, if needed. You have the right to quit any study procedures at any time at no consequence.

COMPENSATION
No compensation will be offered for participation in this study.

ALTERNATIVE PROCEDURES
There are no alternative procedures offered for this study. However, you can elect not to participate in the study or quit at any time at no consequence.

COMPENSATION FOR MEDICAL TREATMENT
The University of Texas at Arlington (UTA) will pay the cost of emergency first aid for an injury that occurs as a result of your participation in this study. UTA will not pay for any other medical treatment. Claims against UTA or any of its agents or employees may be submitted according to the Texas Tort Claims Act (TTCA). These claims may be settled to the extent allowable by the state law as provided under the TTCA, (Tex. Civ. Prac. & Rem. Code, secs, 101.001, et seq.). For more information about claims, you may contact the Chairman of the Institutional Review Board of UTA at 817-272-3723.

IRB Approval Date: APPROVED

IRB Expiration Date: MAR 27 2013 MAR 27 2014

Institutional Review Board

UT Arlington
Informed Consent Document

VOLUNTARY PARTICIPATION
Participation in this research study is voluntary. You have the right to decline participation in any or all study procedures or quit at any time at no consequence.

CONFIDENTIALITY
Every attempt will be made to see that your study results are kept confidential. A copy of this signed consent form and all data collected from this study will be stored in the Maverick Activities Center, MAC 230, (Mark Ricard's office) for at least three (3) years after the end of this research. The results of this study may be published and/or presented at meetings without naming you as a participant. Additional research studies could evolve from the information you have provided, but your information will not be linked to you in anyway; it will be anonymous. Although your rights and privacy will be maintained, the Secretary of the Department of Health and Human Services, the UTA Institutional Review Board (IRB), and personnel particular to this research have access to the study records. Your records will be kept completely confidential according to current legal requirements. They will not be revealed unless required by law, or as noted above. The IRB at UTA has reviewed and approved this study and the information within this consent form. If in the unlikely event it becomes necessary for the Institutional Review Board to review your research records, the University of Texas at Arlington will protect the confidentiality of those records to the extent permitted by law.

CONTACT FOR QUESTIONS
If you have any questions about this research study they may be directed to Mark Ricard at 817-272-0764, or Christopher Ray at 817-272-0082. Any questions you may have about your rights as a research participant or a research-related injury may be directed to the Office of Research Administration; Regulatory Services at 817-272-2105 or regulatoryservices@uta.edu.

As a representative of this study, I have explained the purpose, the procedures, the benefits, and the risks that are involved in this research study:

_____
Signature and printed name of principal investigator or person obtaining consent **Date**

CONSENT

By signing below, you confirm that you are 18 years of age or older and have read or had this document read to you. You have been informed about this study's purpose, procedures, possible benefits and risks, and you have received a copy of this form. You

IRB Approval Date: APPROVED MAR 27 2013

IRB Expiration Date: MAR 27 2014

Institutional Review Board

have been given the opportunity to ask questions before you sign, and you have been told that you can ask other questions at any time.

You voluntarily agree to participate in this study. By signing this form, you are not waiving any of your legal rights. Refusal to participate will involve no penalty or loss of benefits to which you are otherwise entitled. You may discontinue participation at any time without penalty or loss of benefits, to which you are otherwise entitled.

SIGNATURE OF VOLUNTEER **DATE**

APPROVED

IRB Approval Date: MAR 27 2013 MAR 27 2014 4

IRB Expiration Date: Institutional Review Board

Appendix B

Vicon Information

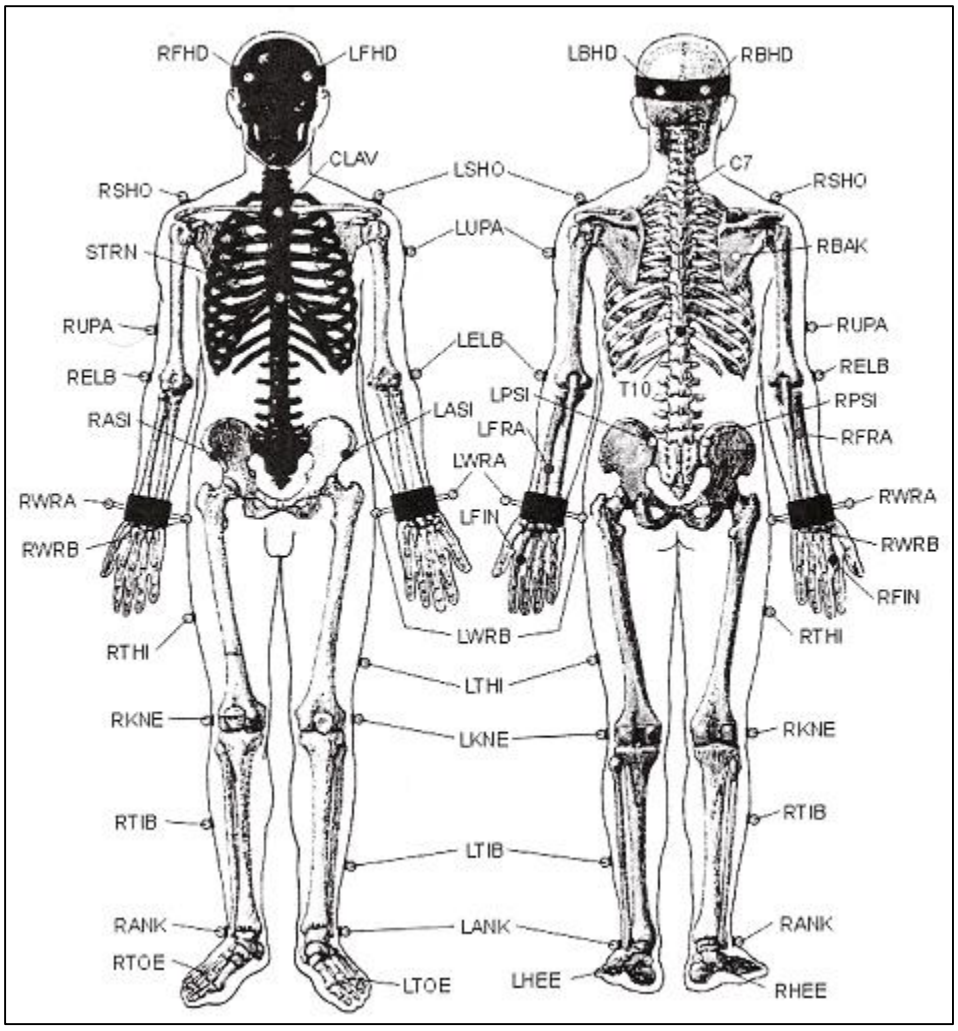

Figure B-1 Vicon's Plug-in-Gait Marker Placement. The locations of each marker utilized for each subject are displayed (Vicon, Oxford, UK).

Body Region	Abbreviation	Location	Description
Pelvis	LASI	Left ASIS	Placed directly over the left anterior superior iliac spine
	RASI	Right ASIS	Placed directly over the right anterior superior iliac spine
	LPSI	Left PSIS	Placed directly over the left posterior superior iliac spine
	RPSI	Right PSIS	Placed directly over the right posterior superior iliac spine
Leg	LKNE	Left Knee	Placed on the lateral epicondyle of the left knee
	RKNE	Right Knee	Placed on the lateral epicondyle of the right knee
	LTHI	Left Thigh	Placed on lower lateral 1/3 surface of thigh below swing of hand
	RTHI	Right Thigh	Placed on lower lateral 1/3 surface of thigh below swing of hand
	LANK	Left Ankle	Placed on lateral malleolus
	RANK	Right Ankle	Placed on lateral malleolus
	LTIB	Left Tibial Wand	Placed on lower 1/3 of shank
	RTIB	Right Tibial Wand	Placed on lower 1/3 of shank
Foot	LTOE	Left Toe	Placed over second metatarsal head
	RTOE	Right Toe	Placed on second metatarsal head
	LHEE	Left Heel	Placed on the calcaneus at same height above the plantar surface of the foot as the toe marker
	RHEE	Right Heel	Placed on the calcaneus at same height above the plantar surface of the foot as the toe marker

Table B-1 Lower extremity subject marker placements. Highlighted markers are utilized by the MATLAB code to determine the limits of stability. Additional markers are required to run the code (CoM and CoP marker positions), but these positions are calculated via the Vicon Workstation software.

Appendix C

Subject Data and Result Figures

Appendix C-1 Subject Data

Table C-1 Individual subject and condition area of stability and maximum theta percent error results. Condition 1 = Eyes open with no obstacle. Condition 2 = Eyes open with obstacle. Condition 3 = Eyes closed with no obstacle.

Subject	Young or Elderly	Age	Mass (kg)	Height (cm)	Condition	Area of Stability (deg^2/sec)	Model Maximum Percent Error Theta (%)	Initial X-Direction Velocity (mm/s)
20	Y	24	75	185.4	1	147.166667	2.62767	1411.036912
					2	122.666667	5.131327	1407.585607
					3	154	4.556444	1530.155924
21	Y	26	77	188	1	156.666667	1.006321	1507.844538
					2	119.166667	5.14622	1462.544455
					3	115.333333	0.49993	1480.644054
22	Y	27	83	175	1	182.083333	4.624762	1555.174716
					2	123.166667	4.294183	1408.337686
					3	140.666667	1.964617	1519.848207
38	Y	23	73.482	177.8	1	172.125	2.507798	1225.001453
					2	130.958333	2.12544	1427.118684
					3	155	1.506491	1462.49821
40	Y	29	93.337	187.96	1	131.333333	4.805312	1360.410401
					2	98	7.828831	1335.739923
					3	124	3.906487	1194.14325
42	Y	23	67.13	170.18	1	94.666667	3.035052	1765.321871
					2	101.833333	2.33666	1587.538652
					3	60	4.386017	1845.805822
44	Y	28	67.132	172.72	1	183.375	4.259658	1376.752428
					2	124.333333	5.495642	1267.313714
					3	157.666667	2.723393	1402.863295
30	E	66	77	169	1	147.666667	3.620904	1142.539021
					2	99.333333	9.977437	1108.092283
					3	131.666667	3.546415	1130.59513

31	E	73	65	159	1	85.333333	2.238991	1350.254888
					2	104.708333	6.174376	1253.943008
					3	166.333333	4.955457	1378.185009
32	E	69	93	154	1	141.666667	6.921156	1296.29332
					2	99.875	9.123201	1274.673666
					3	135.666667	8.567988	1341.011369
33	E	82	72	177	1	90.666667	5.333391	1347.985719
					2	55.666667	7.180954	1354.643816
					3	167.625	7.616402	1512.574075
43	E	77	83.688	172.72	1	163.375	3.791819	1233.671643
					2	70	5.782413	1030.378971
					3	130.75	4.607212	1105.86152
45	E	83	93.44	182.88	1	97.625	0.826656	1139.535309
					2	66.041667	3.500425	1067.820036
					3	114.666667	0.535848	1189.630064
46	E	72	90.718	182.8	1	117	1.607161	1186.767031
					2	75.375	1.028931	1189.795329
					3	120.125	0.592156	1269.989567

Appendix C-2 Result Figures

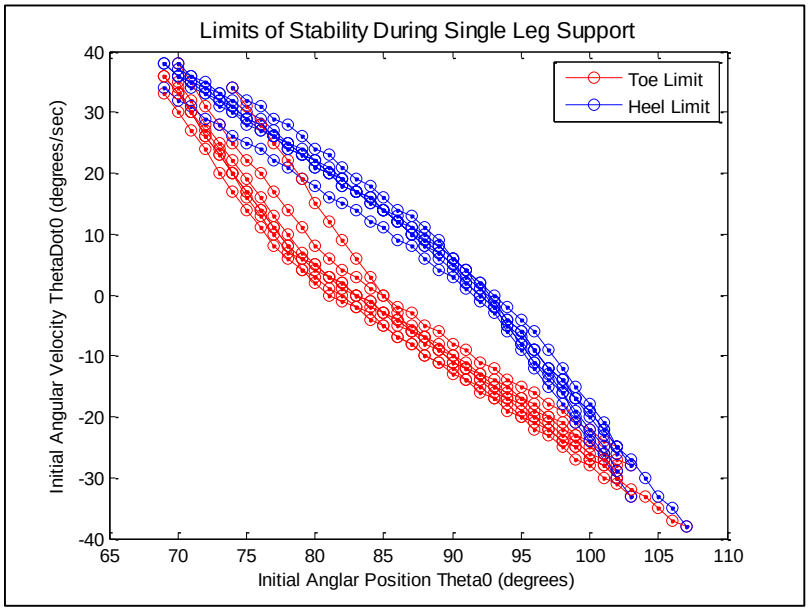

Figure C-2a The limits and areas of stability for all the subjects within the young age group for the eyes open with no obstacle condition.

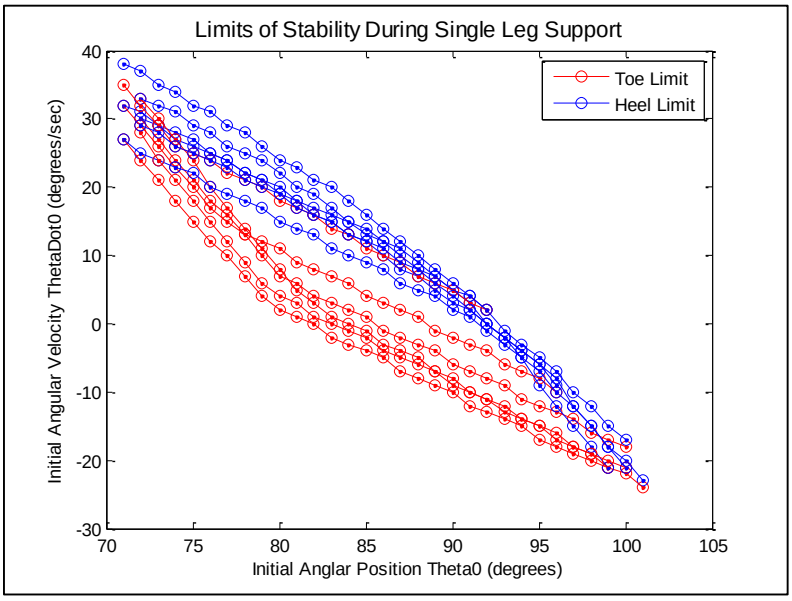

Figure C-2b The limits and areas of stability for all the subjects within the young age group for the eyes open with an obstacle condition.

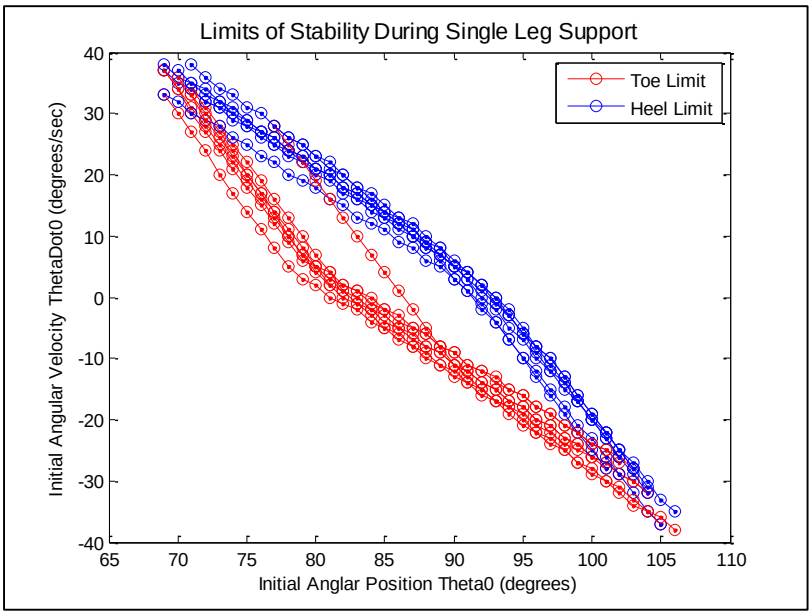

Figure C-2c The limits and areas of stability for all the subjects within the young age group for the eyes closed with no obstacle condition.

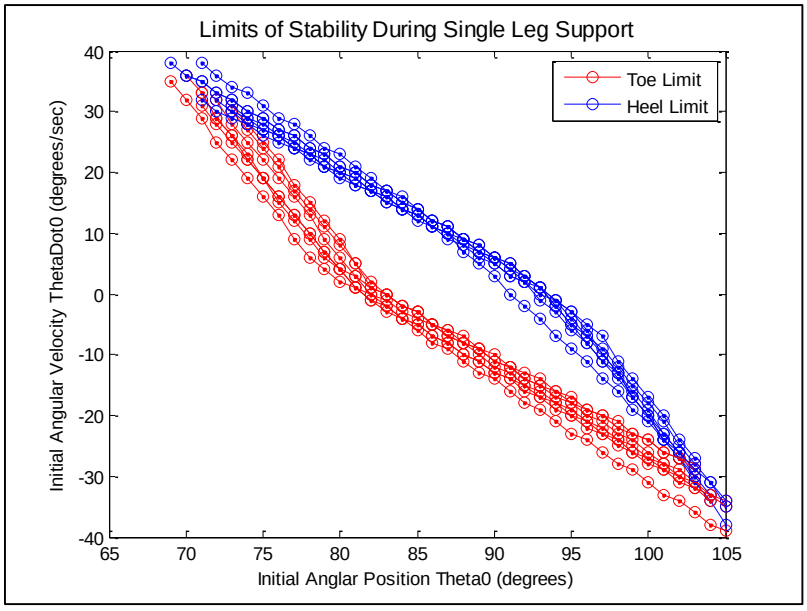

Figure C-2d The limits and areas of stability for all the subjects within the elderly age group for the eyes open with no obstacle condition.

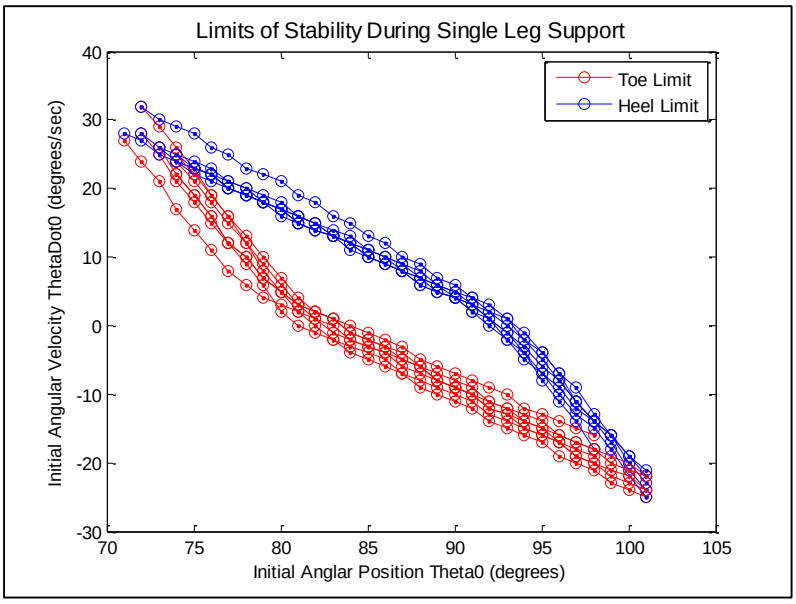

Figure C-2e The limits and areas of stability for all the subjects within the elderly age group for the eyes open with an obstacle condition.

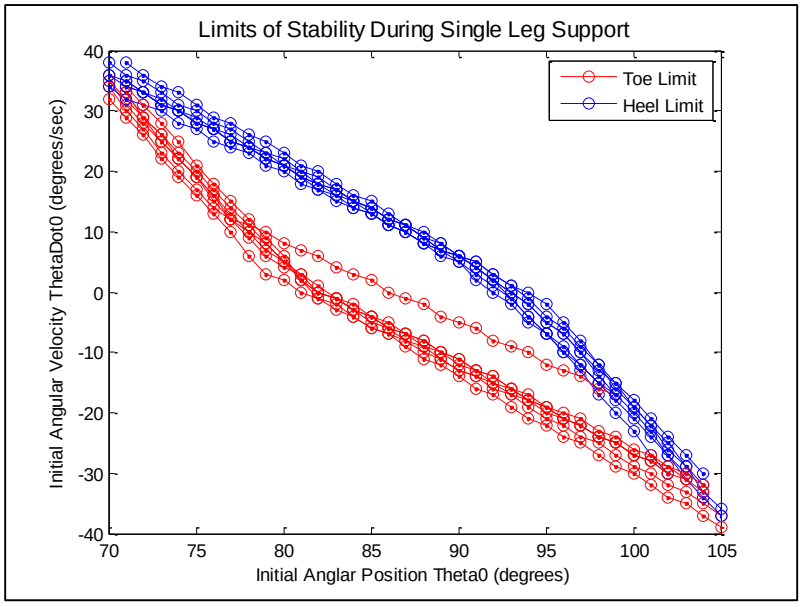

Figure C-2f The limits and areas of stability for all the subjects within the elderly age group for the eyes closed with no obstacle condition.

Appendix D

MATLAB Code

Appendix D-1 Animate3D

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                    ** ABOUT THE CODE **
%
% Animate3D
%
% This code animates the motion of the subject reflective marker data
%
% Author: Clifford Hancock
% Date: 2/28/2013
%
% This code utilizes actual subject reflective marker data.
% Subjects were asked to walk down a walkway with two force plates while
% cameras recorded 3-D locations of reflective markers.
%
% The code does NOT filter the marker position data.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%               ** HOW TO SET UP DATA MATRICES **
%
% This code requires data matrices to be in a specific format to run
% properly.
%
% The reflective marker data should be saved as:
%
% MarkersCondition#Subject# (e.g. MarkersCondition1Subject1)
%
% where the # values can change depending upon the condition or subject
% numbers (e.g. MarkersCondition4Subject2 would be the .mat file containing
% the 2nd subject and his 4th condition).
%
% These MarkersCondition#Subject# .mat files should only contain number
% values (no words) and have the proper columns as follows:
%
% (NOTE: X = Walking Direction, Z = Vertical Direction)
%
% Column 1 - Any Marker X Coordinates
% Column 2 - Any Marker Y Coordinates
```

```
% Column 3 - Any Marker Z Coordinates
% Column 4 - Any Marker X Coordinates
% Column 5 - Any Marker Y Coordinates
% Column 6 - Any Marker Z Coordinates
% Repeat the X,Y,Z cycle until all markers are present
%
% Use the entire data set (all rows) for each column, the code can handle
% blank boxes without numbers (NaN).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Prompt the user to input the appropriate subject number
fprintf('\n\n Which SUBJECT would you like to analyze?')
fprintf('\n')
fprintf('(Insert value and press enter)')
fprintf('\n')
SubjectSelector = input('Subject #');
fprintf('\n')
if isempty(SubjectSelector)
   SubjectSelector = 1;   %Simply pressing "enter" with no user input will utilize
subject 1's file
   fprintf('Subject #1 was selected due to no user input!')
   fprintf('\n\n\n')
end

SubjectSelector=round(SubjectSelector);    %rounds any decimal values to nearest
integer
fprintf('Subject #')
fprintf('%0.0f',SubjectSelector)
fprintf(' was selected...')
fprintf('\n\n')


if SubjectSelector<=0
   fprintf('Subject numbers cannot be negative and must be above zero!')
   fprintf('\n\n\n')
```

```
end

%Prompt the user to input the appropriate condition number
fprintf('Which CONDITION would you like to analyze?')
fprintf('\n')
fprintf('(Insert value and press enter)')
fprintf('\n')
ConditionSelector = input('Condition #');
fprintf('\n')
if isempty(ConditionSelector)
   ConditionSelector = 1;   %Simply pressing "enter" with no user input will
utilize subject 1's file
   fprintf('Condition #1 was selected due to no user input!')
   fprintf('\n\n\n')
end

ConditionSelector=round(ConditionSelector);    %rounds any decimal values to
nearest integer
fprintf('Condition #')
fprintf('%0.0f',ConditionSelector)
fprintf(' was selected...')
fprintf('\n\n')


if ConditionSelector<=0
   fprintf('Condition numbers cannot be negative and must be above zero!')
   fprintf('\n\n\n')
end

MarkersNumber = ['MarkersCondition' int2str(ConditionSelector) 'Subject'
int2str(SubjectSelector)];
Markers = load(MarkersNumber);

Markers = struct2cell(Markers);
Markers = cell2mat(Markers);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Convert Markers
columncount=size(Markers,2);    %Finds number of columns in Markers
j=1;
```

```
for i=1:3:columncount    %i only selects x components from Markers (want every
3rd column (XyzXyzXyz)
   Markersadj(:,j,1)=Markers(:,i); %j makes each column of Markersadj a
different marker (hip/knee/ankle...)
   j=j+1;
end

j=1;
for i=2:3:columncount    %i only selects y components from Markers (want every
3rd column (xYzxYzxYz)
   Markersadj(:,j,2)=Markers(:,i);
   j=j+1;
end

j=1;
for i=3:3:columncount    %i only selects z components from Markers (want every
3rd column (xyZxyZxyZ)
   Markersadj(:,j,3)=Markers(:,i);
   j=j+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% markerviewer
% This script views marker positions in the global coordinate system.
close all;
numofmarkers=size(Markersadj,2);
fprintf('Press "enter" to view animation...')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% setup figure to view data

maxx=max(max(Markersadj(:,:,1)));    %Find x vector containing largest value,
then finds max value
minx=min(min(Markersadj(:,:,1)));    %Find x vector with smallest value, then
finds min value
maxz=max(max(Markersadj(:,:,3)));    %Find z vector containing largest value,
then finds max value
minz=min(min(Markersadj(:,:,3)));    %Find z vector with smallest value, then
finds min value
```

```
miny=min(min(Markersadj(:,:,2)));    %Find y vector with smallest value, then
finds min value
maxy=max(max(Markersadj(:,:,2)));    %Find y vector with smallest value, then
finds min value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plots marker data (side view)

for i=1:1:size(Markersadj,1),

plot3(Markersadj(i,1:numofmarkers,1),Markersadj(i,1:numofmarkers,2),Markersa
dj(i,1:numofmarkers,3),'ko');
   title('3D Animation')
   xlabel('X Marker Position (mm)')
%     ylabel('Y Marker Position (mm)')
   zlabel('Z Marker Position (mm)')
   axis equal;
   axis([minx maxx miny maxy minz maxz]);
   text(minx+100,.1,num2str(i));
   drawnow;
   if i==1
   pause;
   end
   hold off;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% clear extraneous variables

clearvars -except Markersadj
```

Appendix D-2 AnimateSagittal

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       ** ABOUT THE CODE **
%
% AnimateSagittal
%
% This code animates the motion of the subject reflective marker data
%
% Author: Clifford Hancock
% Date: 2/16/2013
%
% This code utilizes actual subject reflective marker data.
% Subjects were asked to walk down a walkway with two force plates while
% cameras recorded 3-D locations of reflective markers.
%
% The code does NOT filter the marker position data.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                 ** HOW TO SET UP DATA MATRICES **
%
% This code requires data matrices to be in a specific format to run
% properly.
%
% The reflective marker data should be saved as:
%
% MarkersCondition#Subject# (e.g. MarkersCondition1Subject1)
%
% where the # values can change depending upon the condition or subject
% numbers (e.g. MarkersCondition4Subject2 would be the .mat file containing
% the 2nd subject and his 4th condition).
%
% These MarkersCondition#Subject# .mat files should only contain number
% values (no words) and have the proper columns as follows:
%
% (NOTE: X = Walking Direction, Z = Vertical Direction)
%
% Column 1 - Any Marker X Coordinates
% Column 2 - Any Marker Y Coordinates
```

```
% Column 3 - Any Marker Z Coordinates
% Column 4 - Any Marker X Coordinates
% Column 5 - Any Marker Y Coordinates
% Column 6 - Any Marker Z Coordinates
% Repeat the X,Y,Z cycle until all markers are present
%
% Use the entire data set (all rows) for each column, the code can handle
% blank boxes without numbers (NaN).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Prompt the user to input the appropriate subject number
fprintf('\n\n Which SUBJECT would you like to analyze?')
fprintf('\n')
fprintf('(Insert value and press enter)')
fprintf('\n')
SubjectSelector = input('Subject #');
fprintf('\n')
if isempty(SubjectSelector)
   SubjectSelector = 1;   %Simply pressing "enter" with no user input will utilize
subject 1's file
   fprintf('Subject #1 was selected due to no user input!')
   fprintf('\n\n\n')
end

SubjectSelector=round(SubjectSelector);    %rounds any decimal values to nearest
integer
fprintf('Subject #')
fprintf('%0.0f',SubjectSelector)
fprintf(' was selected...')
fprintf('\n\n')


if SubjectSelector<=0
   fprintf('Subject numbers cannot be negative and must be above zero!')
   fprintf('\n\n\n')
```

```
end

%Prompt the user to input the appropriate condition number
fprintf('Which CONDITION would you like to analyze?')
fprintf('\n')
fprintf('(Insert value and press enter)')
fprintf('\n')
ConditionSelector = input('Condition #');
fprintf('\n')
if isempty(ConditionSelector)
   ConditionSelector = 1;   %Simply pressing "enter" with no user input will
utilize subject 1's file
   fprintf('Condition #1 was selected due to no user input!')
   fprintf('\n\n\n')
end

ConditionSelector=round(ConditionSelector);    %rounds any decimal values to
nearest integer
fprintf('Condition #')
fprintf('%0.0f',ConditionSelector)
fprintf(' was selected...')
fprintf('\n\n')


if ConditionSelector<=0
   fprintf('Condition numbers cannot be negative and must be above zero!')
   fprintf('\n\n\n')
end

MarkersNumber = ['MarkersCondition' int2str(ConditionSelector) 'Subject'
int2str(SubjectSelector)];
Markers = load(MarkersNumber);

Markers = struct2cell(Markers);
Markers = cell2mat(Markers);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Convert Markers
columncount=size(Markers,2);    %Finds number of columns in Markers
j=1;
```

```
for i=1:3:columncount    %i only selects x components from Markers (want every
3rd column (XyzXyzXyz)
   Markersadj(:,j,1)=Markers(:,i); %j makes each column of Markersadj a
different marker (hip/knee/ankle...)
   j=j+1;
end

j=1;
for i=2:3:columncount    %i only selects y components from Markers (want every
3rd column (xYzxYzxYz)
   Markersadj(:,j,2)=Markers(:,i);
   j=j+1;
end

j=1;
for i=3:3:columncount    %i only selects z components from Markers (want every
3rd column (xyZxyZxyZ)
   Markersadj(:,j,3)=Markers(:,i);
   j=j+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% markerviewer
% This script views marker positions in the global coordinate system.
close all;
numofmarkers=size(Markersadj,2);
fprintf('Press "enter" to view animation...')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% setup figure to view data

maxx=max(max(Markersadj(:,:,1)));    %Find x vector containing largest value,
then finds max value
minx=min(min(Markersadj(:,:,1)));    %Find x vector with smallest value, then
finds min value
maxz=max(max(Markersadj(:,:,3)));    %Find z vector containing largest value,
then finds max value
minz=min(min(Markersadj(:,:,3))-100);    %Find z vector with smallest value,
then finds min value
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plots marker data (side view)

for i=1:1:size(Markersadj,1),
   plot(Markersadj(i,1:numofmarkers,1),Markersadj(i,1:numofmarkers,3),'ko');
   title('Sagittal Plane Animation')
   xlabel('X Marker Position (mm)')
   ylabel('Z Marker Position (mm)')
   axis equal;
   axis([minx maxx minz maxz]);
   text((minx+100),.1,num2str(i));
   drawnow;
   if i==1
   pause;
   end
   hold off;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% clear extraneous variables

clearvars -except Markersadj
```

Appendix D-3 ButterworthLowPassFilter

```
function [ smooth ] = ButterworthLowPassFilter3Dcoords( raw, fs, fc )
%ButterworthLowPassFilter this algorithm is a 2nd order recursive
%Butterworth low pass filter. The data are filtered forward and backwards
%to eliminate the phase shift caused by the filter, as a result it should
%be described as a 4th order recursive Butterworth zero phase shift filter.
% The filter coefficients are corrected for the number of passes (two) thru
% the filter. See p 68 - 70 of David Winter's book:

% David A. Winter, (2009) Biomechanics and Motor Control of Human
Movement,
% 4th edition, John Wiley & Sons, Inc, ISBN 978-0-470-39818-0.

%   raw[1:nframes, 1:3] input noisy signal to be smoothed
%   raw[1:nframes, x y z]  each row contains x y z coords for a frame.
%   fs sampling frequency in (Hz) of raw[] signal
%   fc cutoff frequency in (Hz)
%   smooth[] the function returns the smoothed data array

% fs=120 fc=6 for Markers


%  ---------------------  Compute Filter Coefficients --------------------
nPasses = 2;
cw = (2 ^(1/nPasses) - 1) ^ 0.25;
wc = (tan(pi * fc / fs)) / cw;
k1 = sqrt(2) * wc;
k2 = wc * wc;
a0 = k2 / (1.0 + k1 + k2);
a1 = 2 * a0;
a2 = a0;
k3 = 2 * a0 / k2;
b1 = -2 * a0 + k3;
b2 = 1 - 2 * a0 - k3;
%  ---------------------- pre dimension arrays --------------------------
n = length(raw);
[nrows, ncols] = size(raw);

smooth = zeros(nrows,ncols);
temp = zeros(n + 4,1);
```

```
prime = zeros(n + 4,1);

for col = 1 : ncols
   temp(1) = raw(1,col) + (raw(1,col) - raw(2,col));
   temp(2) = raw(1,col) + (raw(1,col) - raw(3,col));
   temp(n + 4) = raw(n,col) + (raw(n,col) - raw(n - 1,col));
   temp(n + 3) = raw(n,col) + (raw(n,col) - raw(n - 2,col));

   for p = 1 : n
      temp(p+2) = raw(p,col);
   end

   for p = 1 : n + 4
      prime(p) = temp(p);
   end

   for p = 3 : n + 4
      prime(p) = a0 * temp(p) + a1 * temp(p - 1) + a2 * temp(p - 2) + b1 *
prime(p-1) + b2 * prime(p-2);      % Forward Pass
   end

   for p = 1 : n + 4
      temp(p) = prime(p);
   end

   for p = n + 2 : -1 : 1
      prime(p) = a0 * temp(p) + a1 * temp(p + 1) + a2 * temp(p + 2) + b1 *
prime(p + 1) + b2 * prime(p + 2);   % Backward Pass
   end

   for p = 1 : n
      smooth(p,col) = prime(p + 2);   % Return the smoothed signal
   end
end
end
```

Appendix D-4 CriticallyDampedButterworthLowPass

```
function [ smooth ] = CriticallyDampedButterworthLowPass( raw, fs, fc )
%CriticallyDampedButterworthLowPass this algorithm is a 20th order recursive
% Butterworth critically damped low pass filter.
% The data are filtered forward and backwards to eliminate the phase shift
% caused by the filter, as a result it should
% be described as a 4th order recursive Butterworth zero phase shift filter.
% The filter coefficients are corrected for the number of passes (two) thru
% the filter. See p 68 - 70 of David Winter's book:

% David A. Winter, (2009) Biomechanics and Motor Control of Human
Movement,
% 4th edition, John Wiley & Sons, Inc, ISBN 978-0-470-39818-0.

% Robertson DG, Dowling JJ (2003) Design and responses of Butterworth and
% critically damped digital filters. J Electromyograph & Kinesiol;
% 13, 569 - 573.

%   raw[] input noisy signal to be smoothed
%   fs sampling frequency in (Hz) of raw[] signal
%   fc cutoff frequency in (Hz)
%   smooth[] the function returns the smoothed data array

%  ---------------------  Compute Filter Coefficients -------------------
nPasses = 5;                    % 5 double passes gives 20th order
c = sqrt(2^(1/(2*nPasses))-1);   % see Winter 4th ed p 70
wc = (tan(pi * fc / fs)) / c;
k1 = 2 * wc;
k2 = wc * wc;
a0 = k2 / (1.0 + k1 + k2);
a1 = 2 * a0;
a2 = a0;
k3 = 2 * a0 / k2;
b1 = -2 * a0 + k3;
b2 = 1 - 2 * a0 - k3;
%  ---------------------- pre dimension arrays --------------------------
n = length(raw);
smooth = 1:n;
temp = zeros(n + 4,1);
prime = zeros(n + 4,1);
```

```
temp(1) = raw(1) + (raw(1) - raw(2));
temp(2) = raw(1) + (raw(1) - raw(3));
temp(n + 4) = raw(n) + (raw(n) - raw(n - 1));
temp(n + 3) = raw(n) + (raw(n) - raw(n - 2));

for p = 1 : n
   temp(p+2) = raw(p);
end

for p = 1 : n + 4
   prime(p) = temp(p);
end
%  ---------------------  Double Pass 1  --------------------------------
temp(1) = raw(1) + (raw(1) - raw(2));
temp(2) = raw(1) + (raw(1) - raw(3));
temp(n + 4) = raw(n) + (raw(n) - raw(n - 1));
temp(n + 3) = raw(n) + (raw(n) - raw(n - 2));

for p = 1 : n
   temp(p+2) = raw(p);
end

for p = 1 : n + 4
   prime(p) = temp(p);
end

for p = 3 : n + 4
   prime(p) = a0 * temp(p) + a1 * temp(p - 1) + a2 * temp(p - 2) + b1 * prime(p-
1) + b2 * prime(p-2);      % Forward Pass
end

for p = 1 : n + 4
   temp(p) = prime(p);
end

for p = n + 2 : -1 : 1
   prime(p) = a0 * temp(p) + a1 * temp(p + 1) + a2 * temp(p + 2) + b1 * prime(p
+ 1) + b2 * prime(p + 2);   % Backward Pass
end
```

```
%  ---------------------  Double Pass 2  --------------------------------
temp(1) = raw(1) + (raw(1) - raw(2));
temp(2) = raw(1) + (raw(1) - raw(3));
temp(n + 4) = raw(n) + (raw(n) - raw(n - 1));
temp(n + 3) = raw(n) + (raw(n) - raw(n - 2));

for p = 1 : n
   temp(p+2) = raw(p);
end

for p = 1 : n + 4
   prime(p) = temp(p);
end

for p = 3 : n + 4
   prime(p) = a0 * temp(p) + a1 * temp(p - 1) + a2 * temp(p - 2) + b1 * prime(p-
1) + b2 * prime(p-2);      % Forward Pass
end

for p = 1 : n + 4
   temp(p) = prime(p);
end

for p = n + 2 : -1 : 1
   prime(p) = a0 * temp(p) + a1 * temp(p + 1) + a2 * temp(p + 2) + b1 * prime(p
+ 1) + b2 * prime(p + 2);   % Backward Pass
end
%  ---------------------  Double Pass 3  --------------------------------
temp(1) = raw(1) + (raw(1) - raw(2));
temp(2) = raw(1) + (raw(1) - raw(3));
temp(n + 4) = raw(n) + (raw(n) - raw(n - 1));
temp(n + 3) = raw(n) + (raw(n) - raw(n - 2));

for p = 1 : n
   temp(p+2) = raw(p);
end

for p = 1 : n + 4
   prime(p) = temp(p);
end
```

```
for p = 3 : n + 4
   prime(p) = a0 * temp(p) + a1 * temp(p - 1) + a2 * temp(p - 2) + b1 * prime(p-
1) + b2 * prime(p-2);      % Forward Pass
end

for p = 1 : n + 4
   temp(p) = prime(p);
end

for p = n + 2 : -1 : 1
   prime(p) = a0 * temp(p) + a1 * temp(p + 1) + a2 * temp(p + 2) + b1 * prime(p
+ 1) + b2 * prime(p + 2);   % Backward Pass
end
%   ---------------------  Double Pass 4  --------------------------------
temp(1) = raw(1) + (raw(1) - raw(2));
temp(2) = raw(1) + (raw(1) - raw(3));
temp(n + 4) = raw(n) + (raw(n) - raw(n - 1));
temp(n + 3) = raw(n) + (raw(n) - raw(n - 2));

for p = 1 : n
   temp(p+2) = raw(p);
end

for p = 1 : n + 4
   prime(p) = temp(p);
end

for p = 3 : n + 4
   prime(p) = a0 * temp(p) + a1 * temp(p - 1) + a2 * temp(p - 2) + b1 * prime(p-
1) + b2 * prime(p-2);      % Forward Pass
end

for p = 1 : n + 4
   temp(p) = prime(p);
end

for p = n + 2 : -1 : 1
   prime(p) = a0 * temp(p) + a1 * temp(p + 1) + a2 * temp(p + 2) + b1 * prime(p
+ 1) + b2 * prime(p + 2);   % Backward Pass
end
%   ---------------------  Double Pass 5  --------------------------------
```

```
temp(1) = raw(1) + (raw(1) - raw(2));
temp(2) = raw(1) + (raw(1) - raw(3));
temp(n + 4) = raw(n) + (raw(n) - raw(n - 1));
temp(n + 3) = raw(n) + (raw(n) - raw(n - 2));

for p = 1 : n
   temp(p+2) = raw(p);
end

for p = 1 : n + 4
   prime(p) = temp(p);
end

for p = 3 : n + 4
   prime(p) = a0 * temp(p) + a1 * temp(p - 1) + a2 * temp(p - 2) + b1 * prime(p-
1) + b2 * prime(p-2);      % Forward Pass
end

for p = 1 : n + 4
   temp(p) = prime(p);
end

for p = n + 2 : -1 : 1
   prime(p) = a0 * temp(p) + a1 * temp(p + 1) + a2 * temp(p + 2) + b1 * prime(p
+ 1) + b2 * prime(p + 2);   % Backward Pass
end

for p = 1 : n
   smooth(p) = prime(p + 2);   % Return the smoothed signal
end

end
```

Appendix D-5 sIntegrate

```
function [ area ] = sIntegrate( curve, FirstPt, LastPt, dt )
%SINTEGRATE Uses Simpson's rule to integrate discrete points of the vector
%curve[] from FirstPt to LastPt, dt is the time between sample points.
%   curve[] vector to be integrated
%   FirstPt point number in curve[FirstPt] to begin integration.
%   LastPt  point number in curve[LastPt]  to end   integration.
%   dt is the time between sample points in the vector curve[]
area = 0;
nPts = LastPt - FirstPt + 1;
if (nPts > 2)
   if (mod(nPts,2) == 0)          % even number of points to integrate
      area = 3* dt * (curve(FirstPt) + 3 * curve(FirstPt + 1) + 3 * curve(FirstPt +
2) + curve(FirstPt + 3)) / 8;
      for p = FirstPt + 3: 2: LastPt - 2
         area = area + (dt * (curve(p) + 4 * curve(p + 1) + curve(p + 2)) / 3);
      end
   else                           %  odd number of points to integrate
      for p = FirstPt: 2: LastPt - 2
         area = area + (dt * (curve(p) + 4 * curve(p + 1) + curve(p + 2)) / 3);
      end
   end
elseif (nPts == 2)            % If there are only two points use midpoint rule
   area = dt * (curve(FirstPt) + curve(LastPt))/2;
else                          % If there is only one point use Rectangle rule
   area = dt * curve(FirstPt);
end
end
```

Appendix D-6 jbfill

```
function[fillhandle,msg]=jbfill(xpoints,upper,lower,color,edge,add,transparency)
%USAGE:
[fillhandle,msg]=jbfill(xpoints,upper,lower,color,edge,add,transparency)
%This function will fill a region with a color between the two vectors provided
%using the Matlab fill command.
%
%fillhandle is the returned handle to the filled region in the plot.
%xpoints= The horizontal data points (ie frequencies). Note length(Upper)
%        must equal Length(lower)and must equal length(xpoints)!
%upper = the upper curve values (data can be less than lower)
%lower = the lower curve values (data can be more than upper)
%color = the color of the filled area
%edge  = the color around the edge of the filled area
%add   = a flag to add to the current plot or make a new one.
%transparency is a value ranging from 1 for opaque to 0 for invisible for
%the filled color only.
%
%John A. Bockstege November 2006;
%Example:
%    a=rand(1,20);%Vector of random data
%    b=a+2*rand(1,20);%2nd vector of data points;
%    x=1:20;%horizontal vector
%    [ph,msg]=jbfill(x,a,b,rand(1,3),rand(1,3),0,rand(1,1))
%    grid on
%    legend('Datr')
if nargin<7;transparency=.5;end %default is to have a transparency of .5
if nargin<6;add=1;end     %default is to add to current plot
if nargin<5;edge='k';end  %dfault edge color is black
if nargin<4;color='b';end %default color is blue

if length(upper)==length(lower) && length(lower)==length(xpoints)
   msg='';
   filled=[upper,fliplr(lower)];
   xpoints=[xpoints,fliplr(xpoints)];
   if add
      hold on
   end
   fillhandle=fill(xpoints,filled,color);%plot the data
```

```
set(fillhandle,'EdgeColor',edge,'FaceAlpha',transparency,'EdgeAlpha',transparenc
y);%set edge color
    if add
        hold off
    end
else
    msg='Error: Must use the same number of points in each vector';
end
```

Appendix D-7 penddyn

```
function [dydt] = penddyn(t,theta,anktorque,Fx,Fz,La,Ha,L,m)
%Given

g=9.81; %Forces are all in N, Moments in N*mm, distances in mm
I=m*L^2;

%thetadots [theta1dot;theta1dotdot];
dydt = [theta(2);((-
m*g*L*cos(theta(1))/I)+(anktorque(1)/I)+(Fx*Ha/I)+(Fz*La/I))];
```

Appendix D-8 PendAccuracy

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                    ** ABOUT THE CODE **
%
% PendAccuracy
%
% This code calculates and plots the percent error and area differences
% between the model's predicted COM locations and the actual COM locations
% for a group of subjects or conditions.
%
% AreaDifference (Subject Number , Condition Number)
% MaxPercentErrorTheta (Subject Number , Condition Number)
%
% Author: Clifford Hancock
% Date: 2/17/2013
%
% This code utilizes actual subject reflective marker and force plate data.
% Subjects were asked to walk down a walkway with two force plates while
% cameras recorded 3-D locations of reflective markers. The two force plates
% were placed one after the other so that subjects could step from one onto
% the other.
%
% The code utilizes a 2-D pendulum models (Sagittal Plane) ranging from
% the CoM to the ankle rotator point. The code uses the values associated
% with only the single-leg support phase of walking for the 2nd Force Plate.
% Subsequently, it determines how close the model's predicted CoM locations
% are to the actual CoM locations.
%
% The pendulum model utilizes a 4th Order Runge-Kutta Integrator.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%               ** HOW TO SET UP DATA MATRICES **
%
% This code requires data matrices to be in a specific format to run
% properly.
%
% The reflective marker data should be saved as:
%
```

```
% MarkersCondition#Subject# (e.g. MarkersCondition1Subject1)
%
% where the 1 values can change depending upon the condition or subject
% numbers (e.g. MarkersCondition4Subject2 would be the .mat file containing
% the 2nd subject and his 4th condition).
%
% These MarkersCondition#Subject# .mat files should only contain number
% values (no words) and have the proper columns as follows:
%
% (NOTE: X = Walking Direction, Z = Vertical Direction)
%
% Column 1 - Left Ankle Marker X Coordinates
% Column 2 - Left Ankle Marker Y Coordinates
% Column 3 - Left Ankle Marker Z Coordinates
% Column 4 - Right Ankle Marker X Coordinates
% Column 5 - Right Ankle Marker Y Coordinates
% Column 6 - Right Ankle Marker Z Coordinates
% Column 7 - Left Heel Marker X Coordinates
% Column 8 - Left Heel Marker Y Coordinates
% Column 9 - Left Heel Marker Z Coordinates
% Column 10 - Right Heel Marker X Coordinates
% Column 11 - Right Heel Marker Y Coordinates
% Column 12 - Right Heel Marker Z Coordinates
% Column 13 - Left Toe Marker X Coordinates
% Column 14 - Left Toe Marker Y Coordinates
% Column 15 - Left Toe Marker Z Coordinates
% Column 16 - Right Toe Marker X Coordinates
% Column 17 - Right Toe Marker Y Coordinates
% Column 18 - Right Toe Marker Z Coordinates
% Column 19 - Center Of Mass X Coordinates
% Column 20 - Center Of Mass Y Coordinates
% Column 21 - Center Of Mass Z Coordinates
%
% Use the entire data set (all rows) for each column, the code can handle
% blank boxes without numbers (NaN).
%
% *Ensure no rows are missing data (code will not run properly if data is missing
from the single stance phase region)*
% *Code will also fail to run properly if there are "holes" in the data*
%
% Units: All coordinates = mm
```

```
%
% The force plate data should be saved as:
%
% ForcePlateCondition#Subject# (e.g. ForcePlateCondition1Subject1).
%
% The values of 1 can be changed to match the marker's values.
%
% These ForcePlateCondition#Subject# .mat files should only contain number
% values (no words) and have the proper columns as follows:
%
% (NOTE: X = Walking Direction, Z = Vertical Direction)
%
% Column 1 - Center of Pressure X Coordinates for Force Plate 1
% Column 2 - Center of Pressure X Coordinates for Force Plate 2
% Column 3 - Center of Pressure Z Coordinates for Force Plate 2
% Column 4 - X-Direction Force for Force Plate 2
% Column 5 - Z-Direction Force for Force Plate 2
% Column 6 - Y-Direction Moment for Force Plate 2
%
% Use the entire data set (all rows) for each column, the code can handle
% blank boxes without numbers.
%
% *Ensure no rows are missing data (code will not run properly if data is missing
from the single stance phase region)*
% *Code will also fail to run properly if there are "holes" in the data*
%
% Units: CoP = mm, Force = N, Moment = N*mm
%
% Lastly, establish a mass matrix for the subjects named:
%
% SubjectMass (e.g. SubjectMass=[70 70 70]'; OR SubjectMass=ones(3,1)*70;)
%
% where it is a column vector with each row corresponding to the mass of
% the subject.
%
% Units: mass = kg
%
% In addition to setting up the data matrices, alter "USER INPUTS" below to
% utilize the code properly.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% USER INPUTS

%                    ** NOTE **
%If you select different values for First and Last Subject, you must have
%.mat files for all the subject number values in between

FirstSubject = 20; %Put as 1 to start with ForcePlateCondition#Subject1, put as
30 to start with ForcePlateCondition#Subject30
LastSubject = 20; %Put as 3 to end with ForcePlateCondition#Subject3, put as 34
to end with ForcePlateCondition#Subject34
%Number of Conditions per subject (e.g. Eyes Open and Eyes Closed = 2
Conditions)
FirstCondition = 1; %Put as 1 to start with ForcePlateCondition1Subject#, put as
2 to start with ForcePlateCondition2Subject#
LastCondition = 1; %Put as 1 to end with ForcePlateCondition1Subject#, put as 3
to end with ForcePlateCondition3Subject#

  %                    ** NOTE **
  %Code will result in an ERROR if Forceplatefrequency/Camerafrequency
  %is not an integer, e.g. 1080 Hz / 120 Hz = 9 is acceptable,
  %1080 Hz / 110 Hz = 9.82 is not acceptable.

Forceplatefrequency=1080; %units of Hz (Must be higher than Camerafrequency
for code to run)
Camerafrequency=120;    %units of Hz

fs(1,1)=1080;     %Sampling Frequency for Force Plate Data (units of Hz)
fs(2,1)=120;    %Sampling Frequency for Marker Data (units of Hz)
fc(1,1)=30;   %Cutoff Frequency for Force Plate Data (units of Hz)
fc(2,1)=6;   %Cutoff Frequency for Marker Data (units of Hz)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load SubjectMass   %Loads a mass matrix Nx1 with each row corresponding to
the mass of each subject
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Error Messages
if rem(FirstSubject,1)~=0 || rem(LastSubject,1)~=0 || rem(FirstCondition,1)~=0 ||
rem(FirstCondition,1)~=0
   fprintf('ALL First and Last values must be integers! \n')
   break
end
if LastSubject<FirstSubject
   fprintf('FirstSubject value must be less than or equal to the LastSubject value!
\n')
   break
end
if LastCondition<FirstCondition
   fprintf('FirstCondition value must be less than or equal to the LastCondition
value! \n')
   break
end
if FirstSubject<1 || FirstCondition<1 || LastSubject<1 || LastCondition<1
   fprintf('ALL First and Last values must be integers 1 or higher! \n')
   break
end
if size(SubjectMass,1)<FirstSubject || size(SubjectMass,1)<LastSubject
   %Question the User
   fprintf('The SubjectMass matrix must be a column vector with the number of
rows equal to the number of subjects (one mass value (kg) for each subject)! \n\n')
   fprintf('Would you like to use a temporary SubjectMass Matrix where each
subject is 70kg?'); fprintf('\n\n')
   fprintf('1 = Yes'); fprintf('\n')
   fprintf('2 = No'); fprintf('\n\n')
   fprintf('(Insert value and press enter)'); fprintf('\n')
   fileselector2 = input('Choice #'); fprintf('\n')
   if isempty(fileselector2)
      fileselector2 = 1;   %Simply pressing "enter" with no user input will say
"yes"
      fprintf('Question ACCEPTED due to no user input!')
      fprintf('\n\n\n')
   end
```

```
    fileselector2=round(fileselector2);    %rounds any decimal values to nearest
integer
    fprintf('Choice #')
    fprintf('%0.0f',fileselector2)
    fprintf(' was selected...')
    fprintf('\n\n')

    if fileselector2==1
        SubjectMass=ones(LastSubject,1)*70; %temporary SubjectMass matrix with
each subject at 70kg
    else
        fprintf('Please adjust the SubjectMass matrix to proceed! \n')
        break
    end

    %User input Error Messages
    if fileselector2>=3
        fprintf('This is not a valid option!')
        fprintf('\n\n\n')
        break
    elseif fileselector2<=0
            fprintf('This is not a valid option!')
            fprintf('\n\n\n')
            break
    end
end
if Forceplatefrequency<Camerafrequency
    fprintf('Forceplatefrequency must be higher than Camerafrequency for code to
run! \n')
    break
end
if rem(Forceplatefrequency/Camerafrequency,1)~=0
    fprintf('Forceplatefrequency/Camerafrequency value MUST be an integer for
code to run! \n')
    break
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Question the User
```

```
fprintf('Would you like to view the animation of the CoM Pendulum?');
fprintf('\n\n')
fprintf('1 = Yes'); fprintf('\n')
fprintf('2 = No'); fprintf('\n\n')
fprintf('(Insert value and press enter)'); fprintf('\n')
fileselector = input('Choice #'); fprintf('\n')
if isempty(fileselector)
   fileselector = 1;   %Simply pressing "enter" with no user input will say "yes"
   fprintf('Animation will be viewed due to no user input!')
   fprintf('\n\n\n')
end

fileselector=round(fileselector);    %rounds any decimal values to nearest integer
fprintf('Choice #')
fprintf('%0.0f',fileselector)
fprintf(' was selected...')
fprintf('\n\n')

%User input Error Messages
if fileselector>=3
   fprintf('This is not a valid option!')
   fprintf('\n\n\n')
elseif fileselector<=0
   fprintf('This is not a valid option!')
   fprintf('\n\n\n')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NumberOfConditions = size(FirstCondition:LastCondition,2);
NumberOfSubjects = size(FirstSubject:LastSubject,2);

% Load Data and Filter

AreaDifference=zeros(NumberOfSubjects,3);
MaxPercentErrorTheta=zeros(NumberOfSubjects,3);
SubjectPlotCounter=0;
for SubjectCounter=FirstSubject:1:LastSubject
   SubjectPlotCounter=SubjectPlotCounter+1;
for ConditionCounter=FirstCondition:1:LastCondition
```

```
MarkersNumber = ['MarkersCondition' int2str(ConditionCounter) 'Subject'
int2str(SubjectCounter)];
Markers = load(MarkersNumber);
ForcePlateNumber = ['ForcePlateCondition' int2str(ConditionCounter) 'Subject'
int2str(SubjectCounter)];
Forceplatedatafull = load(ForcePlateNumber);

Markers = struct2cell(Markers);
Markers = cell2mat(Markers);

Forceplatedatafull = struct2cell(Forceplatedatafull);
Forceplatedatafull = cell2mat(Forceplatedatafull);

%%% Apply a Butterworth Filter for Forceplatedata

Forceplatedataupper=zeros(1,size(Forceplatedatafull,2));
Forceplatedatalower=zeros(1,size(Forceplatedatafull,2));
filtered_forceplatedata=zeros(size(Forceplatedatafull,1),size(Forceplatedatafull,2)
);
for j=1:1:size(Forceplatedatafull,2)    %does all columns of Forceplatedatafull
   Forceplatedataupper(j)=0;

   for i=1:1:size(Forceplatedatafull(:,j))
      if Forceplatedataupper(j)<1     %if upper bound is being chosen, lower bound
must have been chosen already
         Forceplatedatalower(j)=i;    %Locates the lower range of the data to filter
      end
      if isnan(Forceplatedatafull(i,j))==0 %if isnan=0, the value for
Forceplatedatafull is NOT NaN, hence, Forceplatedatafull IS a real number
         Forceplatedataupper(j)=i;  %Locates the upper range of the data to filter

      end
      if Forceplatedataupper(j)~=0 && isnan(Forceplatedatafull(i,j))==1 &&
isnan(Forceplatedatafull(i+1,j))==1
         break %stops the filtering as soon as NaN appears
      end
      %if a single row's value is missing, replace it with an average of
      %the previous and following row
      if Forceplatedataupper(j)~=0 && isnan(Forceplatedatafull(i,j))==1 &&
isnan(Forceplatedatafull(i+1,j))==0
```

```
        Forceplatedatafull(i,j)=(Forceplatedatafull(i-
1,j)+Forceplatedatafull(i+1,j))/2;
      end
   end

   filtered_forceplatedata(1:(Forceplatedatalower(j)-1),j)=NaN;

filtered_forceplatedata(Forceplatedatalower(j):Forceplatedataupper(j),j)=Criticall
yDampedButterworthLowPass(Forceplatedatafull(Forceplatedatalower(j):Forcepl
atedataupper(j),j),fs(1,1),fc(1,1));   %Calls the Butterpass function

filtered_forceplatedata((Forceplatedataupper(j)+1):size(Forceplatedatafull,1),j)=N
aN;
end

%%% Apply a Butterworth Filter for Marker data

Markersdataupper=zeros(1,size(Markers,2));
Markersdatalower=zeros(1,size(Markers,2));
filtered_markerdata=zeros(size(Markers,1),size(Markers,2));
for jj=1:1:size(Markers,2)    %does all columns of Markers
   Markersdataupper(jj)=0;

   for ii=1:1:size(Markers(:,jj))
      if Markersdataupper(jj)<1     %if upper bound is being chosen, lower bound
must have been chosen already
      Markersdatalower(jj)=ii;    %Locates the lower range of the data to filter
      end
      if isnan(Markers(ii,jj))==0 %if isnan=0, the value for Markers is NOT NaN,
hence, Markers IS a real number
         Markersdataupper(jj)=ii;  %Locates the upper range of the data to filter
      end
   end

   filtered_markerdata(1:(Markersdatalower(jj)-1),jj)=NaN;

filtered_markerdata(Markersdatalower(jj):Markersdataupper(jj),jj)=ButterworthL
owPassFilter3Dcoords(Markers(Markersdatalower(jj):Markersdataupper(jj),jj),fs(
2,1),fc(2,1));   %Calls the Butterpass function
   filtered_markerdata((Markersdataupper(jj)+1):size(Markers,1),jj)=NaN;
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Find Upper and Lower bound rows of where the feet make contact with the
%%% force plates


Jumpsize=Forceplatefrequency/Camerafrequency;   %Helps ensure both
frequencies are the same
ForcePlateAdj=zeros(size(filtered_markerdata,1),size(Forceplatedatafull,2));
ForcePlateAdjUpper=zeros(1,size(Forceplatedatafull,2));
ForcePlateAdjLower=zeros(1,size(Forceplatedatafull,2));
j=1;
for i=1:Jumpsize:size(filtered_forceplatedata,1)

   ForcePlateAdj(j,:)=filtered_forceplatedata(i,:);   %Take every jumpsize row
and make a new forceplate matrix
   j=j+1;

end

for j=1:1:size(ForcePlateAdj,2)    %does all columns of ForceplateAdj
   ForcePlateAdjUpper(j)=0;

   for i=1:1:size(ForcePlateAdj(:,j))
      if ForcePlateAdjUpper(j)<1     %if upper bound is being chosen, lower
bound must have been chosen already
         ForcePlateAdjLower(j)=i;    %Locates the row associated with the lower
range of the data
      end
      if isnan(ForcePlateAdj(i,j))==0 %if isnan=0, the value for Forceplatedatafull
is NOT NaN, hence, Forceplatedatafull IS a real number
         ForcePlateAdjUpper(j)=i;  %Locates the row where the toe lifts from 1st
force plate
      end
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%% Reset ForcePlateAdj to the desired range for SINGLE-LEG SUPPORT

%Decide which foot is lifting in beginning by checking vertical velocity
   i=ForcePlateAdjUpper(1);
   HeelVertVel=zeros(1,i);
   HeelVertAccel=zeros(1,i);
   h=1/Camerafrequency;
   VertVelTestLeft=(filtered_markerdata((ForcePlateAdjUpper(1)+1),9)-
filtered_markerdata((ForcePlateAdjUpper(1)-1),9))/(2*h);
   VertVelTestRight=(filtered_markerdata((ForcePlateAdjUpper(1)+1),12)-
filtered_markerdata((ForcePlateAdjUpper(1)-1),12))/(2*h);
   %If left foot is lifting
   if VertVelTestLeft>VertVelTestRight
      %Find when heel strike occurs
      while (1)
         HeelVertVel(i)=(filtered_markerdata((i+1),9)-filtered_markerdata((i-
1),9))/(2*h);
         HeelVertAccel(i)=(filtered_markerdata((i+1),9)-
2*filtered_markerdata(i,9)+filtered_markerdata((i-1),9))/(h^2);
%           if abs(HeelVertAccel(i))<=200 && abs(HeelVertVel(i))<=200   %200
chosen arbitrarily as cutoff
         if abs(HeelVertVel(i))<200 && HeelVertAccel(i)>0 &&
filtered_markerdata(i,9)<100
            HeelStrike=i;
            break
         end
         i=i+1;
      end
   end

   %If right foot is lifting
   if VertVelTestRight>VertVelTestLeft
      %Find when heel strike occurs
      while (1)
         HeelVertVel(i)=(filtered_markerdata((i+1),12)-filtered_markerdata((i-
1),12))/(2*h);
         HeelVertAccel(i)=(filtered_markerdata((i+1),12)-
2*filtered_markerdata(i,12)+filtered_markerdata((i-1),12))/(h^2);
%           if abs(HeelVertAccel(i))<=200 && abs(HeelVertVel(i))<=200   %200
chosen arbitrarily as cutoff
```

```
        if abs(HeelVertVel(i))<200 && HeelVertAccel(i)>0 &&
filtered_markerdata(i,12)<100
            HeelStrike=i-1;
            break
        end
        i=i+1;
    end
  end

ForcePlateAdj=ForcePlateAdj(ForcePlateAdjUpper(1):HeelStrike,:);
%ForcePlateAdjUpper(1)=toe off of force plate 1

%%% Convert filtered_data into individual columns

   % Force Plate 1 (:,1)

   copx(:,1) = ForcePlateAdj(:,1);    %  make 1st column of ForcePlateAdj "copx"

   % Force Plate 2 (:,2)

   copx(:,2) = ForcePlateAdj(:,2);    %  make 2nd column of ForcePlateAdj
"copx"
   copz(:,2) = ForcePlateAdj(:,3);
   fx(:,2) = ForcePlateAdj(:,4);
   fz(:,2) = ForcePlateAdj(:,5);
   my(:,2) = ForcePlateAdj(:,6);

Limited_MarkerData=filtered_markerdata(ForcePlateAdjUpper(1):HeelStrike,:);
%ForcePlateAdjUpper(1)=toe off of force plate 1

%%% Convert Markers

columncount=size(Limited_MarkerData,2);    %Finds number of columns in
filtered_markerdata
columncountadj=1:3:columncount;
Markersadj=zeros(size(Limited_MarkerData,1),length(columncountadj),3);
j=1;
for i=1:3:columncount    %i only selects x components from Markers (want every
3rd column (XyzXyzXyz)
   Markersadj(:,j,1)=Limited_MarkerData(:,i); %j makes each column of
Markersadj a different marker (hip/knee/ankle...)
```

```
    j=j+1;
end

j=1;
for i=2:3:columncount    %i only selects y components from Markers (want every
3rd column (xYzxYzxYz)
    Markersadj(:,j,2)=Limited_MarkerData(:,i);
    j=j+1;
end

j=1;
for i=3:3:columncount    %i only selects z components from Markers (want every
3rd column (xyZxyZxyZ)
    Markersadj(:,j,3)=Limited_MarkerData(:,i);
    j=j+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%Calculate angle (theta) for CoM pendulum in DEGREES
if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
    ThetaCOM=zeros(size(Markersadj(:,7,1),1),1);
    for i=1:1:(size(Markersadj(:,7,1)))
        ThetaCOM(i,1)=(atan2((Markersadj(i,7,3)-
Markersadj(i,1,3)),(Markersadj(i,7,1)-Markersadj(i,1,1))));
    end
end

if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
    ThetaCOM=zeros(size(Markersadj(:,7,1),1),1);
    for i=1:1:(size(Markersadj(:,7,1)))
        ThetaCOM(i,1)=(atan2((Markersadj(i,7,3)-
Markersadj(i,2,3)),(Markersadj(i,7,1)-Markersadj(i,2,1)))); %7 for CoM, 2 for
RANK
    end
end

%Calculate Length from Ankle to CoM at each time interval
N=length(Markersadj(:,:,1));    %Makes for loop (below) stop when no more
Marker values
```

```
L=zeros(1,i);

if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
   for i=1:1:N
      L(i)=sqrt(((Markersadj(i,7,3)-Markersadj(i,1,3))^2+(Markersadj(i,7,1)-
Markersadj(i,1,1))^2)); %Changes length throughout based upon real subject
changes
   end
end

if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
   for i=1:1:N
      L(i)=sqrt(((Markersadj(i,7,3)-Markersadj(i,2,3))^2+(Markersadj(i,7,1)-
Markersadj(i,2,1))^2)); %Changes length throughout based upon real subject
changes
   end
end

anktorque=zeros(N,1);
Fx=zeros(N,1);
Fz=zeros(N,1);
La=zeros(N,1);
Ha=zeros(N,1);
theta=zeros(N+1,2);
x=zeros(N+1,2);
z=zeros(N+1,2);

%Set up Time Variables
dt=h;   %h=1/120. 120 = camera frequency

time=[0 20];    %5 is completely arbitrary and not really used (prevents infinite
looping)
timebeg= time(1);
timeend= time(2);
t= (timebeg:dt:timeend)';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %Initial x and z locations for pendulum
   theta0(1,1)=ThetaCOM(1,1);
   if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
      x(1,1)= Markersadj(1,1,1);  %1 is for LANK
```

```
        z(1,1)= Markersadj(1,1,3);
    end
    if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
        x(1,1)= Markersadj(1,2,1);  %2 is for RANK
        z(1,1)= Markersadj(1,2,3);
    end
    x(1,2)= x(1,1)+(L(1)*cos(theta0(1,1)));
    z(1,2)= z(1,1)+(L(1)*sin(theta0(1,1)));

    %Calculate inital angular velocity
    theta(1,1)= theta0; %initial angle
    h=1/Camerafrequency;    %120 = camera frequency
    theta(1,2)=(ThetaCOM(3,1)-ThetaCOM(1,1))/(2*h); %initial angular velocity
of CoM Pendulum from ankle to CoM

for i = 1:1:N

        anktorque(i,1)=my(i,2)'; %units of N*mm
        Fx(i,1)=fx(i,2)';   %forces applied at COP location
        Fz(i,1)=fz(i,2)';

        if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
            Ha(i,1)=(Markersadj(i,1,3)-copz(i,2))';    %ankle z position - copz position
            La(i,1)=(copx(i,2)-Markersadj(i,1,1))';  %ankle x position - copz position
        end

        if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
            Ha(i,1)=(Markersadj(i,2,3)-copz(i,2))';    %ankle z position - copz position
            La(i,1)=(copx(i,2)-Markersadj(i,2,1))';  %ankle x position - copz position
        end

        K1=
penddyn(t,theta(i,:),anktorque(i),Fx(i),Fz(i),La(i),Ha(i),L(i),SubjectMass(Subject
Counter,1))';
            thetamid= K1*(dt/2)+theta(i,:);
        K2=
penddyn(t+dt/2,thetamid,anktorque(i),Fx(i),Fz(i),La(i),Ha(i),L(i),SubjectMass(Su
bjectCounter,1))';
            thetamid= K2*(dt/2)+theta(i,:);
```

```
        K3=
penddyn(t+dt/2,thetamid,anktorque(i),Fx(i),Fz(i),La(i),Ha(i),L(i),SubjectMass(Su
bjectCounter,1))';
          thetaend= K3*(dt)+theta(i,:);
        K4=
penddyn(t+dt,thetaend,anktorque(i),Fx(i),Fz(i),La(i),Ha(i),L(i),SubjectMass(Subj
ectCounter,1))';
          theta(i+1,:)= theta(i,:)+(dt/6)*(K1+2*K2+2*K3+K4);

    %Subsequent x and z locations of ankle (1) and CoM (2) for animate.m
    if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
        x(i+1,1)=Markersadj(i,1,1);
        z(i+1,1)=Markersadj(i,1,3);
    end
    if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
        x(i+1,1)=Markersadj(i,2,1);
        z(i+1,1)=Markersadj(i,2,3);
    end
    x(i+1,2)=x(i+1,1)+(L(i)*cos(theta(i+1,1)));
    z(i+1,2)=z(i+1,1)+(L(i)*sin(theta(i+1,1)));

        if timebeg>=timeend,break,end
        if timebeg>(timeend-dt),break,end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Plot CoM Pendulum Movement (side view) for RK4

COMpendx(:,1,1)=x(:,2); %2=CoM positions
COMpendx(:,2,1)=x(:,1); %1=Ankle positions
COMpendz(:,1,1)=z(:,2);
COMpendz(:,2,1)=z(:,1);

if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
    FOOTx=zeros(size(Markersadj(:,3,1),1),4);
    FOOTz=zeros(size(Markersadj(:,3,1),1),4);

    for i=1:1:N
        FOOTx(i,1,1)=Markersadj(i,3,1); %heel
```

```
        FOOTx(i,2,1)=Markersadj(i,5,1); %toe
        FOOTx(i,3,1)=Markersadj(i,1,1); %ankle
        FOOTx(i,4,1)=Markersadj(i,3,1); %heel
        FOOTz(i,1,1)=Markersadj(i,3,3);
        FOOTz(i,2,1)=Markersadj(i,5,3);
        FOOTz(i,3,1)=Markersadj(i,1,3);
        FOOTz(i,4,1)=Markersadj(i,3,3);
    end
end

if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
    FOOTx=zeros(size(Markersadj(:,4,1),1),4);
    FOOTz=zeros(size(Markersadj(:,4,1),1),4);

    for i=1:1:N
        FOOTx(i,1,1)=Markersadj(i,4,1); %heel
        FOOTx(i,2,1)=Markersadj(i,6,1); %toe
        FOOTx(i,3,1)=Markersadj(i,2,1); %ankle
        FOOTx(i,4,1)=Markersadj(i,4,1); %heel
        FOOTz(i,1,1)=Markersadj(i,4,3);
        FOOTz(i,2,1)=Markersadj(i,6,3);
        FOOTz(i,3,1)=Markersadj(i,2,3);
        FOOTz(i,4,1)=Markersadj(i,4,3);
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Setup figure to view CoM Pendulum for RK4

Xmax1=max(x(:,2))+10;    %Find x vector containing largest value, then finds
max value
Xmax2=max(FOOTx(:,2,1));
maxx(1)=max(Xmax1,Xmax2);
minx(1)=min(x(:,2))-10;    %Find x vector with smallest value, then finds min
value
maxz(1)=max(z(:,2))+10;    %Find z vector containing largest value, then finds
max value
minz(1)=min(copz(:,1))-10;    %Find z vector with smallest value, then finds min
value
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plot the actual CoM locations and predicted CoM locations
 figure(14);
 plot(x(1:(length(x)-1),2),z(1:(length(z)-1),2))    %predicted CoM Positions
 hold on; plot(Markersadj(:,7,1),Markersadj(:,7,3),'r') %actual CoM Positions
 xlabel('Position x (mm)')
 ylabel('Position z (mm)')
 title('Comparison of Predicted and Actual COM Positions','FontSize',12)
 COMLegend = legend('Model COM Prediction' , 'Actual COM Locations');
 set(COMLegend,'Location','NorthEast')
 pause;
 hold off;

% Calculate the Difference between the actual CoM locations and predicted
% CoM locations

% Find the Error Associated with the Predicted CoM Locations
DifferenceX=x(1:(length(x)-1),2)-Markersadj(:,7,1);
PercentErrorX=(abs(x(1:(length(x)-1),2)-
Markersadj(:,7,1))./abs(Markersadj(:,7,1)))*100;
DifferenceZ=z(1:(length(z)-1),2)-Markersadj(:,7,3);
PercentErrorZ=(abs(z(1:(length(z)-1),2)-
Markersadj(:,7,3))./abs(Markersadj(:,7,3)))*100;
PercentErrorTheta=(abs(theta(1:N,1)-
ThetaCOM(:,1))./abs(ThetaCOM(:,1)))*100;
MaxPercentErrorTheta(SubjectPlotCounter,ConditionCounter)=max(PercentError
Theta);
MeanPEx=mean(PercentErrorX);
MeanPEz=mean(PercentErrorZ);

toparea=sIntegrate(Markersadj(:,7,3), 1,size(Markersadj(:,7,3),1),1);
bottomarea=sIntegrate(z(1:(length(z)-1),2), 1,size(Markersadj(:,7,3),1),1);
AreaDifference(SubjectPlotCounter,ConditionCounter)=toparea-bottomarea;
%negative means predicted is typically higher, positive means actual is typically
higher

plot(PercentErrorX)
hold on; plot(PercentErrorZ,'r')
xlabel('Data Point Number')
ylabel('Percent Errors (%)')
```

```
title('X- and Z-Direction Percent Errors','FontSize',12)
PercentLegend = legend('PercentErrorX' , 'PercentErrorZ');
set(PercentLegend,'Location','NorthWest')
pause;
hold off;
plot(PercentErrorTheta)
xlabel('Data Point Number')
ylabel('Percent Errors (%)')
title('Theta Percent Errors','FontSize',12)
hold off;
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if fileselector==1
   fprintf('Press "enter" to view animation...')
figure(17);
for i=1:1:size(Markersadj,1)
   UPPER=[FOOTz(i,1),FOOTz(i,3),FOOTz(i,2)];
   SLOPE=(FOOTz(i,1)-FOOTz(i,2))/(FOOTx(i,1)-FOOTx(i,2));
   INTERCEPT=FOOTz(i,1)-(SLOPE*FOOTx(i,1));
   YPOINT=SLOPE*FOOTx(i,3)+INTERCEPT;  %y=mx+b
   LOWER=[FOOTz(i,1),YPOINT,FOOTz(i,2)];
   XLOC=[FOOTx(i,1),FOOTx(i,3),FOOTx(i,2)];
   plot(x(i,2),z(i,2),'ro');
   if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
      hold on; plot(Markersadj(i,3,1),Markersadj(i,3,3),'ko');
      hold on; plot(Markersadj(i,5,1),Markersadj(i,5,3),'ko');
      hold on; plot(Markersadj(i,1,1),Markersadj(i,1,3),'ko');
   end
   if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
      hold on; plot(Markersadj(i,4,1),Markersadj(i,4,3),'ko');
      hold on; plot(Markersadj(i,6,1),Markersadj(i,6,3),'ko');
      hold on; plot(Markersadj(i,2,1),Markersadj(i,2,3),'ko');
   end
   hold on; plot(COMpendx(i,:),COMpendz(i,:),'k-');
   hold on; plot(copx(i,2),copz(i,2),'go');
   hold on; plot(FOOTx(i,:),FOOTz(i,:),'k-')
   hold on; jbfill(XLOC,UPPER,LOWER,'c');
   axis equal;
   axis([minx(1) maxx(1) minz(1) maxz(1)]);
```

```
    xlabel('Position x (mm)')
    ylabel('Position z (mm)')
    title('CoM Pendulum during Single-Leg Support Phase (2nd Force
Plate)','FontSize',12)
    text(minx(1)+25,.1,num2str(i));
   hold on; plot(COMpendx(1:i-1,1),COMpendz(1:i-1,1),'ko');    %plot all previous
CoM positions
   hold on; plot(copx(1:i-1,2),copz(1:i-1,2),'go');
   drawnow;
     if i==1
    pause;
     end
    hold off;
end
    %Plot Static Final Image
    plot(x(i,2),z(i,2),'ro');
    if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
        hold on; plot(Markersadj(i,3,1),Markersadj(i,3,3),'ko');
        hold on; plot(Markersadj(i,5,1),Markersadj(i,5,3),'ko');
        hold on; plot(Markersadj(i,1,1),Markersadj(i,1,3),'ko');
    end
    if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
        hold on; plot(Markersadj(i,4,1),Markersadj(i,4,3),'ko');
        hold on; plot(Markersadj(i,6,1),Markersadj(i,6,3),'ko');
        hold on; plot(Markersadj(i,2,1),Markersadj(i,2,3),'ko');
    end
    hold on; plot(COMpendx(i,:),COMpendz(i,:),'k-');
    hold on; plot(COMpendx(1:i-1,1),COMpendz(1:i-1,1),'ko');    %plot all
previous CoM positions
    hold on; plot(copx(:,2),copz(:,2),'go');
    hold on; plot(FOOTx(i,:),FOOTz(i,:),'k-')
    hold on; jbfill(XLOC,UPPER,LOWER,'c');
    axis equal;
    axis([minx(1) maxx(1) minz(1) maxz(1)]);
    xlabel('Position x (mm)')
    ylabel('Position z (mm)')
    title('CoM Pendulum during Single-Leg Support Phase (2nd Force
Plate)','FontSize',12)
    text(minx(1)+25,.1,num2str(i));
    pause;
    hold off;
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clearvars -except fileselector NumberOfConditions NumberOfSubjects
Forceplatefrequency Camerafrequency...
    SubjectCounter SubjectMass fs fc MaxPercentErrorTheta AreaDifference
FirstSubject FirstCondition ...
    LastSubject LastCondition SubjectPlotCounter

end
end

% Turn any zero value into NaN (Not a Number)
for iii=1:1:NumberOfSubjects
    for jjj=1:1:3   %goes through 3 conditions
        if AreaDifference(iii,jjj)==0
            AreaDifference(iii,jjj)=NaN;
        end
        if MaxPercentErrorTheta(iii,jjj)==0
            MaxPercentErrorTheta(iii,jjj)=NaN;
        end
    end
end

%Print the Results
fprintf('\n o--------------------------------------------------o \n')
fprintf('  Each Row is a Subject. Each Column is a Condition. \n\n')
fprintf('  Area Difference (mm^2) = \n')
for i=1:size(AreaDifference,1)
fprintf('  ')
fprintf( '%f\t',AreaDifference(i,:)) %f for fixed point notation
fprintf('\n\n')
end
fprintf(' o--------------------------------------------------o \n')

fprintf('\n o--------------------------------------------------o \n')
fprintf('  Each Row is a Subject. Each Column is a Condition. \n\n')
fprintf('  Maximum Percent Error Theta = \n')
for i=1:size(MaxPercentErrorTheta,1)
fprintf('  ')
```

```
fprintf( '%f\t',MaxPercentErrorTheta(i,:)) %f for fixed point notation
fprintf('\n\n')
end
fprintf(' o----------------------------------------------o \n')

clearvars -except MaxPercentErrorTheta AreaDifference
close all
```

Appendix D-9 AreaofStability

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                    ** ABOUT THE CODE **
%
% AreaOfStability
%
% This code calculates and plots the area of stability for a group of
% subjects or conditions.
%
% CalcAreaOfStability (Subject Number , Condition Number)
%
% Author: Clifford Hancock
% Date: 2/16/2013
%
% This code utilizes actual subject reflective marker and force plate data.
% Subjects were asked to walk down a walkway with two force plates while
% cameras recorded 3-D locations of reflective markers. The two force plates
% were placed one after the other so that subjects could step from one onto
% the other.
%
% The code utilizes many 2-D pendulum models (Sagittal Plane) ranging from
% the CoM to the ankle rotator point. The code uses the values associated
% with only the single-leg support phase of walking for the 2nd Force Plate
% for a wide range of inputs (theta and thetadot). Subsequently, it determines
% the limits of stability during this walking phase based upon the XcoM
% (Hof et al) values for each input pair. The area of stability is calculated
% as the area between the upper and lower limits of stability.
%
% The pendulum model utilizes a 4th Order Runge-Kutta Integrator.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%               ** HOW TO SET UP DATA MATRICES **
%
% This code requires data matrices to be in a specific format to run
% properly.
%
% The reflective marker data should be saved as:
%
```

```
% MarkersCondition#Subject# (e.g. MarkersCondition1Subject1)
%
% where the 1 values can change depending upon the condition or subject
% numbers (e.g. MarkersCondition4Subject2 would be the .mat file containing
% the 2nd subject and his 4th condition).
%
% These MarkersCondition#Subject# .mat files should only contain number
% values (no words) and have the proper columns as follows:
%
% (NOTE: X = Walking Direction, Z = Vertical Direction)
%
% Column 1 - Left Ankle Marker X Coordinates
% Column 2 - Left Ankle Marker Y Coordinates
% Column 3 - Left Ankle Marker Z Coordinates
% Column 4 - Right Ankle Marker X Coordinates
% Column 5 - Right Ankle Marker Y Coordinates
% Column 6 - Right Ankle Marker Z Coordinates
% Column 7 - Left Heel Marker X Coordinates
% Column 8 - Left Heel Marker Y Coordinates
% Column 9 - Left Heel Marker Z Coordinates
% Column 10 - Right Heel Marker X Coordinates
% Column 11 - Right Heel Marker Y Coordinates
% Column 12 - Right Heel Marker Z Coordinates
% Column 13 - Left Toe Marker X Coordinates
% Column 14 - Left Toe Marker Y Coordinates
% Column 15 - Left Toe Marker Z Coordinates
% Column 16 - Right Toe Marker X Coordinates
% Column 17 - Right Toe Marker Y Coordinates
% Column 18 - Right Toe Marker Z Coordinates
% Column 19 - Center Of Mass X Coordinates
% Column 20 - Center Of Mass Y Coordinates
% Column 21 - Center Of Mass Z Coordinates
%
% Use the entire data set (all rows) for each column, the code can handle
% blank boxes without numbers (NaN).
%
% *Ensure no rows are missing data (code will not run properly if data is missing
from the single stance phase region)*
% *Code will also fail to run properly if there are "holes" in the data*
%
% Units: All coordinates = mm
```

```
%
% The force plate data should be saved as:
%
% ForcePlateCondition#Subject# (e.g. ForcePlateCondition1Subject1).
%
% The values of 1 can be changed to match the marker's values.
%
% These ForcePlateCondition#Subject# .mat files should only contain number
% values (no words) and have the proper columns as follows:
%
% (NOTE: X = Walking Direction, Z = Vertical Direction)
%
% Column 1 - Center of Pressure X Coordinates for Force Plate 1
% Column 2 - Center of Pressure X Coordinates for Force Plate 2
% Column 3 - Center of Pressure Z Coordinates for Force Plate 2
% Column 4 - X-Direction Force for Force Plate 2
% Column 5 - Z-Direction Force for Force Plate 2
% Column 6 - Y-Direction Moment for Force Plate 2
%
% Use the entire data set (all rows) for each column, the code can handle
% blank boxes without numbers.
%
% *Ensure no rows are missing data (code will not run properly if data is missing
from the single stance phase region)*
% *Code will also fail to run properly if there are "holes" in the data*
%
% Units: CoP = mm, Force = N, Moment = N*mm
%
% Lastly, establish a mass matrix for the subjects named:
%
% SubjectMass (e.g. SubjectMass=[70 70 70]';)
%
% where it is a column vector with each row corresponding to the mass of
% the subject.
%
% Units: mass = kg
%
% In addition to setting up the data matrices, alter "USER INPUTS" below to
% utilize the code properly.
%
%                         ** NOTE **
```

```
% It is recommended that either the NumberOfConditions or NumberOfSubjects
% variables =1 (OR BOTH =1) in order to better analyze the resulting plot.
% Each subsequent condition's or subject's limits get plotted on the same
% plot. A large number of conditions or subjects would result in a very long
% computation time and a very clustered plot. The plotting code could be
% altered to produce separate figures for each loop if desired.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic % tic (toc at end) determines time elapsed
clear
% close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% USER INPUTS

%                       ** NOTE **
%If you select different values for First and Last Subject, you must have
%.mat files for all the subject number values in between

FirstSubject = 45; %Put as 1 to start with ForcePlateCondition#Subject1, put as
30 to start with ForcePlateCondition#Subject30
LastSubject = 46; %Put as 3 to end with ForcePlateCondition#Subject3, put as 34
to end with ForcePlateCondition#Subject34
%Number of Conditions per subject (e.g. Eyes Open and Eyes Closed = 2
Conditions)
FirstCondition = 3; %Put as 1 to start with ForcePlateCondition1Subject#, put as
2 to start with ForcePlateCondition2Subject#
LastCondition = 3; %Put as 1 to end with ForcePlateCondition1Subject#, put as 3
to end with ForcePlateCondition3Subject#

   %                       ** NOTE **
   %Code will result in an ERROR if Forceplatefrequency/Camerafrequency
   %is not an integer, e.g. 1080 Hz / 120 Hz = 9 is acceptable,
   %1080 Hz / 110 Hz = 9.82 is not acceptable.

Forceplatefrequency=1080; %units of Hz (Must be higher than Camerafrequency
for code to run)
Camerafrequency=120;    %units of Hz
```

```
fs(1,1)=1080;      %Sampling Frequency for Force Plate Data (units of Hz)
fs(2,1)=120;     %Sampling Frequency for Marker Data (units of Hz)
fc(1,1)=30;    %Cutoff Frequency for Force Plate Data (units of Hz)
fc(2,1)=6;    %Cutoff Frequency for Marker Data (units of Hz)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Define Area to Search for Limits
%
% *Values below typically do not require altering UNLESS limits appear
% cutoff in final outputted figure*

% Horizontal Axis
StartInitialTheta(1,1)=60;   %Degrees %Start point to search for limits
EndInitialTheta(1,1)=110;    %End point to search for limits

% Vertical Axis
StartInitialThetaDot(1,1)=-40;  %Degrees/sec %Start point to search for limits
EndInitialThetaDot(1,1)=40;     %End point to search for limits

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load SubjectMass   %Loads a mass matrix Nx1 with each row corresponding to
the mass of each subject

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Error Messages
if rem(FirstSubject,1)~=0 || rem(LastSubject,1)~=0 || rem(FirstCondition,1)~=0 ||
rem(FirstCondition,1)~=0
   fprintf('ALL First and Last values must be integers! \n')
   break
end
if LastSubject<FirstSubject
   fprintf('FirstSubject value must be less than or equal to the LastSubject value!
\n')
   break
end
if LastCondition<FirstCondition
```

```
    fprintf('FirstCondition value must be less than or equal to the LastCondition
value! \n')
    break
end
if FirstSubject<1 || FirstCondition<1 || LastSubject<1 || LastCondition<1
    fprintf('ALL First and Last values must be integers 1 or higher! \n')
    break
end
if size(SubjectMass,1)<FirstSubject || size(SubjectMass,1)<LastSubject
    %Question the User
    fprintf('The SubjectMass matrix must be a column vector with the number of
rows equal to the number of subjects (one mass value (kg) for each subject)! \n\n')
    fprintf('Would you like to use a temporary SubjectMass Matrix where each
subject is 70kg?'); fprintf('\n\n')
    fprintf('1 = Yes'); fprintf('\n')
    fprintf('2 = No'); fprintf('\n\n')
    fprintf('(Insert value and press enter)'); fprintf('\n')
    fileselector2 = input('Choice #'); fprintf('\n')
    if isempty(fileselector2)
        fileselector2 = 1;   %Simply pressing "enter" with no user input will say
"yes"
        fprintf('Question ACCEPTED due to no user input!')
        fprintf('\n\n\n')
    end

    fileselector2=round(fileselector2);    %rounds any decimal values to nearest
integer
    fprintf('Choice #')
    fprintf('%0.0f',fileselector2)
    fprintf(' was selected...')
    fprintf('\n\n')

    if fileselector2==1
        SubjectMass=ones(LastSubject,1)*70; %temporary SubjectMass matrix with
each subject at 70kg
    else
        fprintf('Please adjust the SubjectMass matrix to proceed! \n')
        break
    end

    %User input Error Messages
```

```
    if fileselector2>=3
        fprintf('This is not a valid option!')
        fprintf('\n\n\n')
    elseif fileselector2<=0
            fprintf('This is not a valid option!')
            fprintf('\n\n\n')
    end
end
if Forceplatefrequency<Camerafrequency
    fprintf('Forceplatefrequency must be higher than Camerafrequency for code to
run! \n')
    break
end
if rem(Forceplatefrequency/Camerafrequency,1)~=0
    fprintf('Forceplatefrequency/Camerafrequency value MUST be an integer for
code to run! \n')
    break
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NumberOfConditions = size(FirstCondition:LastCondition,2);
NumberOfSubjects = size(FirstSubject:LastSubject,2);

% Load Data and Filter

CalcAreaOfStability=zeros(NumberOfSubjects,3);
SubjectPlotCounter=0;
for SubjectCounter=FirstSubject:1:LastSubject
    SubjectPlotCounter=SubjectPlotCounter+1;
for ConditionCounter=FirstCondition:1:LastCondition

MarkersNumber = ['MarkersCondition' int2str(ConditionCounter) 'Subject'
int2str(SubjectCounter)];
Markers = load(MarkersNumber);
ForcePlateNumber = ['ForcePlateCondition' int2str(ConditionCounter) 'Subject'
int2str(SubjectCounter)];
Forceplatedatafull = load(ForcePlateNumber);

Markers = struct2cell(Markers);
Markers = cell2mat(Markers);
```

```
Forceplatedatafull = struct2cell(Forceplatedatafull);
Forceplatedatafull = cell2mat(Forceplatedatafull);

%%% Apply a Butterworth Filter for Forceplatedata

Forceplatedataupper=zeros(1,size(Forceplatedatafull,2));
Forceplatedatalower=zeros(1,size(Forceplatedatafull,2));
filtered_forceplatedata=zeros(size(Forceplatedatafull,1),size(Forceplatedatafull,2)
);
for j=1:1:size(Forceplatedatafull,2)    %does all columns of Forceplatedatafull
   Forceplatedataupper(j)=0;

   for i=1:1:size(Forceplatedatafull(:,j))
      if Forceplatedataupper(j)<1     %if upper bound is being chosen, lower bound
must have been chosen already
         Forceplatedatalower(j)=i;    %Locates the lower range of the data to filter
      end
      if isnan(Forceplatedatafull(i,j))==0 %if isnan=0, the value for
Forceplatedatafull is NOT NaN, hence, Forceplatedatafull IS a real number
         Forceplatedataupper(j)=i;  %Locates the upper range of the data to filter

      end
      if Forceplatedataupper(j)~=0 && isnan(Forceplatedatafull(i,j))==1 &&
isnan(Forceplatedatafull(i+1,j))==1
         break %stops the filtering as soon as NaN appears
      end
      %if a single row's value is missing, replace it with an average of
      %the previous and following row
      if Forceplatedataupper(j)~=0 && isnan(Forceplatedatafull(i,j))==1 &&
isnan(Forceplatedatafull(i+1,j))==0
         Forceplatedatafull(i,j)=(Forceplatedatafull(i-
1,j)+Forceplatedatafull(i+1,j))/2;
      end
   end

   filtered_forceplatedata(1:(Forceplatedatalower(j)-1),j)=NaN;

filtered_forceplatedata(Forceplatedatalower(j):Forceplatedataupper(j),j)=Criticall
yDampedButterworthLowPass(Forceplatedatafull(Forceplatedatalower(j):Forcepl
atedataupper(j),j),fs(1,1),fc(1,1));   %Calls the Butterpass function
```

```
filtered_forceplatedata((Forceplatedataupper(j)+1):size(Forceplatedatafull,1),j)=N
aN;
end

%%% Apply a Butterworth Filter for Marker data

Markersdataupper=zeros(1,size(Markers,2));
Markersdatalower=zeros(1,size(Markers,2));
filtered_markerdata=zeros(size(Markers,1),size(Markers,2));
for jj=1:1:size(Markers,2)    %does all columns of Markers
   Markersdataupper(jj)=0;

   for ii=1:1:size(Markers(:,jj))
      if Markersdataupper(jj)<1     %if upper bound is being chosen, lower bound
must have been chosen already
      Markersdatalower(jj)=ii;    %Locates the lower range of the data to filter
      end
      if isnan(Markers(ii,jj))==0 %if isnan=0, the value for Markers is NOT NaN,
hence, Markers IS a real number
         Markersdataupper(jj)=ii;  %Locates the upper range of the data to filter
      end
   end

   filtered_markerdata(1:(Markersdatalower(jj)-1),jj)=NaN;

filtered_markerdata(Markersdatalower(jj):Markersdataupper(jj),jj)=ButterworthL
owPassFilter3Dcoords(Markers(Markersdatalower(jj):Markersdataupper(jj),jj),fs(
2,1),fc(2,1));   %Calls the Butterpass function
   filtered_markerdata((Markersdataupper(jj)+1):size(Markers,1),jj)=NaN;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Find Upper and Lower bound rows of where the feet make contact with the
%%% force plates

Jumpsize=Forceplatefrequency/Camerafrequency;   %If frequencies differ, want
to align data rows correctly
ForcePlateAdj=zeros(size(filtered_markerdata,1),size(Forceplatedatafull,2));
```

```
ForcePlateAdjUpper=zeros(1,size(Forceplatedatafull,2));
ForcePlateAdjLower=zeros(1,size(Forceplatedatafull,2));
j=1;
for i=1:Jumpsize:size(filtered_forceplatedata,1)

   ForcePlateAdj(j,:)=filtered_forceplatedata(i,:);   %Take every jumpsize row
and make a new forceplate matrix
   j=j+1;

end

for j=1:1:size(ForcePlateAdj,2)    %does all columns of ForceplateAdj
   ForcePlateAdjUpper(j)=0;

   for i=1:1:size(ForcePlateAdj(:,j))
      if ForcePlateAdjUpper(j)<1     %if upper bound is being chosen, lower
bound must have been chosen already
         ForcePlateAdjLower(j)=i;    %Locates the row associated with the lower
range of the data
      end
      if isnan(ForcePlateAdj(i,j))==0 %if isnan=0, the value for Forceplatedatafull
is NOT NaN, hence, Forceplatedatafull IS a real number
         ForcePlateAdjUpper(j)=i;  %Locates the row where the toe lifts from 1st
force plate
      end
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Reset ForcePlateAdj to the desired range for SINGLE-LEG SUPPORT

%Decide which foot is lifting in beginning by checking vertical velocity
   i=ForcePlateAdjUpper(1);    %Uses position where COP on 1st Force Plate
goes to ZERO
   HeelVertVel=zeros(1,i);
   HeelVertAccel=zeros(1,i);
   h=1/Camerafrequency;
   VertVelTestLeft=(filtered_markerdata((ForcePlateAdjUpper(1)+1),9)-
filtered_markerdata((ForcePlateAdjUpper(1)-1),9))/(2*h);
```

```
   VertVelTestRight=(filtered_markerdata((ForcePlateAdjUpper(1)+1),12)-
filtered_markerdata((ForcePlateAdjUpper(1)-1),12))/(2*h);
   %If left foot is lifting
   if VertVelTestLeft>VertVelTestRight
      %Find when heel strike occurs
      while (1)
         HeelVertVel(i)=(filtered_markerdata((i+1),9)-filtered_markerdata((i-
1),9))/(2*h);
         HeelVertAccel(i)=(filtered_markerdata((i+1),9)-
2*filtered_markerdata(i,9)+filtered_markerdata((i-1),9))/(h^2);
         if abs(HeelVertVel(i))<200 && HeelVertAccel(i)>0 &&
filtered_markerdata(i,9)<100
            HeelStrike=i;
            break
         end
         i=i+1;
      end
   end

   %If right foot is lifting
   if VertVelTestRight>VertVelTestLeft
      %Find when heel strike occurs
      while (1)
         HeelVertVel(i)=(filtered_markerdata((i+1),12)-filtered_markerdata((i-
1),12))/(2*h);
         HeelVertAccel(i)=(filtered_markerdata((i+1),12)-
2*filtered_markerdata(i,12)+filtered_markerdata((i-1),12))/(h^2);
         if abs(HeelVertVel(i))<200 && HeelVertAccel(i)>0 &&
filtered_markerdata(i,12)<100
            HeelStrike=i-1;
            break
         end
         i=i+1;
      end
   end

ForcePlateAdj=ForcePlateAdj(ForcePlateAdjUpper(1):HeelStrike,:);
%ForcePlateAdjUpper(1)=toe off of force plate 1

%%% Convert filtered_data into individual columns
```

```
    % Force Plate 1 (:,1)

    copx(:,1) = ForcePlateAdj(:,1);    %  make 1st column of ForcePlateAdj "copx"

    % Force Plate 2 (:,2)

    copx(:,2) = ForcePlateAdj(:,2);    %  make 2nd column of ForcePlateAdj
"copx"
    copz(:,2) = ForcePlateAdj(:,3);
    fx(:,2) = ForcePlateAdj(:,4);
    fz(:,2) = ForcePlateAdj(:,5);
    my(:,2) = ForcePlateAdj(:,6);

Limited_MarkerData=filtered_markerdata(ForcePlateAdjUpper(1):HeelStrike,:);
%ForcePlateAdjUpper(1)=toe off of force plate 1

%%% Convert Markers

columncount=size(Limited_MarkerData,2);    %Finds number of columns in
filtered_markerdata
columncountadj=1:3:columncount;
Markersadj=zeros(size(Limited_MarkerData,1),length(columncountadj),3);
j=1;
for i=1:3:columncount    %i only selects x components from Markers (want every
3rd column (XyzXyzXyz)
    Markersadj(:,j,1)=Limited_MarkerData(:,i); %j makes each column of
Markersadj a different marker (hip/knee/ankle...)
    j=j+1;
end

j=1;
for i=2:3:columncount    %i only selects y components from Markers (want every
3rd column (xYzxYzxYz)
    Markersadj(:,j,2)=Limited_MarkerData(:,i);
    j=j+1;
end

j=1;
for i=3:3:columncount    %i only selects z components from Markers (want every
3rd column (xyZxyZxyZ)
    Markersadj(:,j,3)=Limited_MarkerData(:,i);
```

```
    j=j+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%Calculate angle (theta) for CoM pendulum in DEGREES
if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
    ThetaCOM=zeros(size(Markersadj(:,7,1),1),1);
    for i=1:1:(size(Markersadj(:,7,1)))
        ThetaCOM(i,1)=(atan2((Markersadj(i,7,3)-
Markersadj(i,1,3)),(Markersadj(i,7,1)-Markersadj(i,1,1)))); %7 for CoM, 1 for
LANK
    end
end

if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
    ThetaCOM=zeros(size(Markersadj(:,7,1),1),1);
    for i=1:1:(size(Markersadj(:,7,1)))
        ThetaCOM(i,1)=(atan2((Markersadj(i,7,3)-
Markersadj(i,2,3)),(Markersadj(i,7,1)-Markersadj(i,2,1)))); %7 for CoM, 2 for
RANK
    end
end

%Calculate Length from Ankle to CoM at each time interval
N=length(Markersadj(:,:,1));    %Makes for loop (below) stop when no more
Marker values

L=zeros(1,i);

if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
    for i=1:1:N
        L(i)=sqrt(((Markersadj(i,7,3)-Markersadj(i,1,3))^2+(Markersadj(i,7,1)-
Markersadj(i,1,1))^2)); %Changes length throughout based upon real subject
changes
    end
end

if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
    for i=1:1:N
```

```
        L(i)=sqrt(((Markersadj(i,7,3)-Markersadj(i,2,3))^2+(Markersadj(i,7,1)-
Markersadj(i,2,1))^2)); %Changes length throughout based upon real subject
changes
    end
end

anktorque=zeros(N,1);
Fx=zeros(N,1);
Fz=zeros(N,1);
La=zeros(N,1);
Ha=zeros(N,1);
theta=zeros(N+1,2);
x=zeros(N+1,2);
z=zeros(N+1,2);

%Set up Time Variables
dt=h;   %h=1/120. 120 = camera frequency

time=[0 20];    %20 is completely arbitrary and acts as safety net (prevents infinite
looping)
timebeg= time(1);
timeend= time(2);
t= (timebeg:dt:timeend)';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Convert all to radians from degrees
StartInitialTheta(1,2)=StartInitialTheta(1,1)*pi/180;
EndInitialTheta(1,2)=EndInitialTheta(1,1)*pi/180;
StartInitialThetaDot(1,2)=StartInitialThetaDot(1,1)*pi/180;
EndInitialThetaDot(1,2)=EndInitialThetaDot(1,1)*pi/180;

InitialTheta=StartInitialTheta(1,2);
Counter1=0;

while InitialTheta<=EndInitialTheta(1,2)

    %Initial x and z locations for pendulum
    theta0(1,1)=InitialTheta;
    if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
        x(1,1)= Markersadj(1,1,1);  %1 is for LANK
```

```
        z(1,1)= Markersadj(1,1,3);
    end
    if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
        x(1,1)= Markersadj(1,2,1);  %2 is for RANK
        z(1,1)= Markersadj(1,2,3);
    end
    x(1,2)= x(1,1)+(L(1)*cos(theta0(1,1)));
    z(1,2)= z(1,1)+(L(1)*sin(theta0(1,1)));
    Counter1=1+Counter1;
    Counter2=0;
    InitialThetaDot=StartInitialThetaDot(1,2);

    while InitialThetaDot<=EndInitialThetaDot(1,2)
    %Calculate inital angular velocity
    theta(1,1)= theta0; %initial angle
    h=1/Camerafrequency;    %120 = camera frequency
    theta(1,2)=InitialThetaDot;
    Counter2=1+Counter2;

for i = 1:1:N

        anktorque(i,1)=my(i,2)'; %units of N*mm
        Fx(i,1)=fx(i,2)';   %forces applied at COP location
        Fz(i,1)=fz(i,2)';

        if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
            Ha(i,1)=(Markersadj(i,1,3)-copz(i,2))';    %ankle z position - copz position
            La(i,1)=(copx(i,2)-Markersadj(i,1,1))';  %ankle x position - copz position
        end

        if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
            Ha(i,1)=(Markersadj(i,2,3)-copz(i,2))';    %ankle z position - copz position
            La(i,1)=(copx(i,2)-Markersadj(i,2,1))';  %ankle x position - copz position
        end

        K1=
penddyn(t,theta(i,:),anktorque(i),Fx(i),Fz(i),La(i),Ha(i),L(i),SubjectMass(Subject
Counter,1))';
            thetamid= K1*(dt/2)+theta(i,:);
```

```
        K2=
penddyn(t+dt/2,thetamid,anktorque(i),Fx(i),Fz(i),La(i),Ha(i),L(i),SubjectMass(Su
bjectCounter,1))';
            thetamid= K2*(dt/2)+theta(i,:);
        K3=
penddyn(t+dt/2,thetamid,anktorque(i),Fx(i),Fz(i),La(i),Ha(i),L(i),SubjectMass(Su
bjectCounter,1))';
            thetaend= K3*(dt)+theta(i,:);
        K4=
penddyn(t+dt,thetaend,anktorque(i),Fx(i),Fz(i),La(i),Ha(i),L(i),SubjectMass(Subj
ectCounter,1))';
            theta(i+1,:)= theta(i,:)+(dt/6)*(K1+2*K2+2*K3+K4);

    %Subsequent x and z locations of ankle (1) and CoM (2) for animate.m
    if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
        x(i+1,1)=Markersadj(i,1,1);
        z(i+1,1)=Markersadj(i,1,3);
    end
    if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
        x(i+1,1)=Markersadj(i,2,1);
        z(i+1,1)=Markersadj(i,2,3);
    end
    x(i+1,2)=x(i+1,1)+(L(i)*cos(theta(i+1,1)));
    z(i+1,2)=z(i+1,1)+(L(i)*sin(theta(i+1,1)));

        if timebeg>=timeend,break,end
        if timebeg>(timeend-dt),break,end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate the XcoM

    g=9.81*1000; %units of mm/s^2
    COMvel=zeros(1,N-1);
    EigenFrequency=zeros(1,N-1);
    XcoM=zeros(1,N-1);
    BoundaryTest=zeros(1,N-1);
    if InitialTheta==StartInitialTheta(1,2) &&
InitialThetaDot==StartInitialThetaDot(1,2);
```

```
        ExitPoint=zeros(EndInitialTheta(1,1)-
StartInitialTheta(1,1)+1,EndInitialThetaDot(1,1)-StartInitialThetaDot(1,1)+1);
        ExitSide=zeros(EndInitialTheta(1,1)-
StartInitialTheta(1,1)+1,EndInitialThetaDot(1,1)-StartInitialThetaDot(1,1)+1);
    end
    for i=2:1:(N-1)
        COMvel(i)=(x(i+1,2)-x(i-1,2))/(2*dt);
        EigenFrequency(i)=sqrt(g/L(i));
        XcoM(i)=x(i,2)+(COMvel(i)/EigenFrequency(i));
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Make a Boundaries Matrix
LeadFootPass=0;

    for i=2:1:(N-1)
        if VertVelTestRight>VertVelTestLeft %if Left Foot On Ground
            BoundaryTest(1,i)=XcoM(i)-Markersadj(i,5,1);    %XcoM x position - toe
x position
            BoundaryTest(2,i)=XcoM(i)-Markersadj(i,3,1);    %XcoM x position -
heel x position
        end
        if VertVelTestRight<VertVelTestLeft %if Right Foot On Ground
            BoundaryTest(1,i)=XcoM(i)-Markersadj(i,6,1);    %XcoM x position - toe
x position
            BoundaryTest(2,i)=XcoM(i)-Markersadj(i,4,1);    %heel x position -
XcoM x position
        end

        if BoundaryTest(1,i)>=0 && ExitPoint(Counter1,Counter2)==0
            ExitPoint(Counter1,Counter2)=i;    %chooses the point where instability
occurs
            ExitSide(Counter1,Counter2)=1;     %1 = XcoM passed the toe marker
(falling forwards)
        end
        if BoundaryTest(2,i)<=0 && ExitPoint(Counter1,Counter2)==0
            ExitPoint(Counter1,Counter2)=i;    %chooses the point where instability
occurs
            ExitSide(Counter1,Counter2)=2;     %2 = XcoM passed the heel marker
(falling backwards)
```

```
        end
    end
    InitialThetaDot=InitialThetaDot+(1*pi/180);
    end
    InitialTheta=InitialTheta+(1*pi/180);
end

% Find upper and lower limits for each theta value
LimitTester=zeros(size(ExitSide,1),size(ExitSide,2)-1);
Limit=zeros(size(ExitSide,1)-1,2);
for i=1:1:Counter1 %do all rows for theta
    for j=2:1:Counter2 %do all columns for thetadot
        LimitTester(i,j-1)=ExitSide(i,j)-ExitSide(i,j-1);
        if LimitTester(i,j-1)==-1
            Limit(i,1)=j;   %column 1 = lower limits
        end
        if LimitTester(i,j-1)==2
            Limit(i,2)=j-1; %column 2 = upper limits
            break
        end
    end
end

% Create Limits Plot
ThetaXValues(:,1)=zeros(size(ExitSide,2),1);
j=1;
for i=StartInitialTheta(1,1):1:EndInitialTheta(1,1)
ThetaXValues(j,1)=i;
j=j+1;
end

ThetaVelYValues(:,1)=StartInitialThetaDot(1,1)+Limit(:,1)-1;
ThetaVelYValues(:,2)=StartInitialThetaDot(1,1)+Limit(:,2)-1;

j=1;
% for i=1:1:size(ThetaXValues,1)
for i=1:1:size(ThetaVelYValues,1)
    if Limit(i,2)>0 && Limit(i,1)>0
        Xlimits(j,1)=ThetaXValues(i,1);
        Ylimits(j,:)=ThetaVelYValues(i,:);
        plot(Xlimits(j,1),Ylimits(j,1),'ro-')
```

```
        hold on;
        plot(Xlimits(j,1),Ylimits(j,2),'o-')
        j=j+1;
    end
end
    plot(Xlimits(:,1),Ylimits(:,1),'r-')
    plot(Xlimits(:,1),Ylimits(:,2),'-')
%     jbfill(Xlimits(:,1)',Ylimits(:,2)',Ylimits(:,1)','c');
    xlabel('Initial Anglar Position Theta0 (degrees)')
    ylabel('Initial Angular Velocity ThetaDot0 (degrees/sec)')
    title('Limits of Stability During Single-Leg Support','FontSize',12)
    LimitLegend = legend('Toe Limit' , 'Heel Limit');
    set(LimitLegend,'Location','NorthEast')
    text(0,.1,num2str(i));

%Calculate Areas of Stability
AdjYlimits=Ylimits;
for i=1:1:size(Ylimits,1)
    for j=1:1:2
        if Ylimits(i,1)<0
            AdjYlimits(i,j)=Ylimits(i,1)*-1;
        end
    end
end

AreaUpper=sIntegrate( AdjYlimits(:,2), 1, size(Ylimits,1),1);
AreaLower=sIntegrate( AdjYlimits(:,1), 1, size(Ylimits,1),1);

CalcAreaOfStability(SubjectPlotCounter,ConditionCounter)=AreaUpper-
AreaLower;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clearvars -except NumberOfConditions NumberOfSubjects CalcAreaOfStability
Forceplatefrequency Camerafrequency SubjectCounter SubjectMass fs fc
FirstSubject FirstCondition LastSubject LastCondition SubjectPlotCounter
StartInitialTheta StartInitialThetaDot EndInitialTheta EndInitialThetaDot

end
end
```

```
% Turn any zero value into NaN (Not a Number)
for iii=1:1:NumberOfSubjects
   for jjj=1:1:3   %goes through 3 conditions
      if CalcAreaOfStability(iii,jjj)==0
         CalcAreaOfStability(iii,jjj)=NaN;
      end
   end
end

%Print the Area of Stability Result
fprintf('\n o--------------------------------------------------o \n')
fprintf('  Each Row is a Subject. Each Column is a Condition. \n\n')
fprintf('  Area Of Stability (degrees^2/sec) = \n')
for i=1:size(CalcAreaOfStability,1)
fprintf('  ')
fprintf( '%f\t',CalcAreaOfStability(i,:)) %f for fixed point notation
fprintf('\n\n')
end
fprintf(' o--------------------------------------------------o \n')
clearvars -except CalcAreaOfStability

toc
```

References

Abella, B.S., Sandbo, N., Vassilatos, P., Alvarado, J.P., O'Hearn, N., Wigder, H.N., Hoffman, P., Tynus, K., Vanden Hoek, T.L., Becker, L.B. 2005. Chest Compression Rates During Cardiopulmonary Resuscitation Are Suboptimal: A Prospective Study During In-Hospital Cardiac Arrest. Circulation. 111, 428-434.

Barry, B.K., Warman, G.E., Carson, R.G. 2005. Age-related differences in rapid muscle activation after rate of force development training of the elbow flexors. Exp Brain Res. 162, 122-132.

Berg, W.P., Alessio, H., M., Mills, E., M., Tong, C. 1997. Circumstances and consequences of falls in independent community-dwelling older adults. Age and Ageing. 26, 261-268.

Bhatt, T., Espy, D., Yang, F., Pai, Y.-C. 2011. Dynamic Gait Stability, Clinical Correlates, and Prognosis of Falls Among Community-Dwelling Older Adults. Archives of Physical Medicine and Rehabilitation. 92, 799-805.

Bierbaum, S., Peper, A., Karamanidis, K., Arampatzis, A. 2011. Adaptive feedback potential in dynamic stability during disturbed walking in the elderly. Journal of Biomechanics. 44, 1921-1926.

Carty, C.P., Bennett, M.B. 2009. The use of dimensionless scaling strategies in gait analysis. Human Movement Science. 28, 218-225.

Chiu, M.-C., Wu, H.-C., Chang, L.-Y., Wu, M.-H. In Press. Center of pressure progression characteristics under the plantar region for elderly adults. Gait & Posture.

Chou, L.-S., Kaufman, K.R., Hahn, M.E., Brey, R.H. 2003. Medio-lateral motion of the center of mass during obstacle crossing distinguishes elderly individuals with imbalance. Gait & Posture. 18, 125-133.

Cromwell, R.L., Newton, R.A. 2004. Relationship between balance and gait stability in healthy older adults. Journal of Aging and Physical Activity. 12, 90-100.

Dingwell, J.B., Marin, L.C. 2006. Kinematic variability and local dynamic stability of upper body motions when walking at different speeds. Journal of Biomechanics. 39, 444-452.

England, S.A., Granata, K.P. 2007. The influence of gait speed on local dynamic stability of walking. Gait & Posture. 25, 172-178.

Fukagawa, N.K., Schultz, A.B. 1995. Muscle Function and Mobility Biomechanics in the Elderly: An Overview of Some Recent Research. The Journals of Gerontology Series A: Biological Sciences and Medical Sciences. 50A, 60-63.

Grabiner, M.D., Owings, T.M., Pavol, M.J. 2005. Lower extremity strength plays only a small role in determining the maximum recoverable lean angle in older adults. J Gerontol A Biol Sci Med Sci. 60, 1447-1450.

Gurfinkel, E.V. 1973. Physical foundations of stabilography. Agressologie. 14, 9-13.

Hemami, H., Golliday Jr, C.L. 1977. The inverted pendulum and biped stability. Mathematical Biosciences. 34, 95-110.

Hof, A.L. 1996. Scaling gait data to body size. Gait & Posture. 4, 222-223.

Hof, A.L., Gazendam, M.G., Sinke, W.E. 2005. The condition for dynamic stability. Journal of Biomechanics. 38, 1-8.

Huang, S.-C., Lu, T.-W., Chen, H.-L., Wang, T.-M., Chou, L.-S. 2008. Age and height effects on the center of mass and center of pressure inclination angles during obstacle-crossing. Medical Engineering & Physics. 30, 968-975.

Hurley, M.V., Rees, J., Newham, D.J. 1998. Quadriceps function, proprioceptive acuity and functional performance in healthy young, middle-aged and elderly subjects. Age and Ageing. 27, 55-62.

Ihlen, E.A.F., Sletvold, O., Goihl, T., Wik, P.B., Vereijken, B., Helbostad, J. 2012. Older adults have unstable gait kinematics during weight transfer. Journal of Biomechanics. 45, 1559-1565.

Karamanidis, K., Arampatzis, A. 2007. Age-related degeneration in leg-extensor muscle–tendon units decreases recovery performance after a forward fall: compensation with running experience. European Journal of Applied Physiology. 99, 73-85.

Karamanidis, K., Arampatzis, A., Mademli, L. 2008. Age-related deficit in dynamic stability control after forward falls is affected by muscle strength and tendon stiffness. Journal of Electromyography and Kinesiology. 18, 980-989.

Laufer, Y. 2005. Effect of Age on Characteristics of Forward and Backward Gait at Preferred and Accelerated Walking Speed. The Journals of Gerontology Series A: Biological Sciences and Medical Sciences. 60, 627-632.

Lee, W.A., Patton, J.L. 1997. Learned changes in the complexity of movement organization during multijoint, standing pulls. Biological Cybernetics. 77, 197-206.

Li, L., Haddad, J.M., Hamill, J. 2005. Stability and variability may respond differently to changes in walking speed. Human Movement Science. 24, 257-267.

Lord, S.R., Lloyd, D.G., Keung Li, S. 1996. Sensori-motor Function, Gait Patterns and Falls in Community-dwelling Women. Age and Ageing. 25, 292-299.

Luukinen, H., Koski, K., Hiltunen, L., Kivela, S.L. 1994. Incidence rate of falls in an aged population in northern Finland. J Clin Epidemiol. 47, 843-850.

Marigold, D.S., Patla, A.E. 2002. Strategies for dynamic stability during locomotion on a slippery surface: effects of prior experience and knowledge. J Neurophysiol. 88, 339-353.

Meriam, J.L., Kraige, L.G. 2009. Engineering Mechanics: Dynamics. John Wiley & Sons, Inc, Hoboken, NJ, pp. 720.

Moyer, B.E., Chambers, A.J., Redfern, M.S., Cham, R. 2006. Gait parameters as predictors of slip severity in younger and older adults. Ergonomics. 49, 329-343.

Niino, N., Tsuzuku, S., Ando, F., Shimokata, H. 2000. Frequencies and circumstances of falls in the National Institute for Longevity Sciences, Longitudinal Study of Aging (NILS-LSA). J Epidemiol. 10, S90-94.

Pai, Y.C., Wening, J.D., Runtz, E.F., Iqbal, K., Pavol, M.J. 2003. Role of feedforward control of movement stability in reducing slip-related balance loss and falls among older adults. J Neurophysiol. 90, 755-762.

Patel, M., Magnusson, M., Kristinsdottir, E., Fransson, P.A. 2009. The contribution of mechanoreceptive sensation on stability and adaptation in the young and elderly. Eur J Appl Physiol. 105, 167-173.

Patton, J.L., Lee, W.A., Pai, Y.C. 2000. Relative stability improves with experience in a dynamic standing task. Exp Brain Res. 135, 117-126.

Patton, J.L., Pai, Y., Lee, W.A. 1999. Evaluation of a model that determines the stability limits of dynamic balance. Gait & Posture. 9, 38-49.

Pierrynowski, M.R., Galea, V. 2001. Enhancing the ability of gait analyses to differentiate between groups: scaling gait data to body size. Gait & Posture. 13, 193-201.

Pijnappels, M., Bobbert, M.F., Dieen, J.H.v. 2005. Push-off reactions in recovery after tripping discriminate young subjects, older non-fallers and older fallers. Gait & Posture. 21, 388-394.

Popovic, M.R., Pappas, I.P.I., Nakazawa, K., Keller, T., Morari, M., Dietz, V. 2000. Stability criterion for controlling standing in able-bodied subjects. Journal of Biomechanics. 33, 1359-1368.

Robertson, D.G., Dowling, J.J. 2003. Design and responses of Butterworth and critically damped digital filters. J Electromyogr Kinesiol. 13, 569-573.

Scott, G., Menz, H.B., Newcombe, L. 2007. Age-related differences in foot structure and function. Gait & Posture. 26, 68-75.

Seidler, R.D., Bernard, J.A., Burutolu, T.B., Fling, B.W., Gordon, M.T., Gwin, J.T., Kwak, Y., Lipps, D.B. 2010. Motor control and aging: links to age-related brain structural, functional, and biochemical effects. Neurosci Biobehav Rev. 34, 721-733.

Stel, V.S., Smit, J.H., Pluijm, S.M.F., Lips, P. 2004. Consequences of falling in older men and women and risk factors for health service use and functional decline. Age and Ageing. 33, 58-65.

Sterling, D.A., O'Connor, J.A., Bonadies, J. 2001. Geriatric falls: injury severity is high and disproportionate to mechanism. Journal of Trauma. 50, 116-119.

Townsend, M.A. 1985. Biped gait stabilization via foot placement. Journal of Biomechanics. 18, 21-38.

van der Linden, M.H., Marigold, D.S., Gabreëls, F.J.M., Duysens, J. 2007. Muscle Reflexes and Synergies Triggered by an Unexpected Support Surface Height During Walking. Journal of Neurophysiology. 97, 3639-3650.

van Wegen, E.E.H., van Emmerik, R.E.A., Riccio, G.E. 2002. Postural orientation: Age-related changes in variability and time-to-boundary. Human Movement Science. 21, 61-84.

Vandervoort, A.A., McComas, A.J. 1986. Contractile changes in opposing muscles of the human ankle joint with aging. J Appl Physiol. 61, 361-367.

Winter, D.A. 1995. A.B.C. (Anatomy, Biomechanics and Control) of Balance during Standing and Walking. Waterloo Biomechanics, Waterloo, Canada.

Winter, D.A. 2009. Biomechanics and Motor Control of Human Movement. John Wiley & Sons, Inc, Hoboken, pp. 370.

Wood, G.A. 1982. Data smoothing and differentiation procedures in biomechanics. In: Terjung, R.J., (Eds.), Exercise and Sport Sciences Reviews. Vol. 10. Collamore Press, Lexington, pp. 308-362.

Biographical Information

Clifford Lee Hancock completed his Bachelor of Science in Engineering Science and Mechanics within the Biomechanics Track at Virginia Polytechnic Institute and State University (Virginia Tech) in May of 2011. He subsequently attended the University of Texas at Arlington to obtain a Master's degree within the Bioengineering Department by following the Biomechanics Track. He plans to get hired as a biomechanical engineer to apply his knowledge of biomechanics, mathematics, and various software in the research of human gait or in the development of mechanical components, implant devices, or prosthetics.