Towards Personalized Powered Ankle-Foot Prostheses

Jonathan Realmuto

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

Doctor of Philosophy

University of Washington

2017

Reading Committee:

Santosh Devasia, Chair

Glenn Klute, Chair

Brian Fabian

Kat Steele

Program Authorized to Offer Degree:
UW Department of Mechanical Engineering

University of Washington

Abstract

Towards Personalized Powered Ankle-Foot Prostheses

Jonathan Realmuto

Co-Chairs of the Supervisory Committee:
Professor Santosh Devasia
Department of Mechanical Engineering

Affiliated Professor Glenn Klute
Department of Mechanical Engineering

This research aims to develop a personalized powered ankle-foot prosthesis. Unilateral below-knee amputees generally have walking gait patterns with temporal, kinematic, and kinetic asymmetries resulting from neuromuscular adaptations required to compensate for the lack of ankle function leading to a host of secondary musculoskeletal impairments. Powered ankle-foot prostheses can restore some ankle function by active compensation at the prosthetic joint. However, current methods are limited by: (i) reliance on able-bodied gait data for trajectory synthesis, (ii) expert tuning of subject specific parameters, (iii) an inability to adapt with the user in the long term, and (iv) a limited amount of mechanical personalization. This work seeks to directly address the first two limitations with a symmetry learning controller that automatically tunes control trajectories by iteratively adjusting the prosthetic ankle torque to match the intact ankle torque in the frequency domain. The challenge is to avoid divergence caused by the time-vary human-robot dynamics. Towards this, a rule based iterative learning algorithm is introduced that adjusts the frequency dependent learning rate based on the changes in error. The symmetry controller could conceivably be adapted for indefinite use e.g., by caching, storing, and rewriting training instances, and thus address the third limitation. To reducing the active requirement of the device, a nonlinear cam-based

spring that can be customized to the user and/or activity is introduced, which provides some level of passive personalization (the fourth limitation). The new method enables symmetric ankle moment control and is tailored to the individual. The method was implemented and experimentally verified. Results indicated that symmetry of the ankle moment was significantly increased and unbounded growth of the control signal was avoided.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DEDICATION

to my very first advisor, my dear mother Diva

Chapter 1
INTRODUCTION

1.1 Motivation

1.1.1 Consequences of below-knee amputation

One of the most significant consequences of below-knee amputations is asymmetrical loading of the lower limbs during ambulation. Asymmetrical limb loading has been suggested to predispose below-knee amputees to secondary conditions [1], including degenerative joint diseases in the lower limbs and lower back, such as osteoarthritis (OA) in their intact knee and hip joints, osteopenia and subsequent osteoporosis in the residual limb, and back pain [2]. Typically, below-knee amputees use passive foot prostheses, which, as of yet, cannot completely replicate human ankle function. Ankle plantar flexors produce over 80% of the mechanical power generated during walking [3], and as a result below-knee amputees have distinctly asymmetrical gaits, e.g., temporal, kinematic or kinetic differences between limbs, that arise from neuromuscular adaptations compensating for the loss of ankle function [4–7]. The inability to actively produce ankle power results in a variety of compensatory behaviors including spending less time in stance phase on their prosthetic limb [8–10], loading their prosthetic limb less that their intact limb [11–14], loading their intact limb more relative to people without lower-limb amputations [15], and increased hip extension and knee flexion in the intact limb [6, 16]. In addition to symmetry issues below-knee amputees require 10-60% more metabolic energy and have an 11-40% slower preferred walking speed when compared with non-amputees, [8, 17–22]. Prostheses with improved push-off characteristics have been shown to reduce some biomechanical risk factors linked to OA development [20, 23], and increase walking economy and preferred walking speed [24], thus power ankle-foot prostheses (PAFPs) may provide substantial benefits to the user, including increased mobility and

prevention of future secondary musculoskeletal conditions.

1.1.2 Limitations of the state-of-the-art

PAFPs can restore some ankle function by active compensation at the prosthetic joint [25–31]. Despite that, current approaches are limited by:

- **limited structural personalization**

  Emerging PAFPs have limited amounts of structural customization. Although many PAFPs utilize mechanisms, e.g., springs, clutches, and linkages, optimized to reduce the actuator requirements [30–33], such approaches aren't easily modified to the characteristics of the user and/or activity. The passive dynamics largely determine the active requirement [34], thus a more customized approach could improve performance by reducing device weight, increasing battery life, and increasing user comfort.

- **reliance on able-bodied gait data**

  Many PAFPs synthesize controller trajectories based on averaged able-bodied gait data [30, 31, 33, 35]. However, metabolic rate, gait mechanics and muscle activity can vary widely across users, thus (able-bodied) push-off work should not be the only consideration when formulating control trajectories; other factors, including alternate timing, personalized tuning, and personalized subject characteristics should be taken into consideration [36].

- **expert tuning**

  Nominal able-bodied trajectories still need some level of customization, usually accomplished through trial and error [35,37]. Commercial PAFPs (e.g., BiOM, BionX Medical Technologies Inc.) must be tuned by a certified prosthetist through their experience and the user's feedback. However, tuning during clinical evaluations fail to demonstrate the same benefits in walking economy as found in the laboratory [38], highlighting the challenge of discovering optimal user parameters.

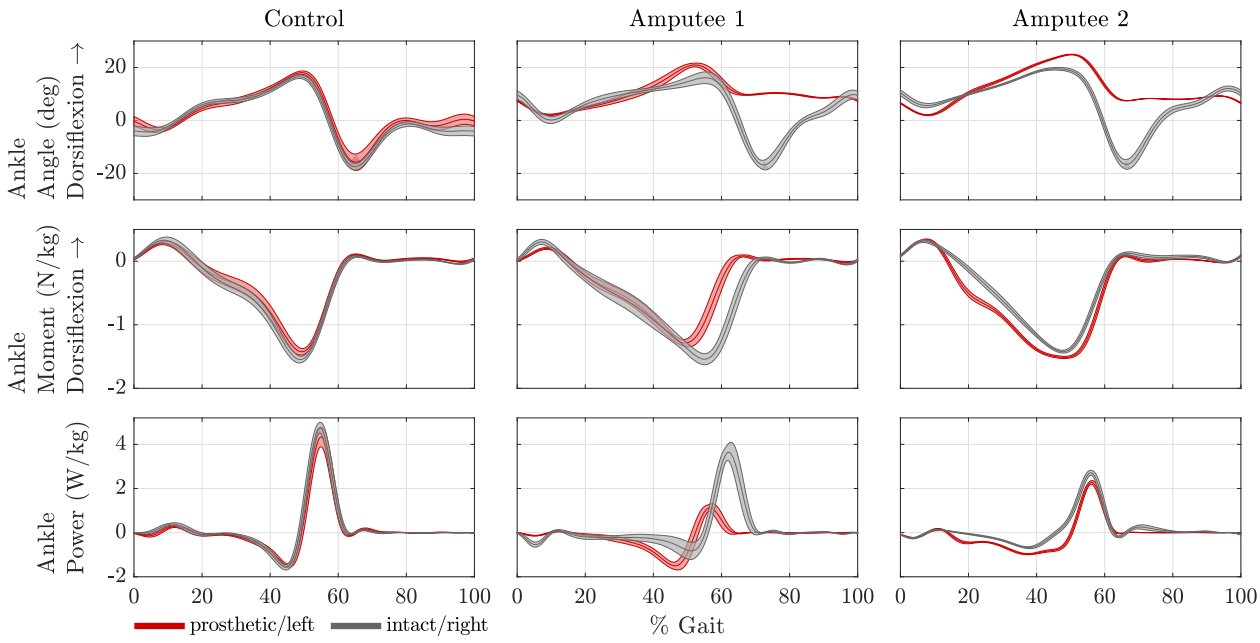


Figure 1.1: Comparison of ankle mechanics for a non-amputee (control) and two amputees using passive foot prostheses. Thickness is ±1 standard deviation. Data collected at self-selected walking speed, during treadmill walking, over 30 (s).

- **inability for adaptations**

  PAFPs today have a limited amount of adaptations, restricted to slopes [39] and speed [35] changes. A human-in-the-loop algorithm, where the control signal is modified based on the achieved human-robot response, could allow for long term adaptations [40].

In summary, PAFPs have a limited amount of **personalization**.

1.1.3 The need for personalization

Figure 1.1, shows ankle mechanics for a non-amputee and two below-knee amputees using passive prostheses during self selected speed treadmill walking. The plots show left/prosthetic limb (red) and right/intact limb (grey) ankle angles, ankle torques normalized by mass, and ankle powers normalized by mass. The non-amputee (control) has a relatively symmetrical

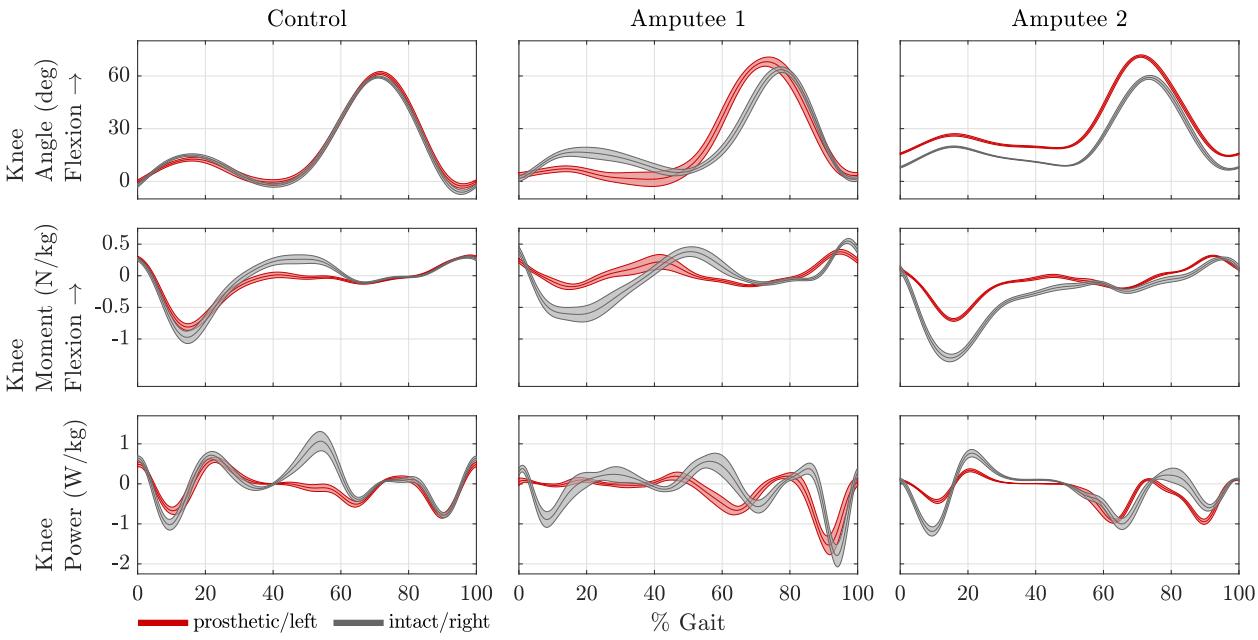


Figure 1.2: Comparison of ankle mechanics for a non-amputee (control) and two amputees using passive foot prostheses. Thickness is ±1 standard deviation. Data collected at self-selected walking speed, during treadmill walking, over 30 (s).

ankle angle, torque, and power. It can be seen that Amputee 1 (middle column) has large asymmetries in ankle angle, torque and power. On the other hand, Amputee 2 (right column) has asymmetrical ankle angle, but less severe asymmetries in torque and power. Knee mechanics, shown in Fig. 1.2, and hip mechanics, shown in Fig. 1.3, exhibit a similar pattern: the non-amputee is mostly symmetrical, while the amputees have substantial kinematic and kinetic differences between limbs. Note that the two amputees' walking strategies are considerably different. For example, the intact knee flexion moment (Fig. 1.2 middle plots) early in the gait cycle is relatively small for Amputee 1, and relatively large in Amputee 2. Additionally, the prosthetic and intact hip angle (Fig. 1.3 top plots) don't reach extension (e.g., $< 0^o$) for Amputee 2, while Amputee 1 hip angles are closer to that of the control. These differences provide qualitative evidence of the need for personalized approaches, as every individual has a unique neuromuscular strategy [41].

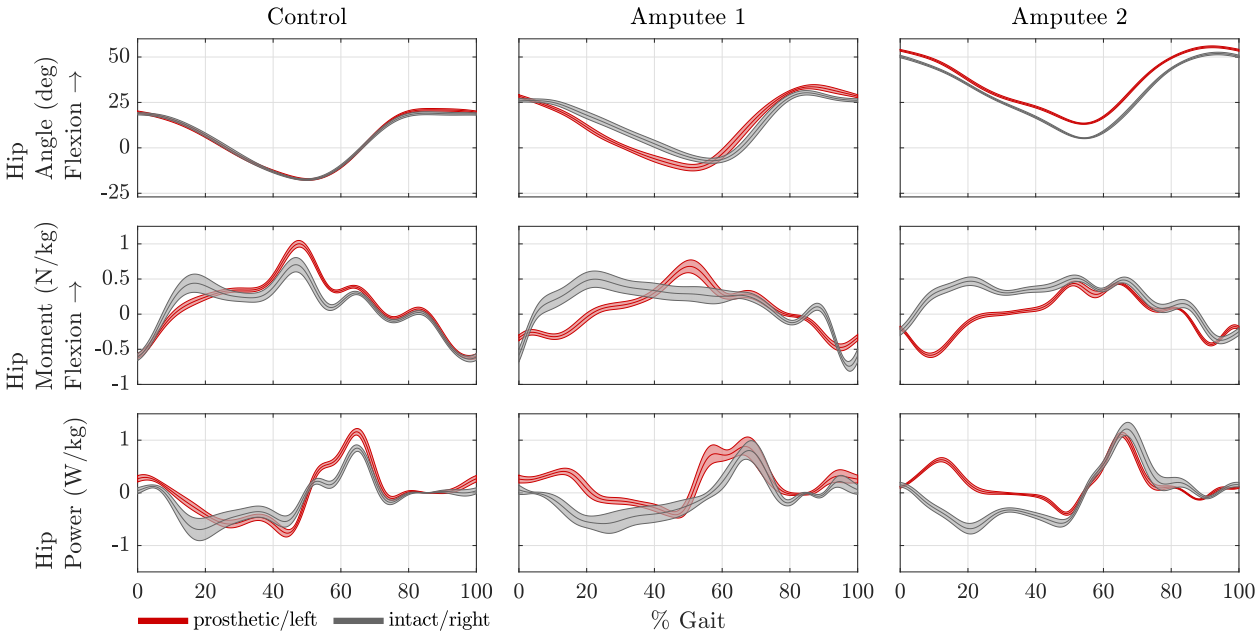


Figure 1.3: Comparison of hip mechanics for a non-amputee (control) and two amputees using passive foot prostheses. Thickness is ±1 standard deviation. Data collected at self-selected walking speed, during treadmill walking, over 30 (s).

1.2 Research goals

To address the need for personalization, the goals of this research are:

- **personalizing the passive response**

  Towards this, a cam-based nonlinear spring is proposed to optimally match a desired nonlinear passive response in order to personalized PAFPs. The use of a cam allows for a modular design, thus only the cam itself need be exchanged for customization, and not the whole device.

- **personalized learning controller**

  Towards this, a symmetry controller is developed that learns the prostheses control signal that results in symmetrical ankle torques. The rational is two fold: (i) asymmetries are functionally limiting and are thought to contribute to secondary conditions, thus

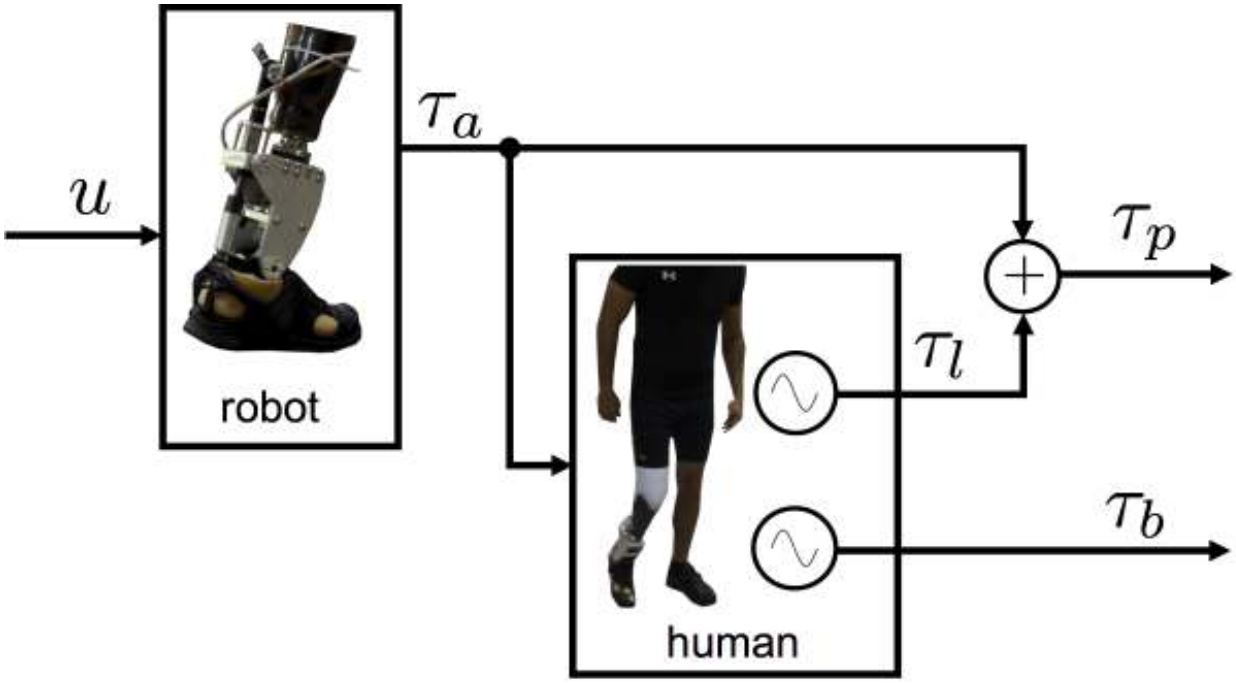


Figure 1.4: Conceptual model illustrating the research problem. The goal is to find the control signal u such that the active prosthetic torque τ_a, summed with the (loading) passive prosthetic torque τ_l, results in a total prosthetic torque τ_p that matches the biological torque τ_b. Note that the conceptual model accounts for the effect of the active prosthetic torque τ_a on both the loading prosthetic torque τ_l and the biological torque τ_b

addressing asymmetries could have long-term clinical implications, and (ii), asymmetries could be accurately estimated online, e.g., with low-cost embedded insole sensors, as opposed to other metrics such as walking economy, which is notoriously hard to measure, especially outside of the laboratory [42].

- **experimental evaluation**

  Towards this goal, the prototype device will be experimentally assessed and the following hypotheses are evaluated:

  H1: *The symmetry controller will result in a reduction of ankle asymmetries (e.g.,*

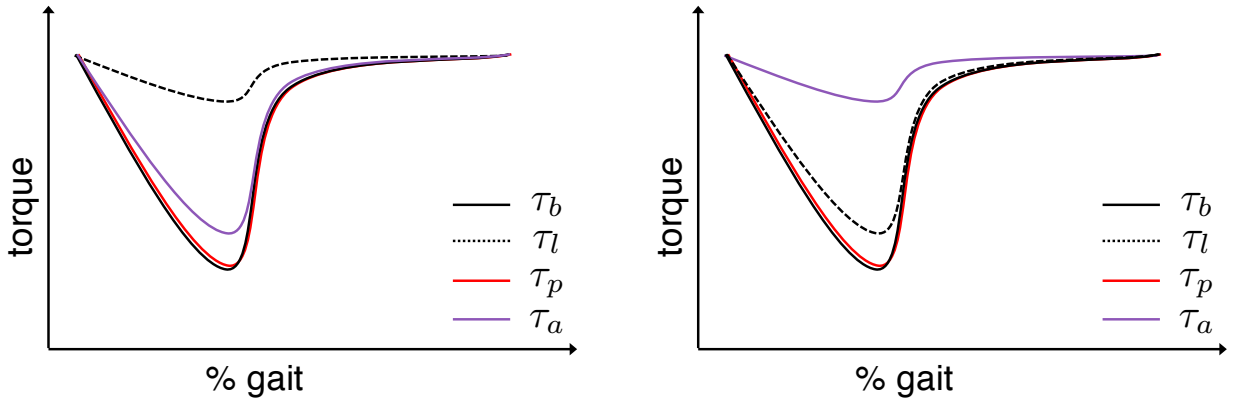
angle, torque and power).

H2: *The symmetry controller will result in a reduction of peak intact knee and hip adduction moments (torques), which are loading factors associated with OA.*

The conceptual model in Fig. 1.4 illustrates the research problem, which is to choose the control signal u such that the prosthetic torque τ_p matches the biological ankle torque τ_b. The powered prostheses (robot) takes as an input the control signal u and outputs an active torque τ_a about the prosthetic ankle joint. In addition, the prosthesis provides a passive or loading torque τ_l, which sums with the active torque τ_a to produce the total prosthetic torque τ_p. The basic approach is to iteratively learn the control signal u by using the measured biological torque signal τ_b as a reference. Note, that the active torque τ_a effects both the loading torque τ_l and the biological torque τ_b. Additionally, the research goal includes making τ_l (e.g., the passive prosthetic torque) as close as possible to τ_b by customizing the passive response of the device.

1.2.1 Addressing the limitations of the state-of-the-art

The research goals address the limitations, outlined in Sect. 1.1.2, as follows. A customized passive structure allows a more personalized passive response, addressing the first limitation. This, in turn, can reduce the active torque requirement of the prosthesis. Reducing the amount of required active torque has two benefits: (i) the device can weigh less by reducing the motor size, (ii) the passive response can better approximate the user's intact ankle response thereby reducing the amount of learning (e.g., if the prosthetic torque is already very close to the intact torque the controller only has to learn a small amount). The symmetry learning controller directly address the limitations of using able-bodied gait data (second limitation) and the need for expert tuning (third limitation), since the learning controller automatically tunes the control trajectory. Conceivably, the symmetry learning controller could allow for long term adaptations (the fourth limitation). For example, the user could



(a) (b)

Figure 1.5: The need for reducing active torque requirement. Illustrative comparison: (a) active torque τ_a required for the prosthetic torque τ_p to match the biological torque τ_b when the loading torque is small τ_l, and (b) active torque τ_a required for the prosthetic torque τ_p to match the biological torque τ_b when the loading torque is large τ_l.

activate the learning controller when needed and the controller could cache learned instances (e.g., a trajectory library). A method of interpolation could be employed to reduce the need of learning every possible scenario (e.g., speed or load carriage). Additionally, if the user's gait adapts, the cache could overwrite trajectories, thereby allowing for longterm adaptations.

1.3 Challenges

1.3.1 Personalizing the passive response for reduced torque requirement

Typically, motor mass scales propotionally with the motor's peak-torque, with high performance motors scaling slightly worst [43]. Hence, by reducing the active peak-torque requirement, e.g., with passive components that can store and release energy, the overall mass of the device can be reduced. In addition, if the passive components are designed to approximate the biological torque profile of the user as close as possible (e.g., personalized), then the amount

of learning required to achieve symmetry can be reduced – that is to say, with an iterative approach if the error is initially small then the amount of learning is also small. This idea is illustrated in Fig. 1.5, which shows the required active torque τ_a when the loading torque τ_l is small (Fig. 1.5a) and when the load torque τ_l is large (Fig. 1.5b). The main challenge is to design the passive components of the power ankle-foot prosthesis to best approximate the user's biological ankle dynamics.

1.3.2 Ensuring control signal boundedness

An important attribute of a prosthesis is user safety. This means the control signal u in Fig. 1.4 must remain bounded e.g., since the active torque τ_a affects the biological torque τ_b, some caution is necessary when iteratively changing the control signal u. This concept is shown in Fig. 1.6, which provides an illustrative example of how the torque signals could change iteratively. Initially (Fig. 1.6a), at iteration $k = 0$, the active torque $\tau_a = 0$, and there is an obvious difference in prosthetic torque τ_p and τ_b. Suppose we add the active torque $\tau_a = \tau_b - \tau_p$. Now, at iteration $k = 1$ (Fig. 1.6b) the added active torque τ_a results in an prosthetic torque τ_p that is closer to the desired biological torque τ_b. During the next iteration $k = 2$ (Fig. 1.6b), the procedure is repeated, and more active torque τ_a has been added. Again, the torque difference is reduced, albeit the prosthetic torque τ_p and the biological torque τ_b have increased in magnitude. By iteration $k = 3$ (Fig. 1.6d) the total error is small, e.g., $\tau_p \approx \tau_b$, but the signals are growing. In the worst case, both τ_p and τ_b could grow unbounded. However, if we assume the human can maintain stability (e.g., bound the reference signal τ_b) for small fluctuation in control input u, then ensuring bounded control signal u would result in a bounded reference signal τ_b. The main challenge is to guarantee boundedness of the control signal.

1.4 Thesis outline

An overview of the research path is illustrated in Fig. 1.7. Chapter 2 reviews the relevant literature and introduces the reader to important concepts, including ankle mechanics, an

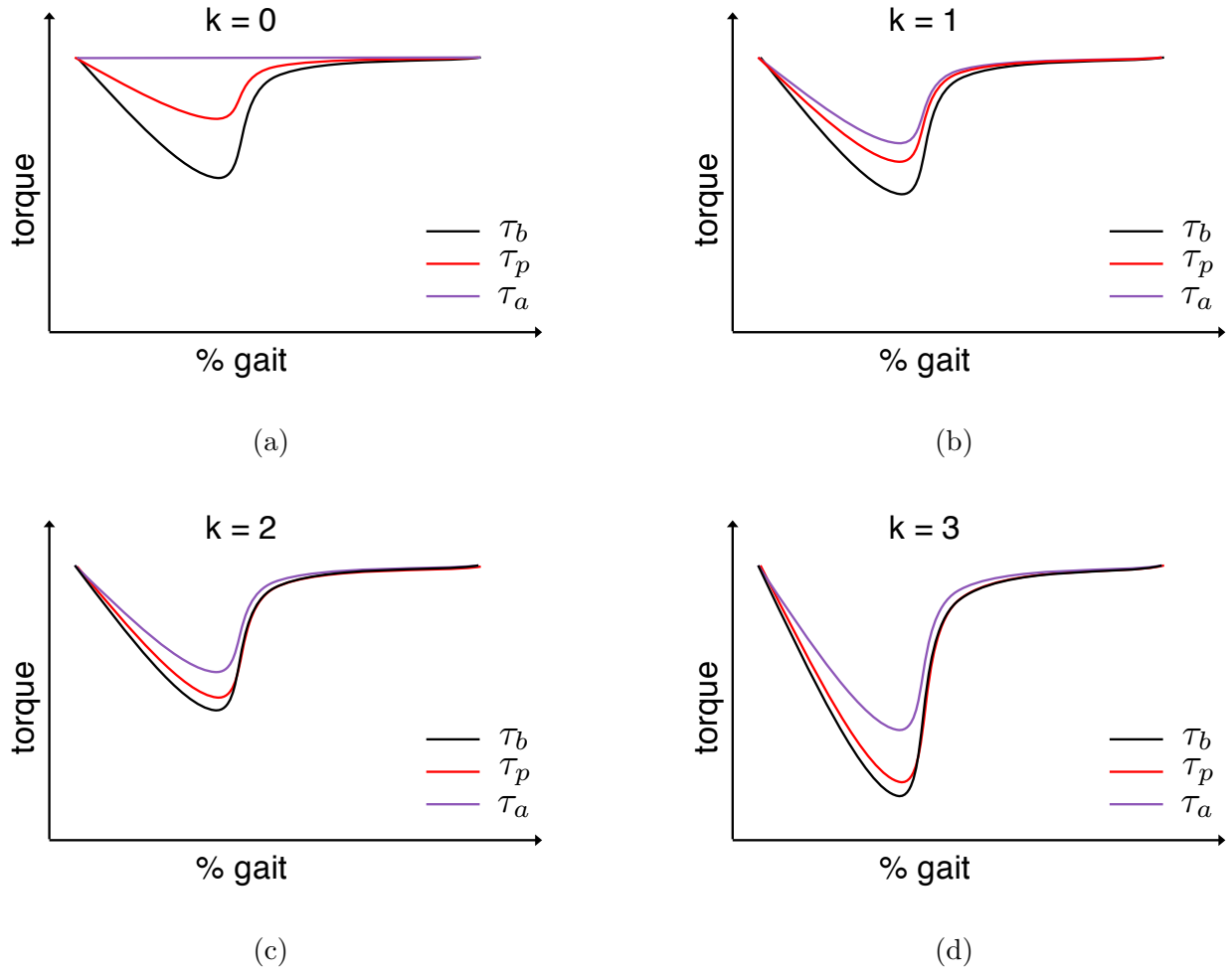


Figure 1.6: Illustrative example of unbounded growth: (a) torque curves during initial iteration $k = 0$, note the active torque $\tau_a = 0$; (b) torque curves after active torque τ_a has been added, note the decrease in the torque difference $\tau_b - \tau_b$, e.g., error; (c) torque curves after more active torque τ_a has been added, note the increase in magnitude of both prosthetic torque τ_p and biological torque τ_b; (d) torque curves after more active torque τ_a has been added, note again the increase in the magnitude of both prosthetic torque τ_p and biological torque τ_b.

overview of passive prosthesis, below-amputee gait, secondary medical conditions, powered prosthesis, and iterative learning control. In Ch. 3, a cam-based nonlinear spring is intro-

duced as a way to customize the passive response of a PAFP. It is shown that the cam-based approach can substantially decrease the active requirements when compared to a direct drive and linear spring approach. Next, an overview of the prototype design is presented, including the powered drive train. The symmetry learning controller, which utilizes an adaptive iterative learning control algorithm, is presented in Ch. 4. The algorithm is designed to bound the control signal growth, even when the reference signal (e.g., the biological ankle torque) grows. System integration, including the controller architecture and embedded implementation, which allows the prototype device and the symmetry controller to provide the desired torque signals, is presented in Ch. 5. Chapter 6 details the experimental evaluations, including measured outcomes and results. A summary of contributions and conclusions are given in Ch. 7.

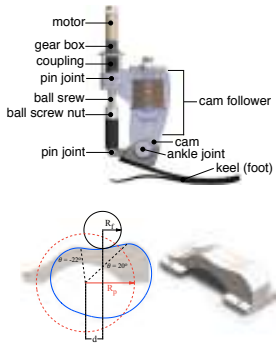
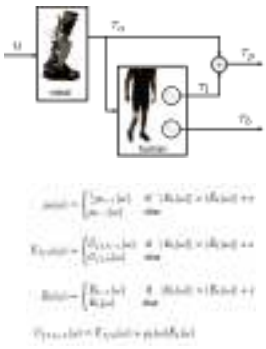
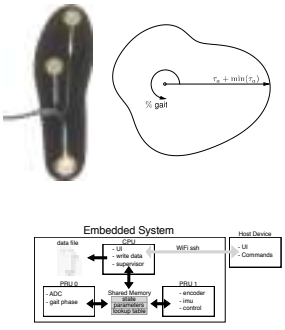
personalized passive torque	avoiding divergence	system integration	evaluation and hypothesis testing
			
• cam-based powered ankle-foot prosthesis	• adaptive iterative learning	• heel-strike detection • pattern generator • system architecture	• testing hypothesis • testing interpolation
ASME DSCC (2013) J. Med. Dev. (2015)	IEEE/ASME MESA (2016) J. Intell. & Rob. Sys. (2017)	ASME DMD (2017)	Trans. Neur/Rehab (prep)

Figure 1.7: Research path.

Chapter 2

BACKGROUND

2.1 Ankle mechanics

Human gait mechanics has been extensively studied, in terms of kinematics, kinetics and energetics, e.g., see [3, 44–49]. The gait cycle begins with heel strike (HS) of one leg and ends with the next heel strike of same leg. Foot flat (FF) occurs after HS when the whole foot makes contact with the ground. Foot flat (FF) marks a transition from plantarflexion to dorsiflexion, where dorsiflexion is defined as the ankle motion that brings the toes closer to the shin, and plantarflexion is defined as ankle motion in the opposite direction. The following sign conventions are used throughout this work: ankle displacement is considered

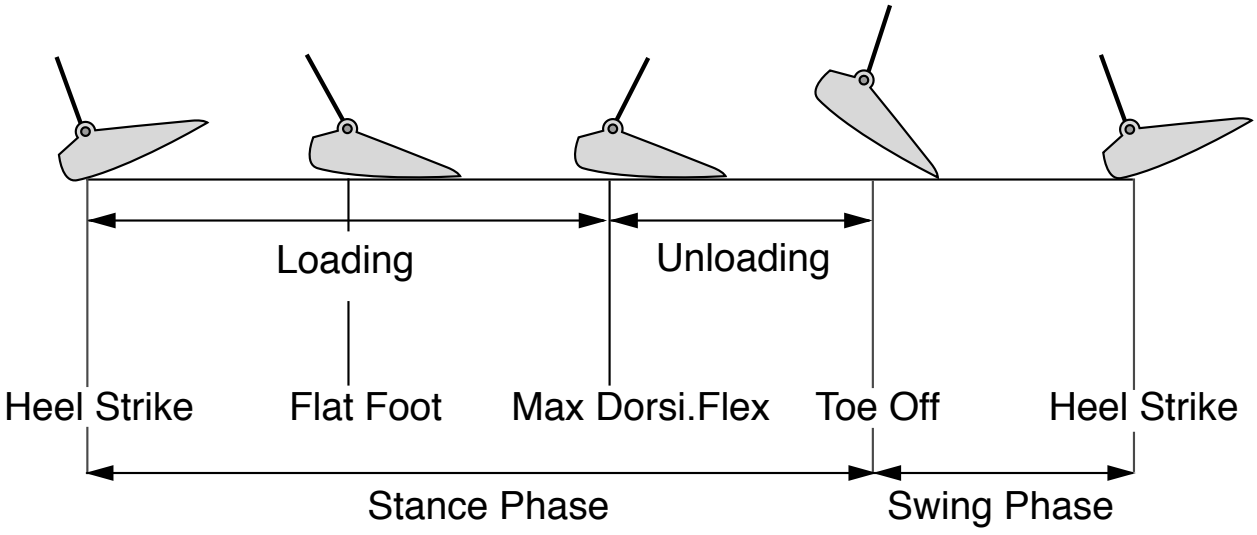


Figure 2.1: Biomechanics of level ground walking. The gait cycle can be divided into a stance phase, and a swing phase.

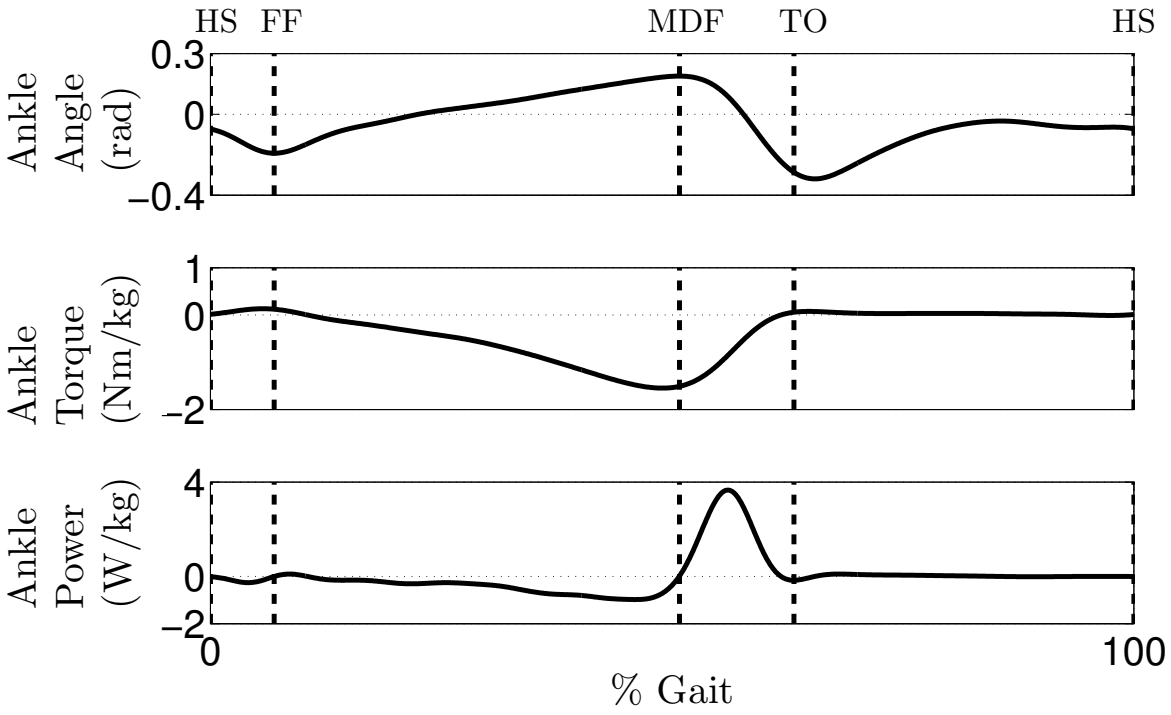


Figure 2.2: Gait trajectories for non-amputee walking at self-selected walking speed [50]. The dotted lines represent the phase transitions from Fig. 2.1. Top: ankle angle during the gait cycle. Middle: Ankle torque normalized by body mass. Bottom: Ankle power normalized by body mass

positive for dorsiflexion, and a torque promoting dorsiflexion is considered to be positive. Zero ankle position is assigned to the position where the foot segment is perpendicular to the shank (or shin) segment. The next gait event is maximum dorsiflexion (MDF), which marks a transition from dorsiflexion to plantarflexion. The last event, toe-off (TO), is defined as the instant when the toe leaves the ground. The stance phase, about 60% of the gait cycle, begins with heel strike (HS) and ends with toe-off (TO). The remaining 40% of the gait cycle is called swing phase, and is characterized by the foot being off the ground. In this article, the stance phase is separated into a period of loading followed by a period of unloading. Loading occurs from HS to to MDF, and unloading occurs from MDF to toe-off TO. Figure 2.1 summarizes the events and phases of level ground gait. The plantarflexor muscles serve two

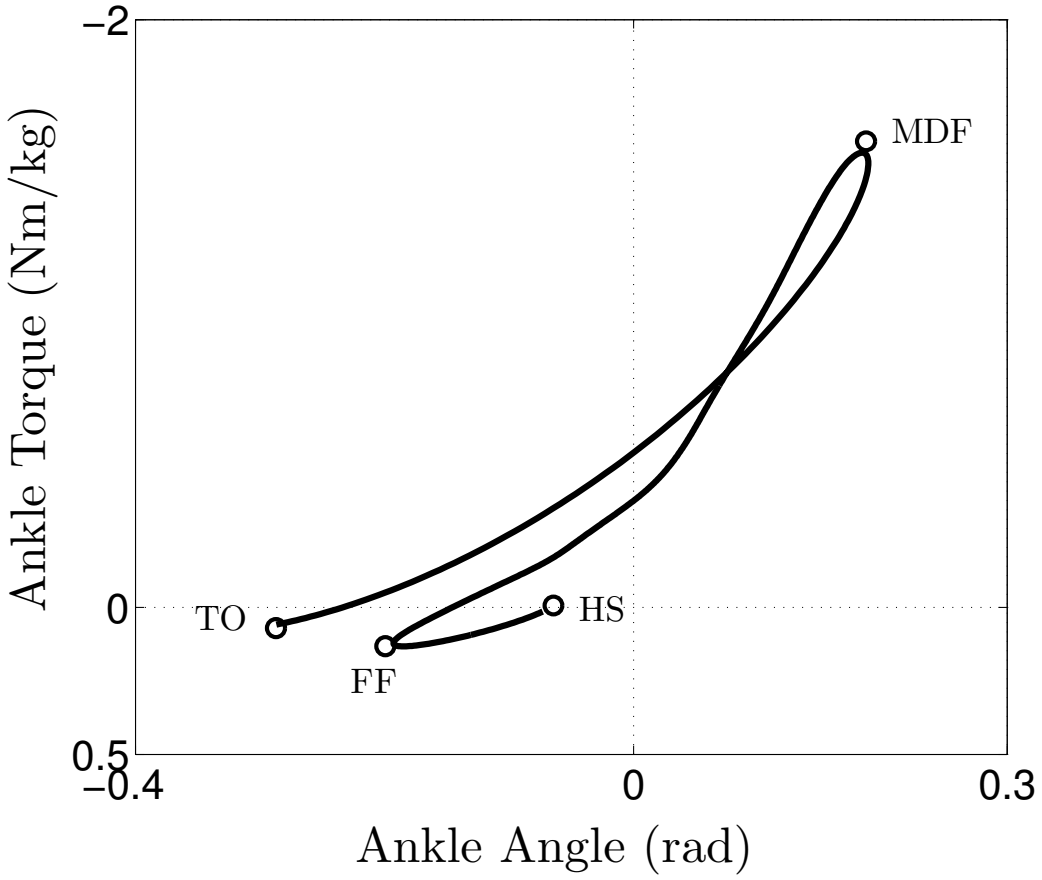


Figure 2.3: Ankle angle-torque characteristics (ankle torque normalized by body mass versus ankle angle) during stance phase for self-selected walking speed. The loading phase begins with HS and ends with MDF. Unloading begins with MDF and ends with TO.

major functions in normal gait: early in stance they assist in controlling the forward rotation of the leg over the supporting foot; and at push-off they are responsible for over 80% of the mechanical power generated during the gait cycle [3]. Gait trajectories for a non-amputee walking at self-selected speed can be seen in Fig. 2.2. The loading period can clearly be seen as the negative part of the power curve (middle plot), followed by the brief unloading period where the power curve is positive and maximal just before TO. The ankle angle-torque

characteristics for the same gait trajectories can be seen in Fig. 2.3. The main challenges in the design of PAFPs is to replicate the (nonlinear) ankle ankle-torque characteristics of the human ankle [49].

2.2 Conventional, energy storing and quasi-passive prostheses

Nowadays, below-knee amputees typically don either energy-storing-and-releasing (ESR) or non-energy storing, e.g., conventional, foot prosthesis. The most common conventional prosthesis, the solid ankle cushioned heel (SACH) foot, is composed of a rigid foot section with a compressible heel wedge which provides minimal flexion. More advance conventional prosthesis, such as the Single-Axis foot, utilize a sagittal joint with two adjustable bumpers that provide increased loading response. Although conventional prosthetics can restore some ankle stiffness characteristics to the users, the peak torque delivered is only two thirds of what is achieved during intact walking [25].

More advanced ESR foot prosthesis, also known as dynamic elastic response prosthesis, are capable of storing energy during stance and returning it to the individual to assist in forward propulsion in late stance. Such ESR prosthesis have sophisticated designs and use advanced composite materials. Various studies have compared conventional to ESR foot prosthesis [51–53]. One study reports almost 70% of user's felt their gait had improved with an ESR foot when compared with a conventional foot [52]. However, the range of motion of the human ankle is larger than what is achievable with a passive prostheses and human ankle stiffness varies within the gait cycle and with walking speed [49], while passive devices have fixed stiffness characteristics.

Quasi-passive devices use actuation to increase controllability of the elastic element, but do not provided actuation of the joint. The Proprio-Foot (Ossur) is a commercial quasi-passive foot that can adjust the neutral position of the ankle, enabling increased mobility on ramps [54]. Collins et al. designed a prosthetic device that can capture and store collision energy and recycle this energy just prior to toe off [55], leading to restoration of push-off energy of the prosthetic limb [23, 56]. However, when compared with a patient's prescribed

prosthesis, a net increase in user's metabolic cost was observed; the researchers theorized this may be due to the collision-loading characteristics of the device causing extra muscular activity needed for stability during the loading phase [56].

None of the device methodologies outlined above can completely replicate the function the human ankle. As a consequence, below-knee amputees' gaits deviate largely from the normal ankle mechanics, described in Sect. 2.1.

2.3 Below-knee amputee gait

The gait of below-knee amputees has been extensively studied [4–7], and the following common deviations from normal gait have been observed:

- Increased hip extension, knee flexion, and ankle dorsiflexion of the intact side, and increased hip and knee flexion of the prosthetic side [6].
- Increased intact total knee work [16].
- Reduced stance phase on the prosthetic side [8–10].
- Loading of the prosthetic limb less that the intact limb [11–14].
- Increased rate of loading, magnitude of loading, impulse, and time of loading of the intact limb, compared with unaffected gait [15].
- Reduced center of mass work from prosthetic-side push-off, increased intact-side collision work, increased work overall, across walking speeds and increased initial peaks in vertical ground reaction forces on the intact side [7].
- Increased energy expenditure [17, 19].

Note with the exception of increased energy expenditure, the deviations from normal gait can be described as asymmetries. Adamczyk et al. theorize that the weak prosthetic side push-off leads to a greater loss of energy during collision of the intact limb with the ground,

and to compensate intact side positive hip work must be performed [7]. This increased collisions and mechanical work may increase the risk of secondary conditions [1, 23].

2.4 Secondary medical conditions

2.4.1 Conditions and prevalence

Lower-limb amputees have increased rates of musculoskeletal problems, such as OA in the knee and hip, and chronic backache [2], which can be additively debilitating. A chief complaint among amputees is knee pain in the intact limb, and to a lesser degree, hip pain [57]. However, below-knee amputees were five times less likely to develop knee pain in their amputated limb, when compared with the general population [58]. Multiple studies have shown an increase in the prevalence of knee and hip OA of the intact limbs joints in amputees compared with the general population [59–61]. Burke et al. further reports osteoporosis in the amputated limb, backache, and high rates of scoliosis (64%), which the author attributed to mechanical factors [61]. Such secondary conditions are thought to be a result of increased loading of the intact limb and altered body mechanics [1, 2].

2.4.2 Mechanisms

There is strong evidence that mechanical loading of the knee and joints, in particular the knee adduction moment (KAM) and the hip adduction moment (HAM), are associated with OA severity and progression in the general population [62, 63]. Only recently have KAMs been studied in amputees. One study demonstrated increased asymmetry in peak KAM in below-knee amputees compared with controls [64]. Royer et al. report that the intact limb of below-knee amputees have approximately 65% larger peak adduction moments at the knee and hip compared with the prosthetic limb, and intact limb adduction moments are larger than the general population [65]. These observations support the hypothesis that knee and hip OA in the amputee population is a direct result of altered mechanics, e.g., asymmetries resulting from compensatory behaviors addressing the loss of ankle function. PAFPs can

potentially replicate human ankle behavior, thus, providing amputees with a substantial benefit over passive devices, and, hopefully, reducing the occurrence of secondary conditions.

2.5 Powered prosthesis

2.5.1 Reducing active requirements

Passive elements parallel to the actuator can reduce the maximum ankle-torque requirement, and thereby, reduce the actuator size in powered lower-limb prostheses, e.g. [66]. By reducing the size of the actuator, used to supply the active ankle torque required for fuller restoration of lower-limb function [23, 49, 67–69], the size and weight of the powered prostheses can be kept similar to that of the missing limb. In turn, the actuator size can be reduced by decreasing the maximum required actuator torque, e.g., as shown for electric motors in [43]. The challenge, however, is to achieve most of the typically nonlinear ankle response with the passive element so that the required active ankle-torque from the actuator can be small.

There is substantial research on the design of passive elastic devices to achieve a specified nonlinear response, e.g., see review article [70]. One approach is to tailor the geometry (splined cantilevers or conical springs) and material properties such that the elastic properties vary as the spring displaces, e.g., [71]. In lower-limb prostheses, on-off type nonlinearities have been introduced through unidirectional elastic elements [30], and clutch or locking mechanisms [32, 72–74]. Mechanisms such as four-bar linkages [66] and hypocycloids [75] can be used to exactly match several points on the elastic (force-displacement) response by selecting a finite number of design parameters such as the link-lengths of the four-bar mechanism or radii of the circles generating the hypocycloids. In addition, these parameters can be used to achieve the best fit possible of the desired elastic response. In comparison to other mechanisms, cam-based designs have more design flexibility (i.e., in the choice of the cam profile) and can, therefore, be used to match a substantially larger segment of the force-displacement curve, e.g., as in [76–78]. In addition, a cam could be made modular, and therefore can be come a customization of a general prostheses structure.

The aim is to match the nonlinear passive response of an ankle for a specific self-selected walking gait. Variations in the ankle response, due to different walking conditions, would then be accommodated by the actuator. If a wide range of gaits are expected, one approach is optimizing to best fit the ankle response over a range of walking conditions [34]. Alternatively, variable stiffness elements, which use motors to change the spring preload or adjust the transmission ratio between actuator and spring, e.g., [70], can be used to adapt for different gaits. Nevertheless, since the majority of prosthesis usage is for self-selected walking, this work focusses on optimizing the passive design for a single gait as in previous works, e.g., [66].

2.5.2 Control methodology

The objective of the controller is to coordinate the action of the powered ankle-foot prosthesis. Many control approaches have emerged [79], however, no single approach has been universally accepted by the community.

A common approach is to use state-based controllers, with intrinsic sensing to detect state transitions (e.g., early stance, swing), where each state alters the ankle mechanics (e.g., impedance) of the prosthesis with parameters derived from average able-bodied human ankle mechanics [31, 33, 35, 37]. Additionally, state parameters can be initialized from nominal (able-bodied) parameters, then tuned based on qualitative user feedback [35, 37]. Another approach is to encode the motor command signal, derived from able-bodied data, as a time-based function that initiates on HS [26]. A reflex controller approach, based on a neuromuscular model fitted to match the human ankle torque-angle profile of a weight and height-matched intact subject, can provide some adaptation, e.g., to speed and slope changes [39]. The major limitations in these approaches are: (i) able-bodied trajectories don't account for personalized altered body mechanics of amputee gait, (ii) the need for qualitative tuning of subject specific control parameters, which is a limitation that may exclude these methods from future clinical application, and (iii) an inability to automatically adapt to the user, especially in the long-term as the user's ability either degrades or improves. Proportional myoelectric control could increase the adaptability of the prosthesis by allowing for

volitional control, however, signal processing and sensor design challenges must be overcome before such methods become reliably implemented on portable devices [80].

Human-in-the-loop methods, where the control signal is automatically tuned and continuously adapts to the user, could overcome the limitations outlined above. Indeed, such methods have proved to be successful at minimizing walking economy with ankle exoskeletons [81,82]. Additionally, these methods have provided insights into the need for personalization, evident from the large variations in optimized assistance patterns found across individuals [82]. However, in the case of human-in-the-loop methods for PAFPs, identifying a measurable user-related metric, e.g., comfort or satisfaction is not as straightforward. The relationship between activity limitations and the metabolic cost of gait in the amputee population remains unclear [42], thus walking economy may not be an ideal candidate for optimization. Alternatively, in this work, ankle torque asymmetry is proposed as a candidate for optimization. The rational is two fold: (i) as described above, asymmetries are functionally limiting, therefore, in theory, asymmetry may provide a proxy measure of comfort, and (ii), asymmetries could potentially be accurately estimated online, e.g., with low-cost embedded insole sensors. The challenge, as with any in human-in-the-loop approach, is to avoid divergent responses cause by time-varying human dynamics.

2.6 Iterative learning control

The proposed symmetry control method utilizes an adaptive iterative learning control (ILC) algorithm [83]. ILC methods are a class of algorithms for synthesizing feed-forward control inputs for processes that are repetitive. Such methods originated in robotics applications, starting with early works, e.g., [84], along with the use of an inverse model of the robot dynamics in early works [85]. These approaches converge to the desired output if the modeling error is small. Improvements of the model through parameter adaptation with data acquired during the iteration was studied in [86] for robotics application using a discrete time implementation. In contrast, this work adapts the ILC methodology to learn a symmetrical torque signal at the prosthetic limb that matches that of the intact limb. Thus, the desired output

is not fixed, as in previous industrial robotic applications, but varies as the prosthetic torque changes. Consequently, since personalized human-dynamics dominate the response of the intact limb, the main goal is to avoid divergence, as no human model is proposed. Note that model-inversion techniques could be possible, e.g., recent work on personalized human-robot models for collaborative tasks could be adapted for prosthesis applications [87]. However, such extensions are beyond the scope of this thesis.

Chapter 3

REDUCING ACTIVE TORQUE THROUGH PERSONALIZATION

3.1 Introduction

The ability to walk comfortably in an urban environment received the highest rankings among lower limb amputees in a study examining prosthesis-related issues of importance [88]. Therefore, the current research aims to match the nonlinear passive response of an ankle for a specific self-selected walking gait. Variations in the ankle response, due to different walking conditions, would then be accommodated by the actuator. If a wide range of gaits are expected, then the current approach could be extended by optimizing the cam profile to best fit the ankle response over a range of walking conditions [34]. Alternatively, variable stiffness elements, which use motors to change the spring preload or adjust the transmission ratio between actuator and spring, e.g., [70], can be used to adapt for different gaits. Nevertheless, since the majority of prosthesis usage is for self-selected walking, this article focusses on optimizing the passive cam-design for a single gait as in previous works, e.g., [66]. For the selected gait (data from [50]) the cam-based nonlinear design can: (i) reduce the overall actuator torque requirement substantially, by $\sim 74\%$; (ii) yield better performance than a linear design, e.g., reduce peak actuator torque by $\sim 21\%$ when compared with a linear design. Passive bench test results are presented to experimentally demonstrate that the current cam-based prototype can achieve the desired nonlinear response to within 10%.

3.2 The design problem

3.2.1 Desired gait trajectory

The desired (reference) gait trajectory used in this study is for level ground walking at self-selected speed (the data is from [50]). Table 3.1 gives the details of the subject. The associated desired angular displacement θ_d, torque τ_d and power P_d time-trajectories are shown in Fig. 2.2. The corresponding ankle stiffness characteristics of the stance phase, or equivalently the torque-displacement relationship, is shown in Fig. 2.3. This trajectory has a small hysteresis loop, and is consistent with the typical gait at self-selected walking speed [49]. In the following, the powered ankle prosthesis will follow these desired angular displacement θ_d and torque τ_d trajectories.

Table 3.1: Subject gait parameters for reference trajectory [50].

Mass (kg)	Height (m)	Mean Leg Length (m)	Age (yrs)	Gait Speed (m/s)
84.4	1.82	0.97	27	1.21

3.2.2 Design issues

The design problem is to effectively mimic the ankle stiffness characteristics shown in Fig. 3.2 over different parts of the stance phase. Moreover, negative power represents ankle power absorption, and positive power corresponds to ankle power generation [89]. Therefore, an ideal device should maximize absorption of energy by passive components during the loading phase of stance (from heel strike (HS) to maximum dorsiflexion (MDF)) and release energy during the unloading phase of stance.

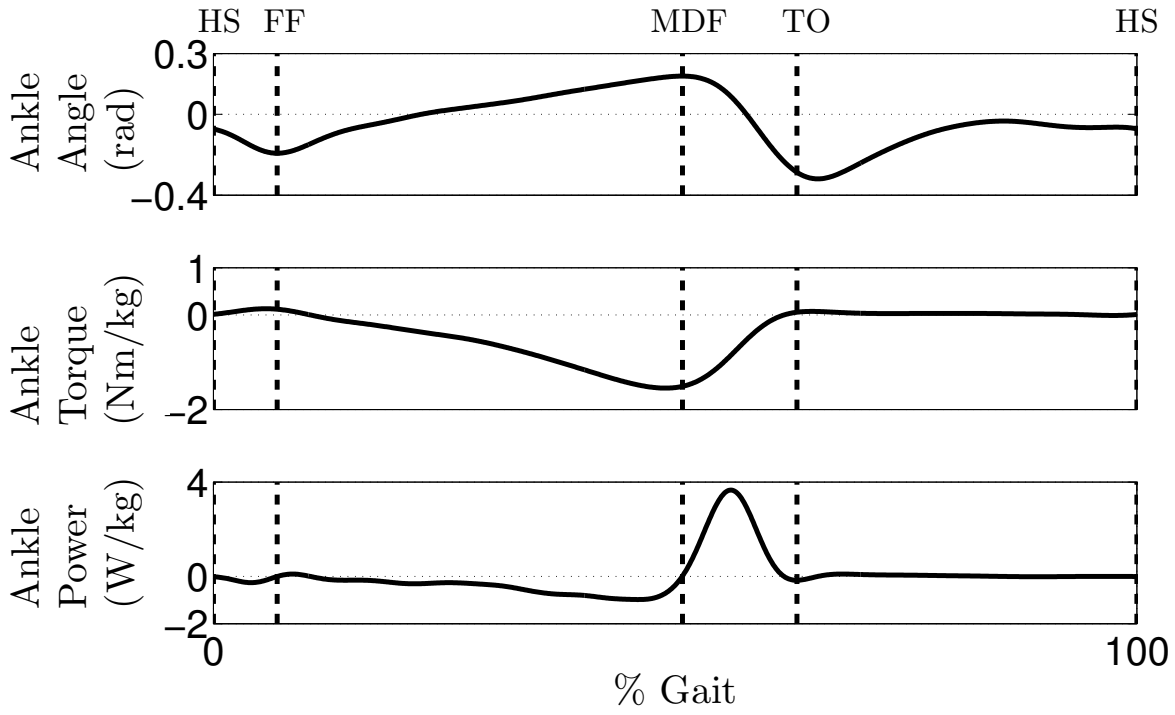


Figure 3.1: Gait trajectories for non-amputee walking at self-selected walking speed [50]. The dotted lines represent the phase transitions from Fig. 2.1. Top: Angular displacement θ_d during the gait cycle. Middle: Ankle torque τ_d normalized by body mass. Bottom: Ankle power P_d normalized by body mass.

3.3 Modeling and optimization

Two configurations for the elastic elements in powered ankle prostheses are: parallel elastic actuator (PEA) [30, 34, 90–92] and series elastic actuator (SEA) [26, 30, 93], as illustrated in Fig. 3.3. A series elastic element (spring k_s) can increase shock tolerance and decrease reflected inertia [94], but cannot reduce torque requirement because the actuator torque τ_a must supply the reference torque τ_d, i.e.,

$$\tau_a = -k_s(\theta_d - \theta_a) = \tau_d, \tag{3.1}$$

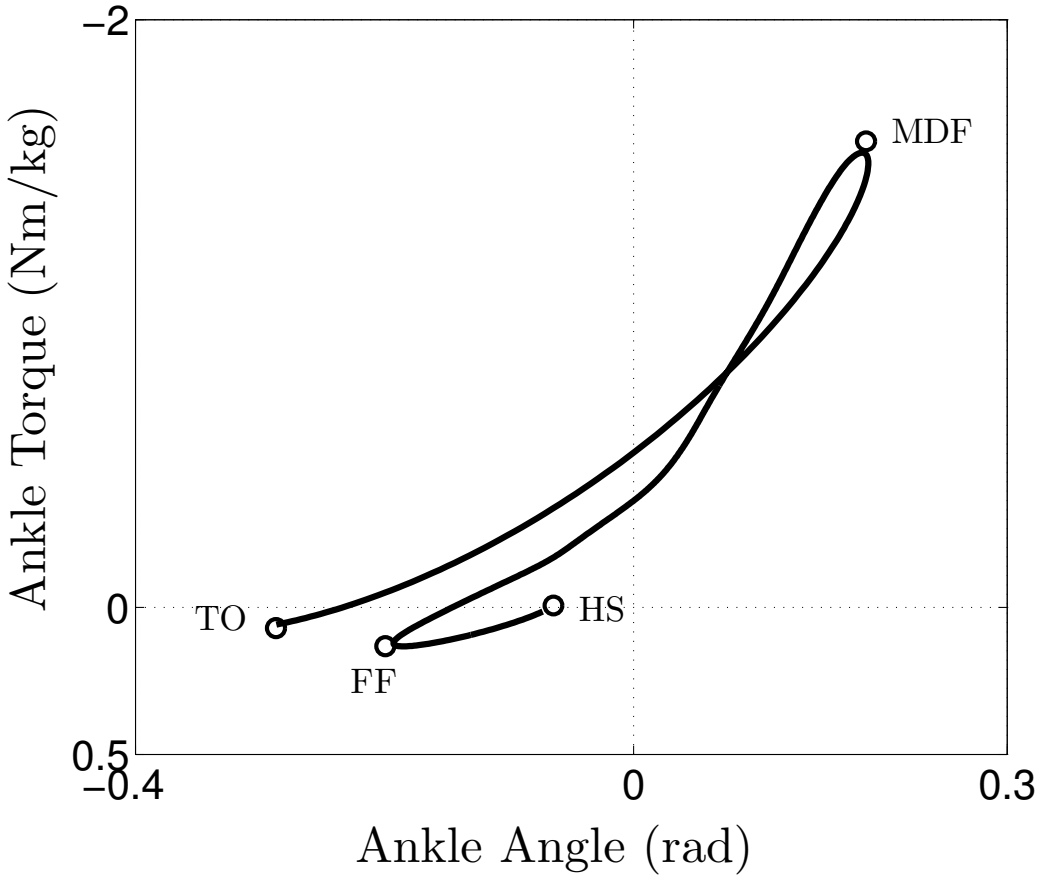


Figure 3.2: Ankle joint stiffness characteristics (ankle torque τ_d normalized by body mass versus angular displacement θ_d) during stance phase for self-selected walking speed. The loading phase begins with HS and ends with MDF. Unloading begins with MDF and ends with TO.

where θ_d is the desired angular displacement, and θ_a is the actuator displacement. Reducing the actuator torque needs a parallel elastic element [30, 34, 90–92]. Hence, this analysis focusses on the optimization of the parallel elastic element.

3.3.1 PEA model

The PEA configuration is illustrated in Fig. 3.3. In order to achieve the desired reference torque τ_d (shown in Fig. 3.1), the actuator must provide the residual active torque τ_a, i.e., the difference between the desired torque τ_d and the torque τ_l produced by the loading of the parallel spring

$$\tau_a = \tau_d - \tau_l. \tag{3.2}$$

For a linear passive spring the torque τ_l cab be expressed in terms the of angular displacement θ_d and spring constant k_p as

$$\tau_l = -k_p\theta_d - k_0, \tag{3.3}$$

where k_0 is the preload of the spring. This expression can be generalized for a nonlinear n^{th} degree polynomial spring to

$$\tau_l = -\sum_{i=0}^{n} \theta_d^i k_{p_i} = -\Theta_n^{\mathrm{T}}\mathbf{k}_p \tag{3.4}$$

where transpose is denoted by superscript T, Θ_n is a vector containing a geometric progression of θ_d up to power n

$$\Theta_n = \begin{bmatrix} 1 & \theta_d^1 & \theta_d^2 & \cdots & \theta_d^n \end{bmatrix}^{\mathrm{T}}, \tag{3.5}$$

and the vector $\mathbf{k}_p$ contains the $n+1$ nonlinear spring parameters:

$$\mathbf{k}_p = \begin{bmatrix} k_{p,0} & k_{p,1} & \cdots & k_{p,n} \end{bmatrix}^{\mathrm{T}}. \tag{3.6}$$

The active power P_a (the active torque τ_a multiplied by the angular velocity of the actuator $\dot{\theta}_a$) can be expressed as

$$P_a = \tau_a\dot{\theta}_a = \tau_d\dot{\theta}_d + \left(\Theta_n^{\mathrm{T}}\mathbf{k}_p\right)\dot{\theta}_d, \tag{3.7}$$

where the actuator displacement θ_a is identically the desired angular displacement θ_d.

3.3.2 Optimization

The goal of the optimization is to choose PEA spring parameter $\mathbf{k}_p$ such that the smallest actuator and energy resources can be used to maintain the reference trajectory. Therefore,

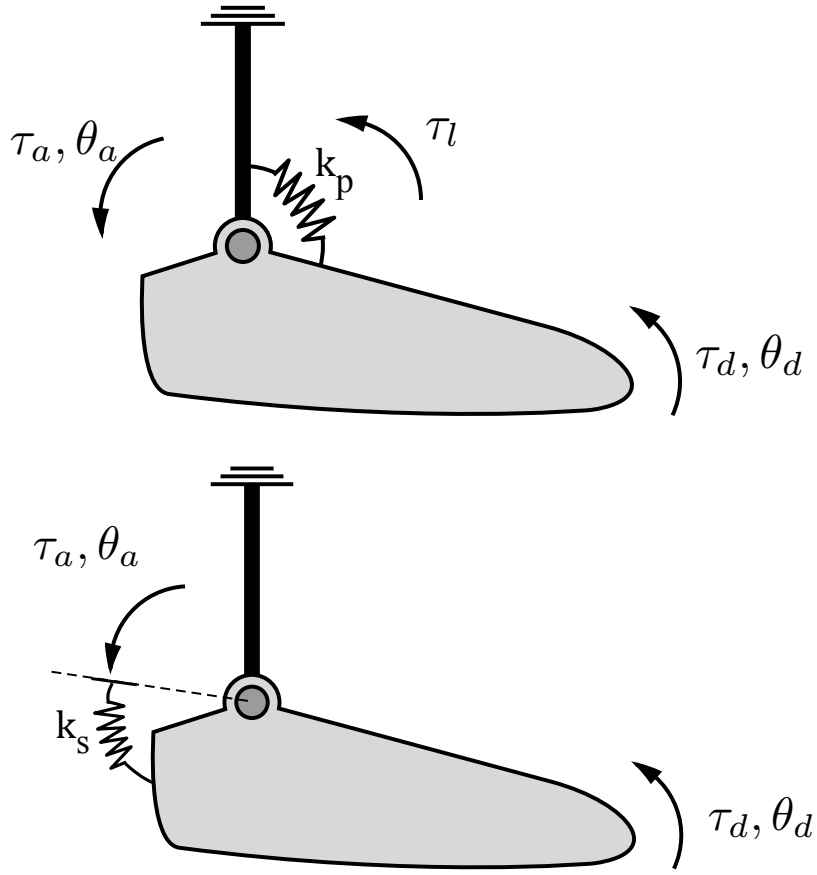


Figure 3.3: Schematic representations of 1 DOF powered ankle configurations. (Top) The PEA configuration is characterized by an ideal torque source τ_a in parallel with a rotational passive component. The parallel component generates a torque τ_l as a function of the angular displacement θ_d. (Bottom) The SEA configuration consists of an ideal torque source τ_a in series with a rotational passive component.

the optimization seeks to minimize the peak torque $\tau_{a,p}$ and positive peak power $P^+_{a,p}$ over the entire gait cycle. It is noted that the negative mechanical power is not penalized in the optimization, as it can potentially be stored and reused as well, e.g., by converting to electrical energy [30]. Additionally, a penalty on the positive mechanical energy E^+_a, used

during a gait cycle, is added to yield the following minimization problem.

$$\min_{\mathbf{k}_p} \quad w_1 E_a^+ + w_2 P_{a,p}^+ + w_3 \tau_{a,p} \tag{3.8}$$

where $\mathbf{k_p}$ are the spring parameters to be optimized, and the weighting parameters can be expressed as a vector $\mathbf{w}$

$$\mathbf{w} = [w_1 \; w_2 \; w_3]. \tag{3.9}$$

In the above minimization problem (in Eq. (3.8)), the peak torque supplied by the active component $\tau_{a,p}$ is defined as the maximum absolute value of the active torque over all sampled values in the entire gait cycle, i.e.,

$$\tau_{a,p} := \max_{i \in \{1,\cdots,p\}} \left| \tau_d[i] + \Theta_n^{\mathrm{T}}[i] \mathbf{k}_p \right|, \tag{3.10}$$

where $[i]$ represents the i^{th} sample, and p is the total number of samples. The power P_a can be decomposed into positive P_a^+ and negative P_a^- parts, where

$$\begin{aligned} P_a^+ &= \frac{1}{2} \left(P_a + |P_a| \right) \\ &= \frac{1}{2} \left(\tau_d \dot{\theta}_d + \left(\Theta_n^{\mathrm{T}} \mathbf{k}_p \right) \dot{\theta}_d + \left| \tau_d \dot{\theta}_d + \left(\Theta_n^{\mathrm{T}} \mathbf{k}_p \right) \dot{\theta}_d \right| \right) \end{aligned} \tag{3.11}$$

and

$$\begin{aligned} P_a^- &= \frac{1}{2} \left(P_a - |P_a| \right) \\ &= \frac{1}{2} \left(\tau_d \dot{\theta}_d + \left(\Theta_n^{\mathrm{T}} \mathbf{k}_p \right) \dot{\theta}_d - \left| \tau_d \dot{\theta}_d + \left(\Theta_n^{\mathrm{T}} \mathbf{k}_p \right) \dot{\theta}_d \right| \right). \end{aligned} \tag{3.12}$$

Then the positive peak power is defined as the maximum over all sampled values in the entire gait cycle, i.e.,

$$P_{a,p}^+ := \max_{i \in \{1,\cdots,p\}} \; P_a^+[i]. \tag{3.13}$$

Positive and negative mechanical energy requirements are defined as the integral of positive and negative power respectively, and are given by

$$E_a^+ = \int_0^T P_a(t)^+ \, dt \approx \sum_{i=1}^{p} P_a[i]^+ \Delta_t \tag{3.14}$$

$$E_a^- = \int_0^T P_a(t)^- \, dt \approx \sum_{i=1}^{p} P_a[i]^- \Delta_t \tag{3.15}$$

where $\Delta_t = T/p$ is the sampling period and the integral is approximated with the sampled values.

The minimization problem in Eq. (3.8) is a convex optimization problem, since the objective function is a summation of convex functions with optimization variable $\mathbf{k}_p$ appearing linearly in the expressions for peak torque (Eq.(3.10)), peak positive power (Eq.(3.13)), and positive energy (Eq.(3.14)). Hence, standard optimization algorithms can be used to solve the optimization problem in Eq. (3.8), and global optimality can be guaranteed [95].

3.3.3 Optimization results

In this section performance of the different configurations are quantified. First, a comparison between a linear and nonlinear PEA reveals that a nonlinear PEA can significantly decrease the torque requirement of the active component. Additionally, the decrease in torque requirement can be traded off to reduce the positive energy requirement and the positive peak power by appropriately choosing the weights in Eq. (3.9).

Torque reduction with nonlinear PEA

Towards reducing the actuator torque, optimal spring parameters were found for PEA_n (i.e., different n in Eq. (3.4)) using the objective function Eq. (3.8) with weighting on only the peak torque (i.e., $\mathbf{w} = [0\ 0\ 1]$), where the polynomial order n of the PEA is denoted with a subscript. The effect of polynomial degree on each component of the objective function, Eq. (3.8), is shown in Fig. 3.4, where each component has been normalized to the common scale $[0, 1]$, e.g.

$$\begin{aligned}
\hat{E}_a^+(n) &= \frac{E_a^+(n)}{\max\limits_{j \in \{1,\ \cdots,\ 6\}} E_a^+(j)}, \\
\hat{P}_{a,p}^+(n) &= \frac{P_{a,p}^+(n)}{\max\limits_{j \in \{1,\ \cdots,\ 6\}} P_{a,p}^+(j)}, \\
\hat{\tau}_{a,p}(n) &= \frac{\tau_{a,p}(n)}{\max\limits_{j \in \{1,\ \cdots,\ 6\}} \tau_{a,p}(j)},
\end{aligned} \tag{3.16}$$

and (n) corresponds to the polynomial degree. There is a large decrease (48.4%) in the peak torque between the linear PEA_1 and second-order nonlinear PEA_2 element. However, performance plateaus as the order n of the nonlinearity is increased; only marginal performance increases are realized with higher orders and the peak torque decrease is 48.8% between the linear PEA_1 and 6^{th} order nonlinear PEA_6.

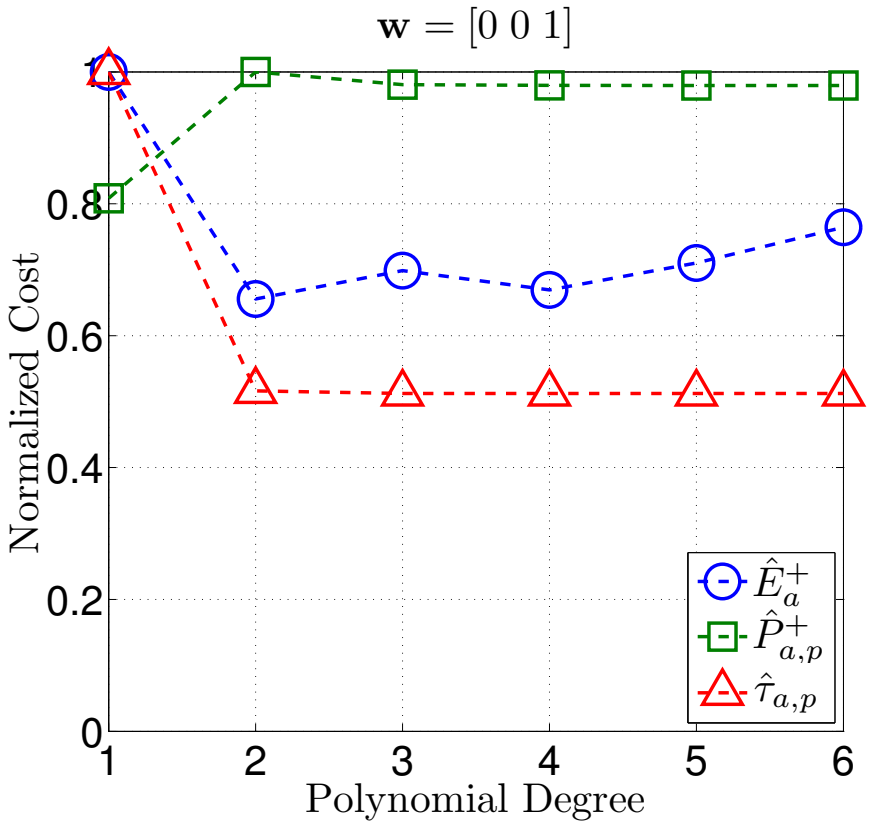


Figure 3.4: Normalized component costs (Eq. 3.16) versus polynomial degree n for optimal PEA. The optimization weights are $\mathbf{w} = [0\ 0\ 1]$ in Eq. (3.9).

Mixed weighting

Optimizing the nonlinear spring to reduce only the peak torque requirement results in a large torque reduction. However, a consequence is an increase in the power requirement, e.g.,

the positive peak power $P^{+}_{a,p}$ increased by 23.7% between the linear PEA_1 and nonlinear PEA_2 as seen in Fig. 3.4. The optimization weights $\mathbf{w}$ in Eq. (3.9) can be chosen to trade-off the reduction in peak torque $\tau_{a,p}$ to decrease the peak positive power $P^{+}_{a,p}$. To illustrate, optimizations were performed for different degree n nonlinear PEA_n with the weighting chosen as $\mathbf{w} = [1\ 1\ 1]$, which penalizes all three: positive energy, peak positive power, and peak torque. The effect of polynomial degree on each component of the objective function, Eq. (3.8), is shown in Fig. 3.5. A reduction in all components of the objective function is realized with increasing polynomial order, but the benefits plateau after polynomial degree $n = 3$, and hence a third-degree nonlinear PEA_3 provides sufficient nonlinearity, and is selected as the nonlinearity to be achieved with a cam device in the following.

To illustrate the advantage of a nonlinear design over a linear design, Fig. 3.6 and Table 3.2 compare the performance of the linear PEA_1 and the nonlinear PEA_3 optimized with two different weights: $\mathbf{w} = [0\ 0\ 1]$ (optimized for torque only) and $\mathbf{w} = [1\ 1\ 1]$ (optimized for energy, power and torque). Note, the PEA_3 with unity weights ($\mathbf{w} = [1\ 1\ 1]$) can reduce the positive energy and positive peak power without substantially giving up the reduction in peak torque as seen in Table 3.2. Moreover, the passive stiffness characteristics (e.g., torque-displacement relationship) of the third-degree nonlinear PEA_3 with $\mathbf{w} = [1\ 1\ 1]$ is closer to the desired ankle properties when compared to the linear PEA_1 as seen in Fig. 3.7. Thus, the closer match results in a reduction of actuator requirements as seen in Fig. 3.8. In particular, the actuator requirements for the third-degree nonlinear PEA_3 with $\mathbf{w} = [1\ 1\ 1]$ are smaller than the linear PEA_1 with $\mathbf{w} = [1\ 1\ 1]$, i.e., are decreased by $\sim 40\%$, $\sim 29\%$, and $\sim 21\%$, for positive energy, peak positive power, and peak torque, respectively, as seen by comparing the third and fifth row in Table 3.2. Moreover,the actuator requirements for the third-degree nonlinear PEA_3 with $\mathbf{w} = [1\ 1\ 1]$ are substantially smaller than the case without the PEA, i.e., are decreased by $\sim 68\%$, $\sim 91\%$, and $\sim 74\%$, for positive energy, peak positive power, and peak torque, respectively, as seen by comparing the first and fifth row in Table 3.2.

The optimal parameters of the third-degree nonlinear PEA_3 with weighting $\mathbf{w} = [1\ 1\ 1]$,

for a person with mass $M = 85$ kg (typical value for a patient in the VA Hospital), are

$$\tau_l^* = -\Theta_3^{\mathrm{T}} \mathbf{k}_p^*, \quad \mathbf{k}_p^* = \begin{bmatrix} 41 & 301 & 617 & 439 \end{bmatrix}^{\mathrm{T}}. \tag{3.17}$$

The design of the cam device to achieve this nonlinearity is discussed next.

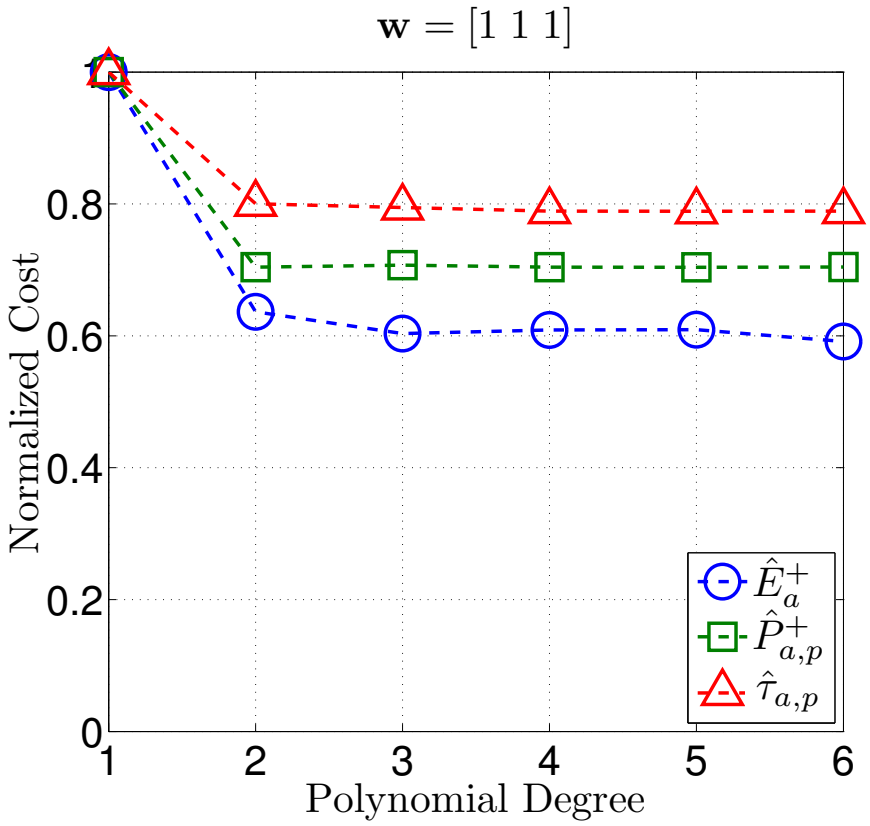


Figure 3.5: Normalized component costs (Eq. 3.16) versus polynomial degree n for optimal PEA. The optimization weights are $\mathbf{w} = [1\ 1\ 1]$ in Eq. (3.9).

3.4 Cam device

The working principle of the cam-follower mechanism is illustrated in Fig. 3.9. This section describes the cam design problem, the selection of the device parameters, and the integration into a powered ankle prosthesis.

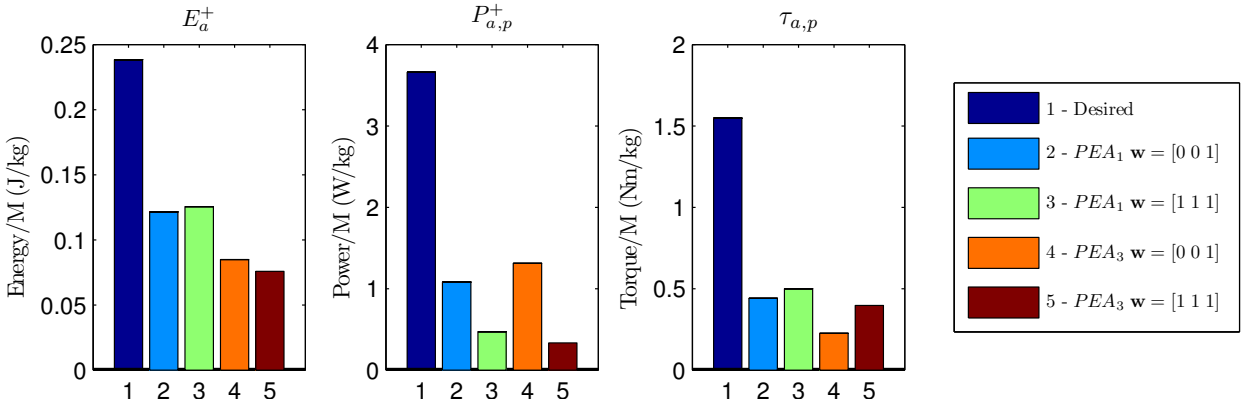


Figure 3.6: Performance comparison of PEAs: positive energy E_a^+ (left); peak positive peak power $P_{a,p}^+$ (middle); and peak torque $\tau_{a,p}$ (right), normalized by body mass of 85 (kg).

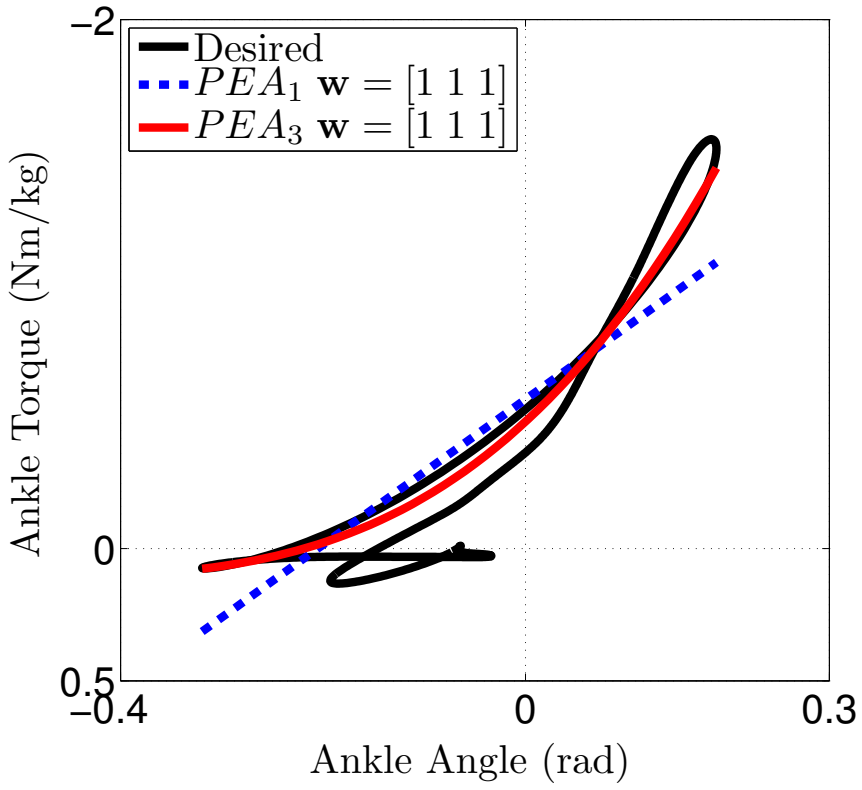


Figure 3.7: The non-amputee ankle stiffness characteristics for the entire gait cycle, superimposed with the linear PEA_1 and third-degree nonlinear PEA_3 stiffness profiles.

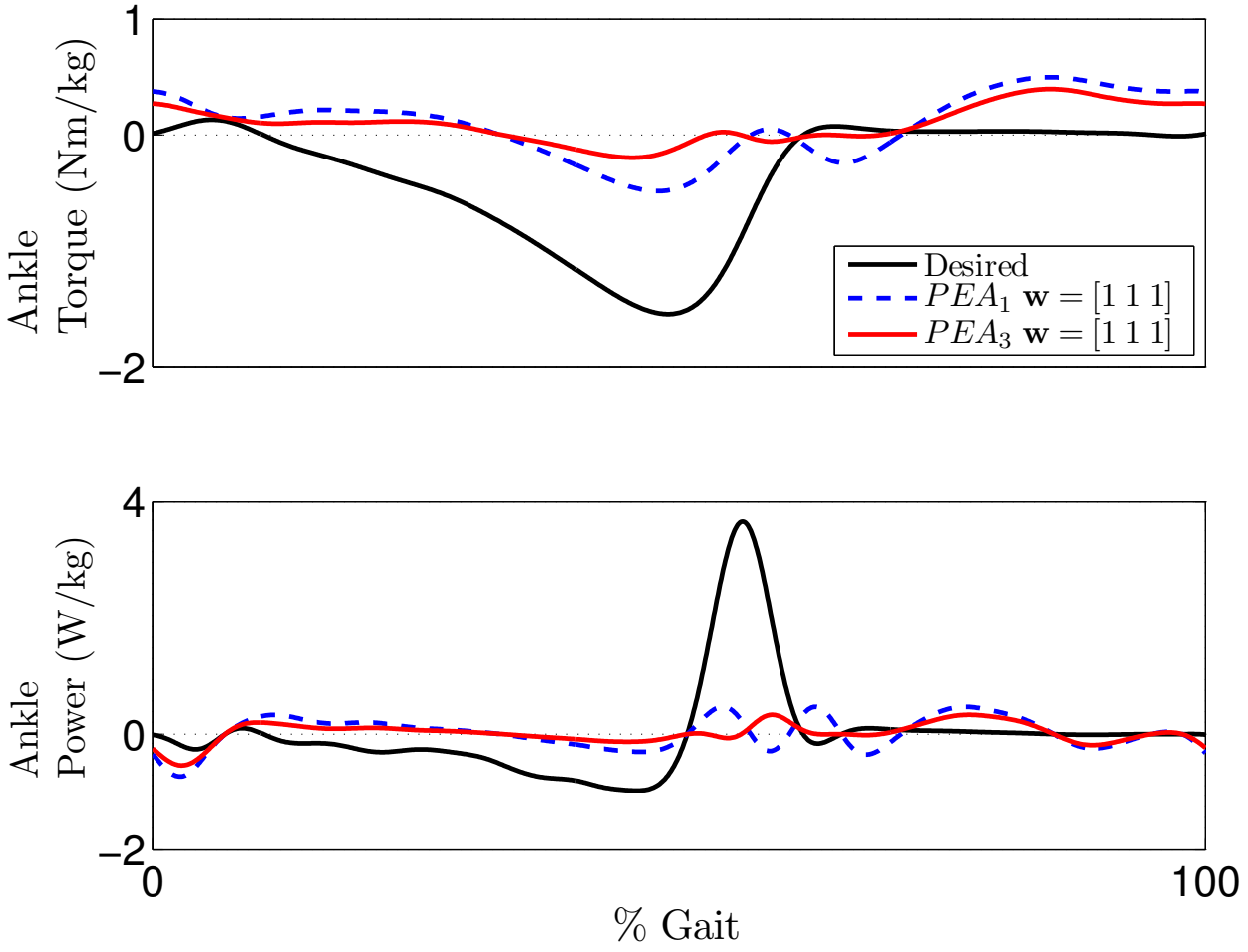


Figure 3.8: Comparison of normalized actuator torque τ_a and power P_a for optimal linear PEA_1 with those for the optimal, third-degree nonlinear PEA_3 with $\mathbf{w} = [1\ 1\ 1]$.

Table 3.2: Comparison of PEA performance in Fig. 3.6, normalized by body mass of 85 (kg).

	E_a^+ (J/kg)	$P_{a,p}^+$ (W/kg)	$\tau_{a,p}$ (Nm/kg)
Desired	0.2384	3.6601	1.5475
PEA_1 $\mathbf{w} = [0\ 0\ 1]$	0.1215	1.0806	0.4404
PEA_1 $\mathbf{w} = [1\ 1\ 1]$	0.1256	0.4719	0.4996
PEA_3 $\mathbf{w} = [0\ 0\ 1]$	0.0849	1.3111	0.2257
PEA_3 $\mathbf{w} = [1\ 1\ 1]$	0.0758	0.3337	0.3970

3.4.1 Cam design problem

The cam shape is affected by the cam-follower displacement s needed to generate a specified torque τ_c, expressed as

$$\tau_c = -N_y b = -(N\cos\phi)b = -(k_c s)b \tag{3.18}$$

where N is the normal force acting on the cam, N_y is the vertical component of the normal force, ϕ is the pressure angle (e.g., the angle between the normal force N and the line of action of the follower), k_c is the spring constant of the follower spring, s is the displacement of the follower (which also corresponds to the change in the follower-spring length) and b is the horizontal distance from the cam axis of rotation (point O in Fig. 3.10) to the instant center of velocity of the cam and the follower (point A in Fig. 3.10), which is a point that moves with the cam and has the same velocity, v_f, as the follower. Note, that the distance b is numerically equal to the rate of change $s' = ds/d\theta_d$ of the follower displacement s [96], because

$$v_f = \frac{ds}{dt} = \left(\frac{d\theta_d}{dt}\right) b, \tag{3.19}$$

which results in

$$s' = \frac{ds}{d\theta_d} = b. \tag{3.20}$$

Hence, the cam torque T_c in Eq. (3.18) maybe rewritten as

$$\tau_c = -k_c s s'. \tag{3.21}$$

The cam torque τ_c should match the optimal nonlinear passive torque τ_l^* (Eq. (3.17)), which leads to

$$-k_c s s' = -\sum_{i=0}^{n} \theta_d^i k_{p_i}^* \tag{3.22}$$

Integrating Eq. (3.22) yields an expression for the follower displacement s as a function of the desired angular displacement θ_d

$$\begin{aligned} -k_c s \frac{ds}{d\theta_d} &= -\sum_{i=0}^{n} \theta_d^i k_{p_i}^* \\ \int s ds &= \int \frac{1}{k_c} \sum_{i=0}^{n} \theta_d^i k_{p_i}^* d\theta_d \\ s &= \left(\frac{2}{k_c} \sum_{i=0}^{n} \frac{1}{i+1} \theta_d^{i+1} k_{p_i}^* + c \right)^{1/2} \qquad (3.23) \\ &= \left[f(\theta_d) + c \right]^{1/2} \qquad (3.24) \end{aligned}$$

where c is a constant and the function f is independent of c.

Large side forces on the cam-roller can lead to jamming. To avoid this, the normal force (N in Fig. 3.10) on the roller should be aligned with the motion v_f of the follower, i.e., the pressure angle ϕ between them must remain sufficiently small (e.g., $\max|\phi| < 30^o$ [96]). The pressure angle ϕ can be expressed as

$$\phi = \tan^{-1} \frac{b-d}{\sqrt{R_p^2 - d^2} + s} = \tan^{-1} \frac{s'-d}{\sqrt{R_p^2 - d^2} + s}, \tag{3.25}$$

where the follower displacement, s, is given by Eq. (3.23), R_p is the prime radius of the cam, and d is the eccentricity of the cam, i.e., the distance from the cam rotation axis, denoted point O, to the line of action of the follower as shown in Fig. 3.10. Additionally, to ensure a single point of contact between the cam and the roller, the minimum radius of curvature ρ of the cam must be sufficiently greater than the radius of the roller R_f (e.g., $\min|\rho| \geq 1.5R_f$ [96]). The radius of curvature ρ of the cam can be expressed as [96]

$$\rho = \frac{\left((R_p + s)^2 + s'^2 \right)^{3/2}}{(R_p + s)^2 + 2s'^2 - s''(R_p + s)}. \tag{3.26}$$

Thus, the two constraints in the cam design are

$$\min |\rho| \geq 1.5R_f, \quad \max |\phi| < 30^o. \tag{3.27}$$

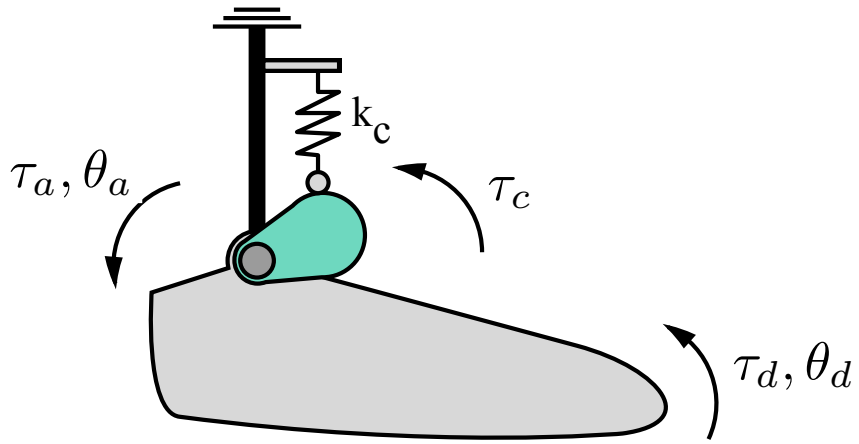


Figure 3.9: Design concept for cam-based parallel component. The cam is fixed to the foot segment and displaces the spring attached to the shank segment.

3.4.2 Selection of cam design parameters

The parameters that can be chosen are the integration constant c in the cam profile s (see Eq. 3.23), the prime radius R_p, the roller radius R_f, the eccentricity d, and the follower spring constant k_c. To keep the cam size small, the prime radius and the roller radius R_f were chosen to be small — the selected prime radius was $R_p = 25$ mm and the roller radius was $R_f = 9.5$ mm (Smith Bearing Yoke Type Metric Cam Follower MYR-6-S).

The two constraints in Eq. (3.27) can be satisfied by choosing a sufficiently large integration constant c in Eq. (3.23). In particular, from Eq. (3.24), the follower displacement s increases with c while the derivatives of s with respect to the angular displacement θ_d

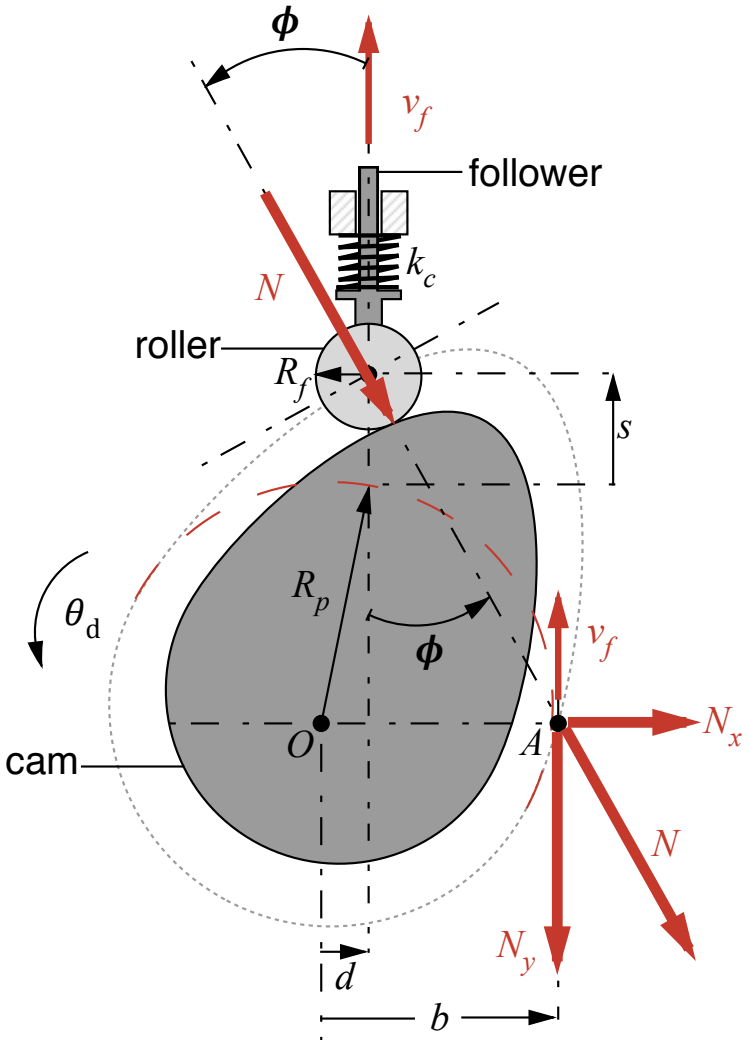


Figure 3.10: Schematic of the cam-follower mechanism adapted from [96]. The cam rotates about point O (the center of the prime radius R_p), which is d away from the line of action of the cam follower, whose displacement is s and velocity is v_f. The pressure angle ϕ is the angle between the normal force N and the follower's line of action. The instant center of rotation is A, which is b away from the center of rotation O.

decrease with increasing c because

$$s' = \frac{ds}{d\theta_d} = \frac{1}{2s} f'(\theta_d) \tag{3.28}$$

$$s'' = \frac{d^2 s}{d\theta_d^2} = -\frac{1}{4s^3} f'(\theta_d)^2 + \frac{1}{2s} f''(\theta_d), \tag{3.29}$$

and f', f'' are independent of c. This implies that as the integration constant c becomes large (alternatively, as s becomes large), the magnitude of the radius of curvature ρ tends to infinity since the numerator dominates in Eq. (3.26), i.e.,

$$\begin{aligned}\rho &= \frac{\left((R_p+s)^2+\frac{1}{4s^2}f'(\theta_d)^2\right)^{3/2}}{(R_p+s)^2+\frac{5s+Rp}{4s^3}f'(\theta_d)^2-\frac{s+R_p}{2s}f''(\theta_d)} \\ &\approx (R_p+s) \quad \text{as} \quad s\to\infty.\end{aligned} \tag{3.30}$$

Similarly, as integration constant c become large, the pressure angle ϕ tends to zero since the denominator dominates in Eq. (3.25). Thus, both constraints can be satisfied by choosing a sufficiently-large, integration constant c in Eq. (3.23).

The selected cam parameters (in Table 3.3) satisfy the constraints. For example, The effect of varying the eccentricity d and spring constant k_c on the pressure angle can be seen in Fig. 3.11, where isolines represent the maximum pressure angle ϕ (Eq. (3.25)), and the black dotted line denotes the maximum allowable pressure angle, $\max|\phi| = 30^o$. The selected cam parameters (denoted by the red dot in Fig. 3.11 for an off-the-shelf spring $k_c = 817$ kN/m, Anchor Lamina JIS Compression Spring 95–5060), satisfies the pressure angle constraint in Eq. (3.27). Also, the constraint on the radius of curvature ρ in Eq. (3.27) is satisfied by choosing the integration constant c to yield a sufficiently-large minimum radius of curvature $\min|\rho| = 16$ mm that is more than 1.5 times the roller radius $R_f = 9.5$ mm. The resulting cam shape is shown in Figure 3.12. The cam-based spring was manufactured and it's stiffness profile tested, which matched the desired nonlinear response to within 10% (see Appendix A.1).

3.5 Device integration

The cam can be exchanged (if needed) since it is attached to the ankle link by four screws. Moreover, a specially designed ramp section is incorporated into the cam shape as shown in Fig. 3.12, which allows the spring to be unloaded by sufficiently rotating the ankle in the plantarflex direction. The main components of the device (including those in the follower

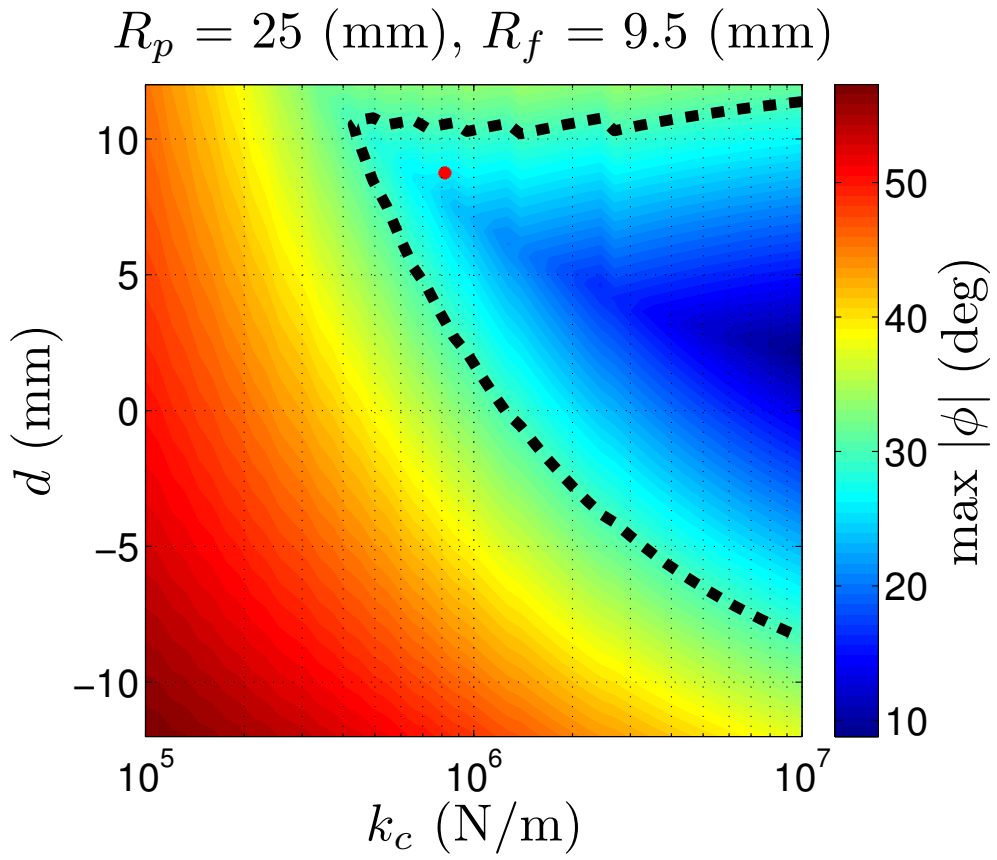


Figure 3.11: Pressure angle ϕ for different follower stiffness k_c and eccentricity d. Black dotted line denotes maximum allowable pressure angle, $\max|\phi| = 30^o$. The red dot denotes the prototype configuration.

assembly) are shown in Fig. 3.13, and an image of the actual device is provided in Fig. 3.14.

3.5.1 Drive Train Model

The powered drive train, a motorized link acting across the shank and ankle links (see Fig. 3.13), provides the active torque τ_a, defnined in about the ankle joint. The drive train consists of the following: a brushless DC motor (Maxon EC-22, 100 W, 24 V), attached to a pin joint located near the top of the shank link (pin joint A), in series with a planetary gearhead with a transmission ratio of $R_g = 19$ (Maxon GP 22 HP), followed by $\ell = 4$ (mm)

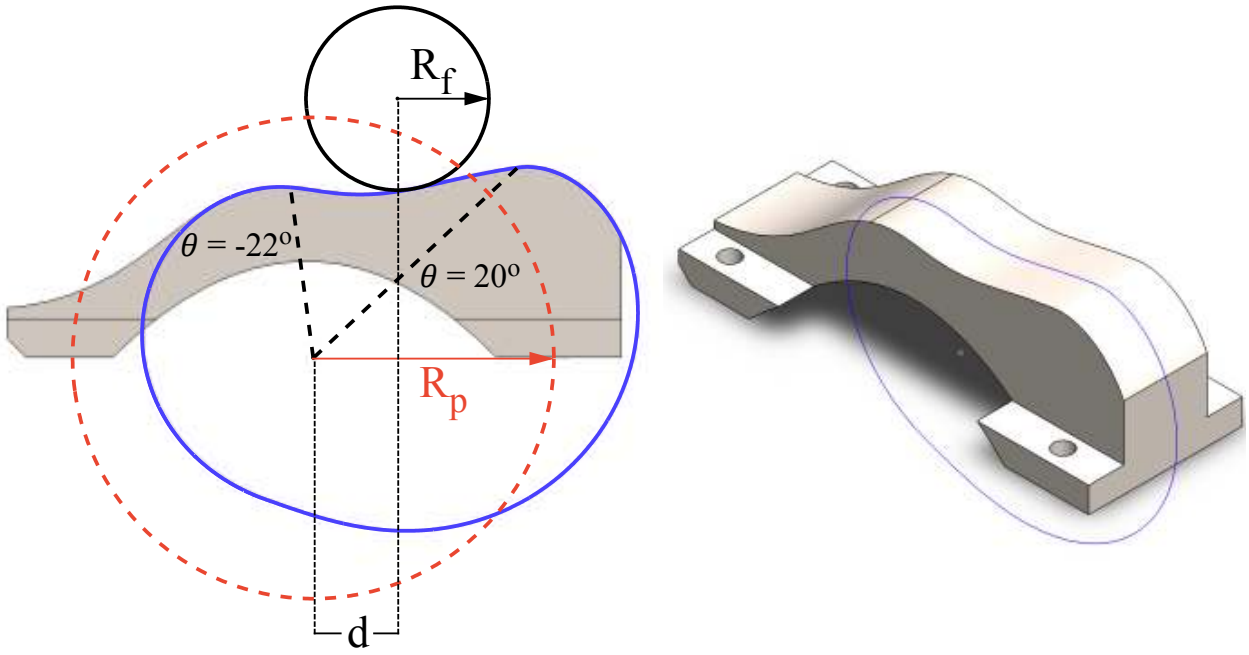


Figure 3.12: Cam prototype for PEA_3 with parameters in Table 3.3. The black dashed lines represent the boundaries of the functional area: -22^o for plantarflexion and 20^o for dorsiflexion.

Table 3.3: Prototype cam design parameters.

Prime Radius R_p (mm)	25
Roller Radius R_f (mm)	9.5
Follower Stiffness k_c (kN/m)	817
Eccentricity d (mm)	8.75
Constant c (m^2)	1.25e-5
Max Pressure Angle (deg)	26
Min Cam Curvature (mm)	16

pitch linear ball screw (Thompson NEFF Rolled Ball Screw) attached to a pin joint on the ankle link (pin joint B) that acts with a moment arm $d = 6$ (mm) from the ankle joint (ankle

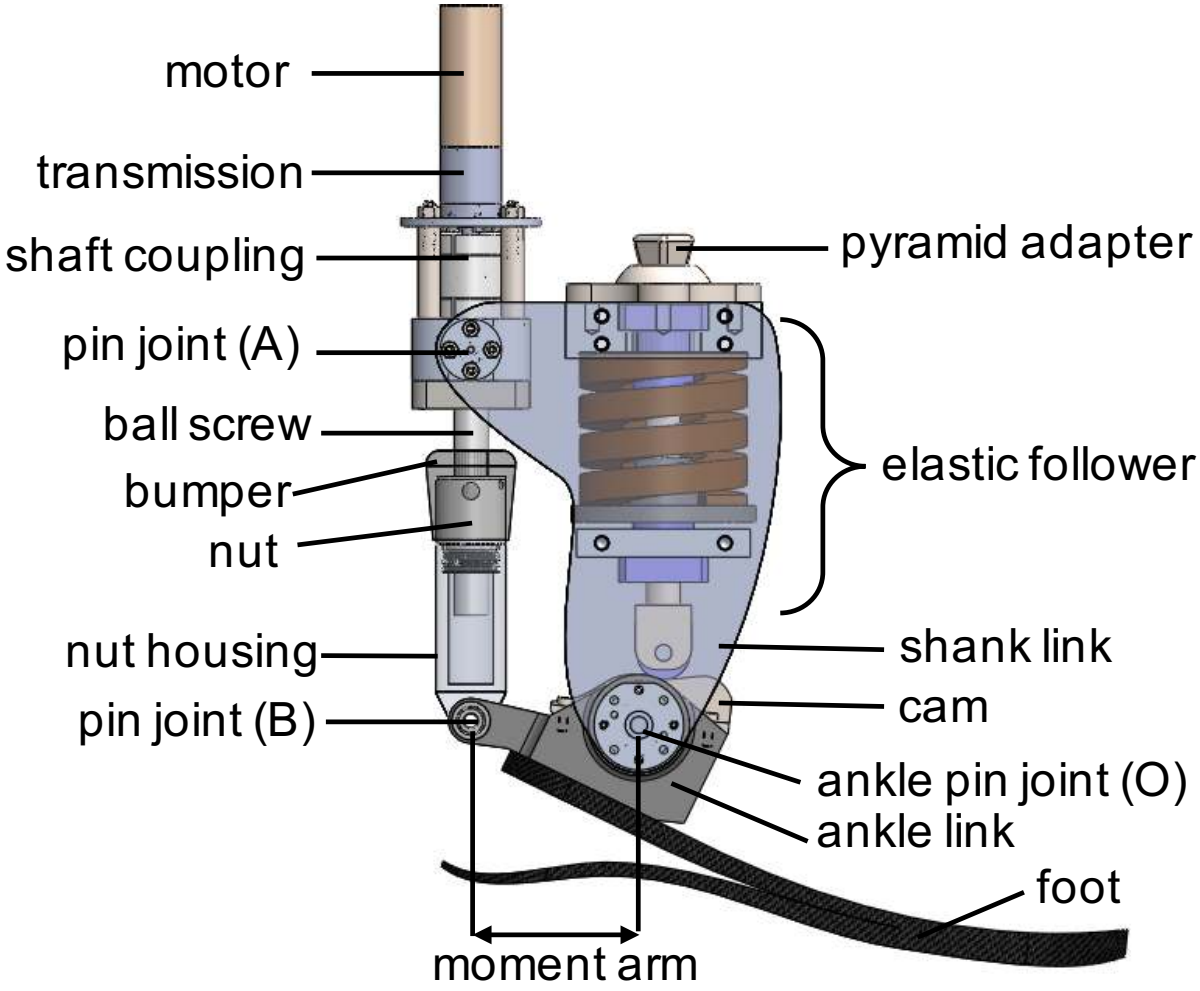


Figure 3.13: Illustrative rendering of the powered ankle-foot prosthesis prototype with major components labeled (note some components are transparent for increased visualization).

pin joint O). The active torque τ_a, generated by the powered drive train can be estimated as

$$\tau_a = rF_s. \tag{3.31}$$

where F_s is resultant linear force of the ball screw acting at joint (B) and r is the moment arm. Note (3.31) ignores the geometric nonlinearity cause by the drive train linkage mechanism. The force from the ball screw F_s is related to the torque τ_g at the transmission ouput shaft

(a) (b)

Figure 3.14: Images of the prototype PAFP with socket and shoe. (a) Side view. (b) Font view.

as

$$\eta_s 2\pi\tau_g = \ell F_s \tag{3.32}$$

where η_s is the ball screw efficiency. Additionally, the torque at the transmission output shaft τ_g and the motor torque τ_m are related by

$$\tau_g = \eta_g R_g \tau_m. \tag{3.33}$$

Therefore, the effective transmission gain R_T can be found by combining (3.31)-(3.33):

$$\tau_a = R_T \tau_m, \quad R_T = \eta_s \eta_g d R_g \frac{2\pi}{\ell} \tag{3.34}$$

Note that the parameters in (3.34) can be estimated from the component datasheets. The gain from the motor current command i_a to the active torque can be expressed as:

$$\tau_a = R_T k_\tau i_a, \tag{3.35}$$

where k_τ is the rated motor torque constant.

3.6 Discussion

Improvements to the modeling and optimization presented in this chapter are possible, for example, by including the motor dynamics and inertial effects into the optimization. Although this chapter focused exclusively on the parallel element (since it directly affects the torque), the approach could be extended to include optimization of combinations of nonlinear serial and parallel elastic actuators. The design presented in this chapter will serve as the prototype device for experimental evaluations.

Chapter 4

PERSONALIZED SYMMETRY CONTROL

4.1 Introduction

This Chapter introduces the personalized symmetry learning controller. The method relies on the adaptive backstepping iterative learning control (AB-ILC) algorithm [83], and can avoid divergent responses caused by unmodeled human dynamics. The algorithm learns a periodic command signal by iteratively reducing torque asymmetries of the prosthetic and intact limbs in the frequency domain. Divergence is avoided by monitoring the error, e.g., the difference between the prosthetic and intact ankle torques, and adjusting the frequency dependent learning gain accordingly. Consequently, subject parameter tuning is unnecessary, there is no need for able-bodied trajectory data, and both human and controller jointly adapt.

4.2 Problem formulation

4.2.1 Model abstraction

The conceptual model in Fig. 4.1 illustrates the research problem: choose the control signal u such that the prosthetic torque τ_p matches the biological (intact) ankle torque τ_b, e.g.,

$$\tau_p(t) = \tau_b(t), \tag{4.1}$$

where t in (4.1) represents the time-normalized gait cycle. The time-normalized gait cycle of each limb begins with heel-strike (HS) and ends with the next HS of same limb, thus $t \in [0\%\ 100\%]$. Note that in the following all biomechanical variables (e.g., angles, torques and powers) should be assumed in the sagittal plane, unless explicitly stated otherwise. The PAFP (robot) takes as an input the control signal u and outputs an active torque τ_a about the prosthetic ankle joint. In addition, the prosthesis provides a loading torque τ_l (e.g., from

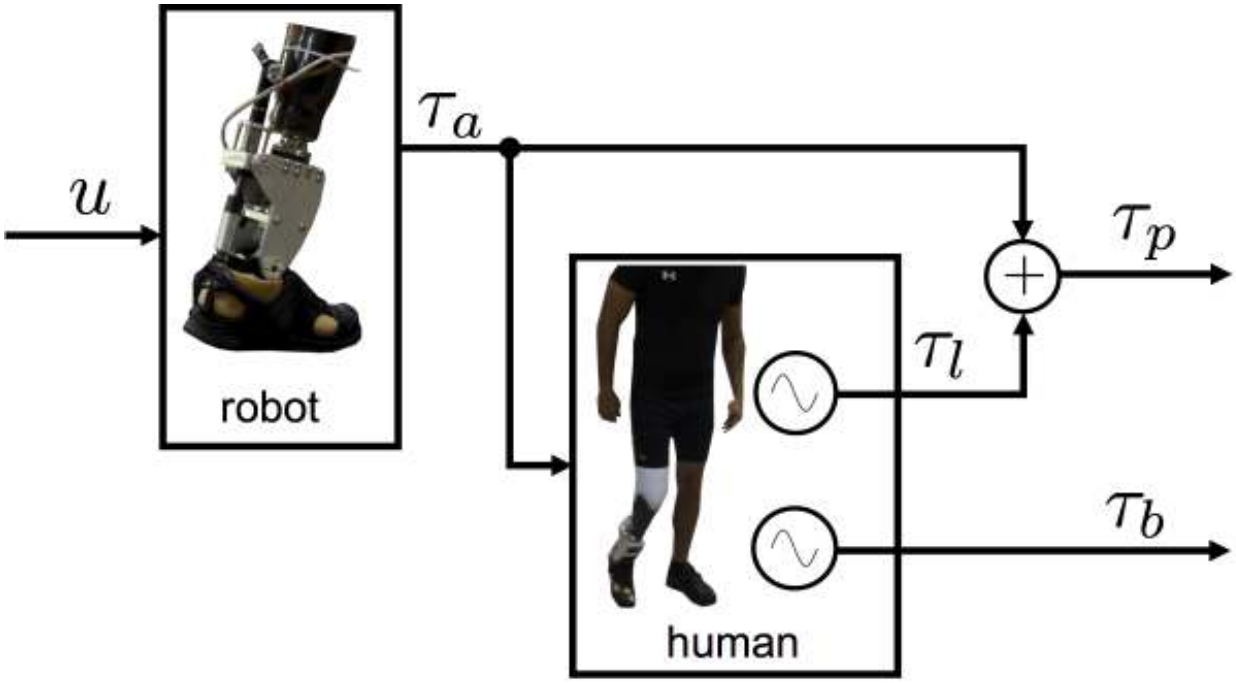


Figure 4.1: Conceptual model illustrating the research problem. The goal is to find the control signal u such that the active torque τ_a, summed with the (loading) passive torque τ_l, results in a total prosthetic torque τ_p that matches the biological (intact) torque τ_b. Note that the conceptual model assumes the loading torque τ_l and the biological torque τ_b are periodic signals generated by the human.

the passive dynamics of the prosthesis), which sum with the active torque τ_a to produce the total prosthetic torque τ_p,

$$\tau_p(t) = \tau_l(t) + \tau_a(t). \tag{4.2}$$

Note, that the active torque τ_a effects both the loading torque τ_l and the biological torque τ_b through the human-response dynamics, which can vary depending on the individual.

4.2.2 Iterative learning control

The iterative learning control (ILC) approach is to update the motor current command u at each iteration k (in the frequency domain):

$$u_{k+1}(\omega) = u_k(\omega) + \rho_k(\omega) G_m^{-1}(\omega) e_k(\omega), \tag{4.3}$$

where ρ_k is the learning gain, e_k is the error signal, and G_m is a model of the robot dynamics, e.g., the transfer function from motor current to active torque

$$G_m(\omega) = \frac{\tau_a(\omega)}{u(\omega)} \tag{4.4}$$

The error signal e_k in (4.3) is taken as the difference between the time-normalized mean biological torque and the mean prosthetic torque,

$$e_k(\omega) = \bar{\tau}_{b,k}(\omega) - \bar{\tau}_{p,k}(\omega), \tag{4.5}$$

and represents a measure of ankle torque asymmetry. In practice, the mean torque signals at each iteration k are computed over a sufficient walking trial, e.g., 30 (s), by taking the mean value, at each discrete percent gait, over all steps. Note that the frequencies ω in (4.3)-(4.5), e.g., the bin assignment of the discrete Fourier transform (DFT), depends on the discretization of the time-normalization and the mean step period $\bar{T}$, and are related by

$$\omega \in \left[0 \quad \omega_0 \quad 2\omega_0 \cdots \quad \tfrac{N}{2}\omega_0\right], \quad \omega_0 = \frac{2\pi}{\bar{T}} \tag{4.6}$$

where N is the (even) number of points in the time-normalized gait cycle, and ω_0 is the fundamental frequency of the mean step. Note the first element in (4.6) corresponds to the DC gain of the signal, and the last element the Nyquist frequency.

4.2.3 ILC Convergence

Ideally, repetitive application of the update law in (4.3) will result in convergence, at all frequencies ω, of the error signal in (4.5), e.g.,

$$\lim_{k \to \infty} |e_k(\omega)| = 0. \tag{4.7}$$

Convergence of (4.3) can be achieved at each frequency ω provided: (i) the phase error in the model G_m and the choice of learning gain ρ_k are sufficiently small [97], and (ii) the reference signal (e.g., $\bar{\tau}_{b,k}$ in this case) is fixed. Note that since the control signal u effects the biological torque τ_b (e.g., through the active torque τ_a) the reference signal can vary according to the personalized human-response dynamics, and thus it may not be possible to guarantee convergence.

4.2.4 Avoiding Divergence

Since the reference signal $\bar{\tau}_{b,k}$ is driven by the human-response dynamics, the problem investigated in the following is the selection of the frequency-dependent learn gain ρ_k such that the control signal u remains bounded, e.g.,

$$\lim_{k\to\infty} |u_k(\omega)| \leq K(\omega), \tag{4.8}$$

where $K(\omega)$ represents the frequency dependent upper bound. Boundedness of the control signal u is critical for safety. Simply applying the update law in (4.3) with a fixed learning gain $\rho_k(\omega) = \rho(\omega)$ could result in an unbounded control signal u_k. Another scenario is that the magnitude of the error signal in (4.5) could decreases at each iteration k, but the magnitudes of each torque component – prosthetic torque $\bar{\tau}_{p,k}$ and biological torque $\bar{\tau}_{b,k}$ – could increase at each iteration k. Such a scenario would lead to unbounded growth of the control signal u, and poses a safety issue for the user.

4.3 Adaptive Backstepping ILC

To address the problems outline above, this work proposes an adaptive backstepping technique, which monitors the error and the reference signals growth and adapts the frequency dependent learning gain ρ_k. There are three components to the AB-ILC algorithm: (i) conditional statements that determine how the algorithm updates its internal states, (ii) a cache that stores the algorithm's internal states, and (iii) the update law that results in a new control signal u_{k+1}. Each component is described below.

4.3.1 Update Conditions

Three update conditions determine how the update law evolves and are stored as boolean variables. The first condition $\mathcal{C}_1$ determines if the error signal e_k is smaller in magnitude than the previously smallest observed error signal e^*_{k-1},

$$\mathcal{C}_1(\omega) = \begin{cases} 1 & \text{if} \quad |e_k(\omega)| < |e^*_{k-1}(\omega)| \\ 0 & \text{else} \end{cases}. \tag{4.9}$$

Another condition, $\mathcal{C}_2$, determines if the error signal is larger in magnitude than the previously smallest observed error signal e^*_{k-1} plus padding ϵ,

$$\mathcal{C}_2(\omega) = \begin{cases} 1 & \text{if} \quad |e_k(\omega)| > |e^*_{k-1}(\omega)| + \epsilon(\omega) \\ 0 & \text{else} \end{cases} \tag{4.10}$$

where the frequency dependent padding $\epsilon(\omega)$ is an estimate of the expected variation in the error signal e_k, (e.g., a noise estimate). The final condition $\mathcal{C}_3$ determines if the reference signal $\bar{\tau}_{b,k}$ has grown larger in magnitude than a scaled version of the initial error e_0,

$$\mathcal{C}_3(\omega) = \begin{cases} 1 & \text{if} \quad |\bar{\tau}_{b,k}(\omega) - \bar{\tau}_{b,0}(\omega)| > \alpha |e_0(\omega)| \\ 0 & \text{else} \end{cases} \tag{4.11}$$

where the constant α represents the allowable growth of the reference signal $\bar{\tau}_{b,k}$ relative to the initial error e_0.

4.3.2 Backstepping Cache

The backstepping cache updates and stores the state of the algorithm, based on the conditions outlined above, as follows:

$$\rho_k(\omega) = \begin{cases} \frac{1}{\gamma}\rho_{k-1}(\omega) & \text{if} \quad \mathcal{C}_2(\omega) \text{ or } \mathcal{C}_3(\omega) \\ \rho_{k-1}(\omega) & \text{else} \end{cases} \tag{4.12}$$

$$e_k^*(\omega) = \begin{cases} e_k(\omega) & \text{if} \quad \mathcal{C}_1(\omega) \text{ and } \neg\mathcal{C}_3(\omega) \\ e_{k-1}^*(\omega) & \text{else} \end{cases} \tag{4.13}$$

$$u_k^*(\omega) = \begin{cases} u_k(\omega) & \text{if} \quad \mathcal{C}_1(\omega) \text{ and } \neg\mathcal{C}_3(\omega) \\ u_{k-1}^*(\omega) & \text{else} \end{cases} \tag{4.14}$$

where $\gamma > 1$ determines the decay rate of the learning gain ρ_k if a sufficient increase in the error signal e_k is observed (condition $\mathcal{C}_2$ in (4.10)) or a sufficient increase in the reference signal $\tau_{b,k}$ is observed (condition $\mathcal{C}_3$ in (4.11)) Note that the frequency dependent padding ϵ in (4.10) allows the error signal to fluctuate within the expected noise region without penalty to the learning gain ρ_k. Additionally, e_k^* in (4.13) and u_k^* in (4.14) are updated if and only if magnitude of the error e_k is strictly decreasing (condition $\mathcal{C}_1$ in (4.9)) and the magnitude of the reference signal $\bar{\tau}_{b,k}$ remains bounded by the initial error e_0 (condition $\neg\mathcal{C}_3$ e.g., the opposite of $\mathcal{C}_3$ in (4.11)).

4.3.3 AB-ILC Update Law

The update law utilizes the cached values, outlined above, to determine the new control signal $\hat{u}_{k+1}$, as:

$$\hat{u}_{k+1}(\omega) = u_k^*(\omega) + \rho_k(\omega) G_m^{-1}(\omega) e_k^*(\omega). \tag{4.15}$$

In practice, it is useful to eliminate the control signal during some region of the gait cycle, e.g., during swing phase, therefore, the signal defined in (4.15), during the regions defined

by t_1 and t_2, is set to zero:

$$\tilde{u}_{k+1}(t) = \begin{cases} \hat{u}_{k+1}(t) & \text{if } (t < t_1) \text{ and } (t > t_2) \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

Finally, the new control signal u_{k+1} is found as

$$u_{k+1}(\omega) = \zeta(\tilde{u}_{k+1}(\omega)) \tag{4.17}$$

where ζ represents a zero-phase low-pass filter with cutoff frequency ω_c that smooths the signal.

4.4 AB-ILC Implementation

To understand the choice in ω_{max}, truncated versions of an error signal (from preliminary human subject testing), defined in (4.5), e.g., with higher harmonics removed, were analyzed and are shown in Fig. 4.2. This is important, since from the control law in (4.15) uses the error signal e_k (or e_k^*) to update the new control law u_k, thus if not enough harmonics are used, critical frequency information could be lost. In contrast, high frequencies are typically more sensitive, and should be eliminated if possible. Note that the data in Fig. 4.2 was processed in the way described is Sect. 6.1.4, namely filtered at 6 (Hz), thus the signal has very little high frequency content. Additionally, the step period was approximately $T = 1.3$ (s), thus the fundamental frequency is $f_0 = 0.77$ (Hz). Hence, harmonics higher then 10 won't contribute any new information. Note that the bulk of the error content is concentrated near 60% gait, which is close to toe off. If not enough harmonics are used the pronounced peak near toe off is diminished, and their is content larger than the original error in some areas, e.g., see harmonics $1 - 4$ near 50% gait.

4.5 Discussion

Note that since the control signal u is determined by the previous optimal signal u^* and the previously smallest achieved error e^*, the growth is always bounded since

$$\rho_k e_k^*(\omega) \leq \rho_0 e_0. \tag{4.18}$$

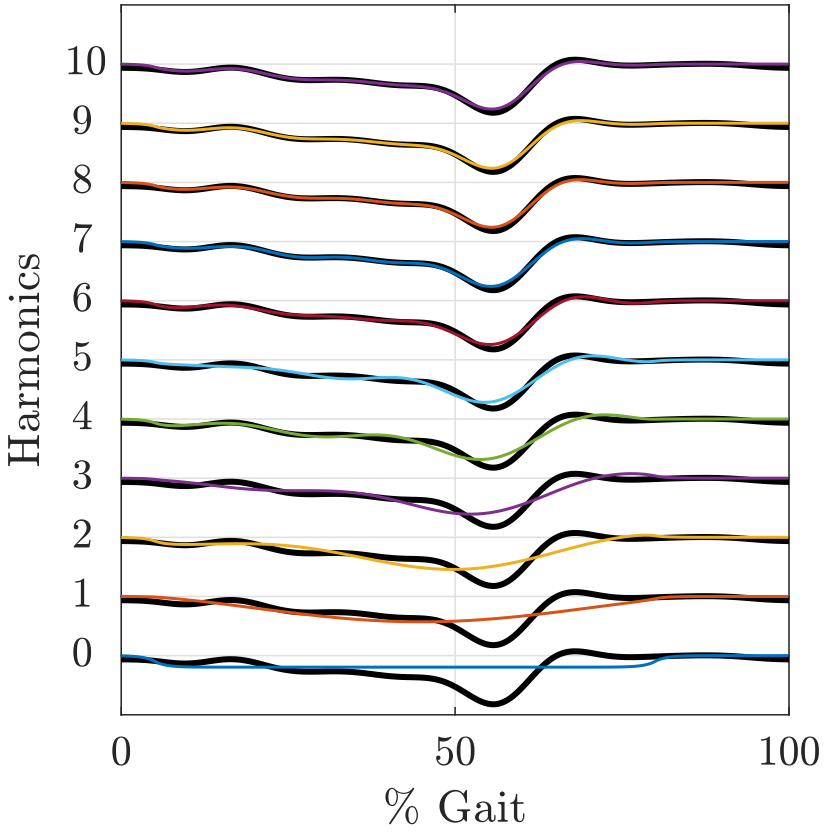


Figure 4.2: Error e, from preliminary human subject testing, reconstructed using truncated inverse Fourier transforms.

Chapter 5

SYSTEM INTEGRATION

5.1 Overview

The controller architecture is illustrated as a block diagram in Fig. 5.1 and consists of two main signal paths: real-time and offline. The real-time signals are handled by the embedded system, illustrated in Fig. 5.2, worn by the user as a small backpack, and include: sampled insole force sensor signal $\hat{F}_h$, a heel-strike (HS) detection algorithm that stores the timestamp n_h of the most recent HS and estimates the mean step period $\bar{P}$, a pattern generator for modulating the active motor current i_a, and a low-level proportional integral (PI) controller for tracking the desired motor current i_d. Offline, after each walking trial (e.g., learning iteration k), inverse dynamics are computed on the motion capture data by the host computer, depicted in Fig 5.2, resulting in the mean prosthetic torque $\bar{\tau}_{p,k}$ and mean biological torque $\bar{\tau}_{b,k}$, which are used to compute the error signal e_k define in (4.5), and subsequently, using the AB-ILC algorithm, a new control signal u_{k+1} defined in (4.17), is then encoded into the pattern generator (lookup table). Each block in Fig. 5.1 is discussed next.

5.2 Insoles

The custom insoles, shown in Fig. 5.3(a) are composed of three force-sensitive resistors (FSRs) (Tekscan FlexiForce A401) located approximately at the heel, mid, and toe regions. The resistance of the FSR is converted to a voltage via a custom sensor circuit and, as shown in Fig. 5.2, interfaces with the embedded system to subsequently digitized the force signals with a 12-bit analog-to-digital-converter (ADC). Details of the circuit are provided in Appendix D.

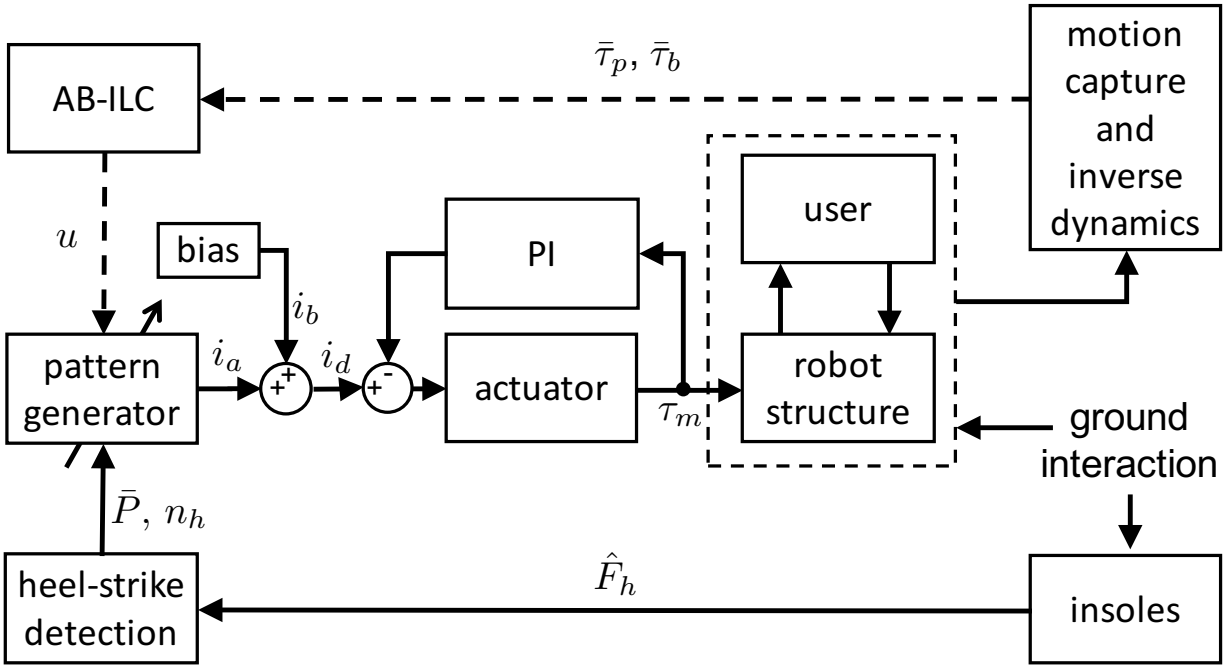


Figure 5.1: Block diagram of controller architecture. Solid lines represent real-time signals and dashed lines represent off-line signals.

5.3 Heel-strike detection

The HS detection algorithm determines the time instant of the most recent HS n_h (stored as an integer corresponding to the timestamp e.g., number of cycles). The detection algorithm utilizes the heel sensor output from the custom insole pictured in Fig. 5.3(a), a finite state machine (FSM) shown Fig. 5.3(b), a cascaded digital filter structure shown in Fig. 5.3(c), and a moving average filter shown in Fig. 5.3(d). In general other sensors (e.g., ankle angle encoder, accelerometer) could be combined in one FSM to provide gait phase detection [35]. The HS detection works as follows. For each control loop n, the raw force signal $\hat{F}_h[n]$ is digitally filtered according to the filter structure depicted in Fig. 5.3(c), which is a cascade of two digital low-pass filters. The filter coefficients for each stage of the filter structure are derived from a continuous time low-pass first-order Butterworth filter with cutoff frequency

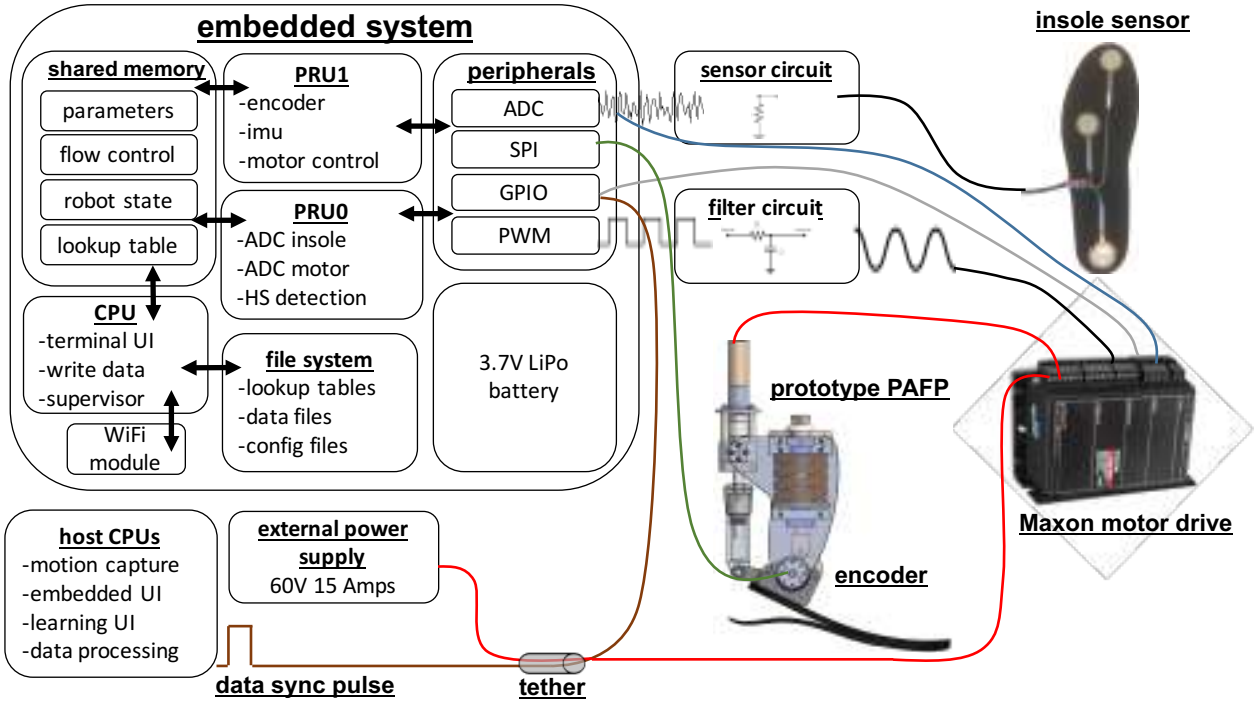


Figure 5.2: Embedded system integration and wiring.

of 6 Hz. An estimate of the force rate of change $\delta F_h[n]$ is computed as the difference between the output of the first stage and the final (filtered) output $F_h[n]$. The FSM for HS detection is depicted in Fig. 5.3(b) and consists of two states: state 1 (no heel contact) and state 2 (heel contact). The transition between states is determined by thresholding the force signal $F_h[n]$ and the force rate of change $\delta F_h[n]$. If the force rate of change $\delta F_h[n]$ is increasing (e.g., $\delta F_h[n] > \delta F_{th}$) and the force $F_h[n]$ below the threshold value (e.g., $F_h[n] < F_{th}$), then a transition from state 1 to state 2 occurs. If the force rate of change $\delta F_h[n]$ is decreasing (e.g., $\delta F_h[n] < -\delta F_{th}$) and the force $F_h[n]$ above the threshold value (e.g., $F_h[n] > F_{th}$), then a transition from state 2 to state 1 occurs. When a transition from state 1 to state 2 is identified, the current timestamp n is store in the HS variable n_h, the period of the current step s is calculate as

$$P[s] = n_{h-1} - n_h, \tag{5.1}$$

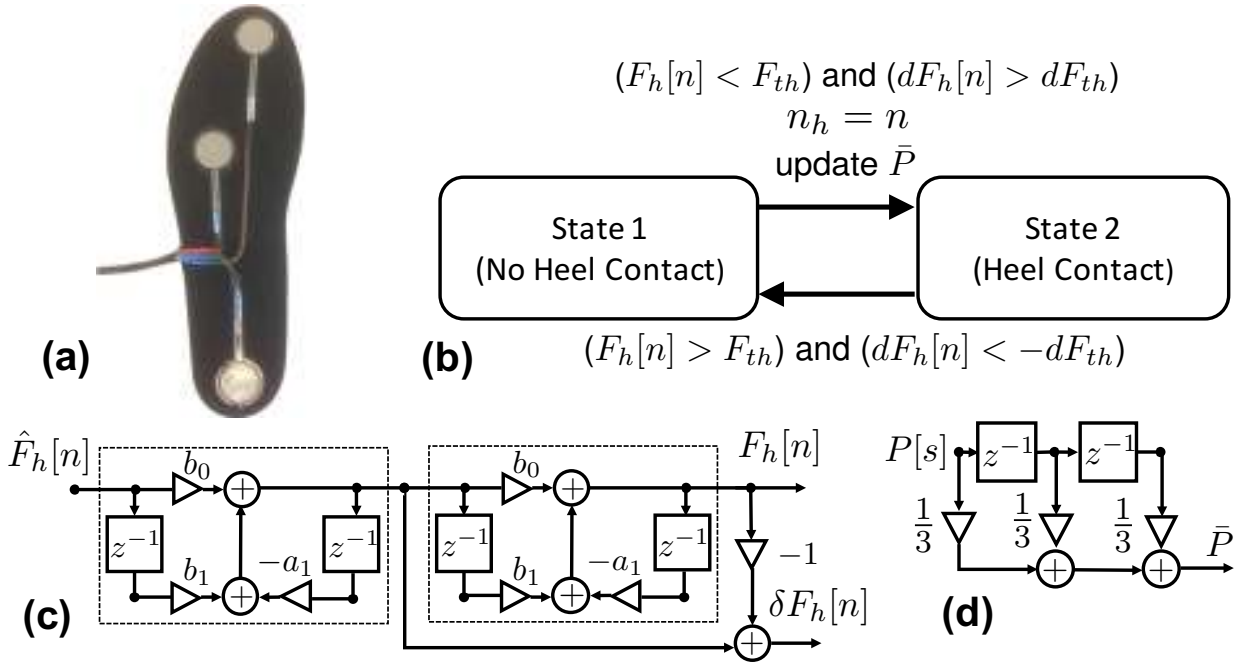


Figure 5.3: Components of the HS detection algorithm: (a) Custom insole sensor composed of three FSRs located approximately at the heel, mid, and toe regions; (b) Finite state machine (FSM) for HS detection, where F_h is the filtered force signal from the heel sensor, δF_h is an estimate of the rate of change of the force signal, F_{th} is the force threshold, δF_{th} is the rate of change threshold, n_h is the timestamp, in cycles, corresponding to HS, i is the current timestamp, and $\bar{P}$ is an estimate of the mean step period; (c) the cascaded filter structure, where $\hat{F}_h$ is the raw force signal; (d) moving average filter used to estimate the mean step period $\bar{P}$, where $P[s]$ is the period of the most recent step s.

where n_{h-1} is the previous HS timestamp, and the moving average filter with window size of three, depicted in Fig. 5.3(d), determines the mean step period $\bar{P}$, which allows for fluctuations in walking speed (e.g., by adjusting the control signal). Note that it is conceivable to implement the HS detection algorithm with analog circuits as presented above, which could be important for low-power commercial implementations. Figure 5.4 shows the results of the HS detection during a human subject experiment.

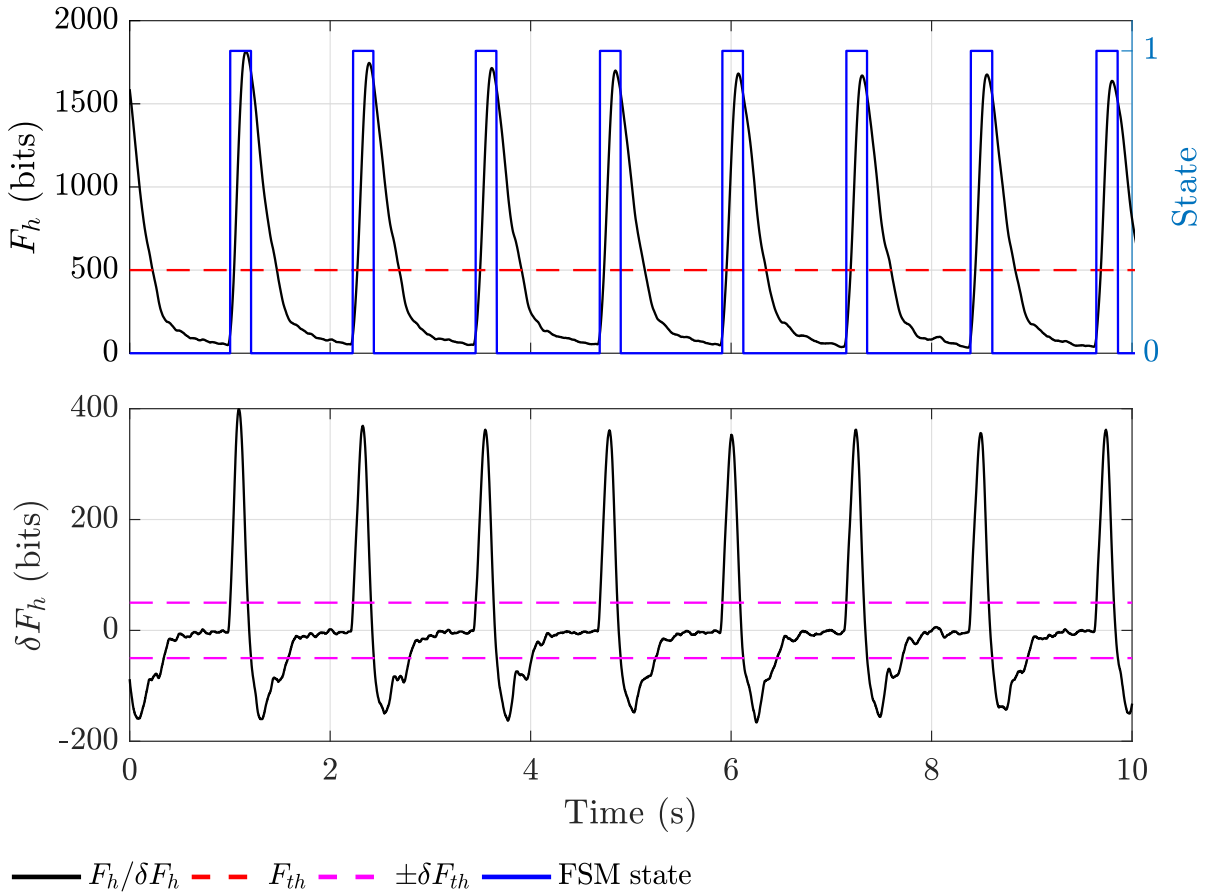


Figure 5.4: Heel-strike detection during human subject experiment. Top plot: time traces of heel force sensing resistor (FSR), after digital filtering (see Fig. 5.3). Threshold F_{th} can be seen as the dashed red line. Finite state machine (FSM) output can be seen as solid blue trace. Bottom plot: rate of change δF_h (black) and thresholds $\pm\delta F_{th}$ (magenta).

5.4 Pattern generator

The pattern generator is implemented as a lookup table. For each iteration k (e.g., each learning trial) the lookup table is populated with the time normalized control signal u_k, defined in (4.17), from the AB-ILC algorithm. The lookup table is stored as a 1000 element array, with each element a 16-bit signed integer. The first element corresponds to 0% gait and the last element corresponds to 99.9% gait. At each time step n the corresponding active

motor current command i_a is found as

$$i_a[n] = \kappa \cdot u[n^*] \tag{5.2}$$

where $\kappa \in [0, 1]$ allows scaling of the learned command signal u, the lookup table array index n^* is computed as

$$n^* = \text{floor}\left(\frac{(n - (n_h - \Delta_{HS})) \cdot 1000}{\bar{P} - \frac{\bar{P}}{1000}}\right), \tag{5.3}$$

where the floor operator rounds down to the nearest integer, and Δ_{HS} is an estimate of the delay in HS detection, described in Sect. 5.3, calibrated with post-hoc HS identification using the vGRFs signals from force plates (details of the HS indication and data processing of the vGRFs and motion capture data are is presented in Sect. 6.1.4). The scalar gain κ is used to scale the learned command signal u slowly, through the user interface, described in Sect.5.8.1, scaling up as the user walks, over about a $1 - 2$ minute period, for each new learning iteration k. Note interpolation between adjacent elements in the lookup table is possible, however, in practice the resolution of (5.3) has been found to be sufficient with a lookup table of array length 1000.

5.5 Bias current

The cam-based parallel spring results in the prototype PAFP having a plantarflexed neutral position, e.g., when unloaded, caused from the preload in the elastic follower. Therefore, to prevent toe drag during swing, a bias current term i_b offsets the preload. The procedure for choosing the bias term i_b is described in Sect. 6.1.3.

5.6 Low-level PI controller

The low-level current tracking is implemented with a 4-quadrant servo controller (Maxon, ESCON 70/10), and can be seen in Fig. 5.2. The desired motor current at the n^{th} control loop,

$$i_d[n] = i_a[n] + i_b[n], \tag{5.4}$$

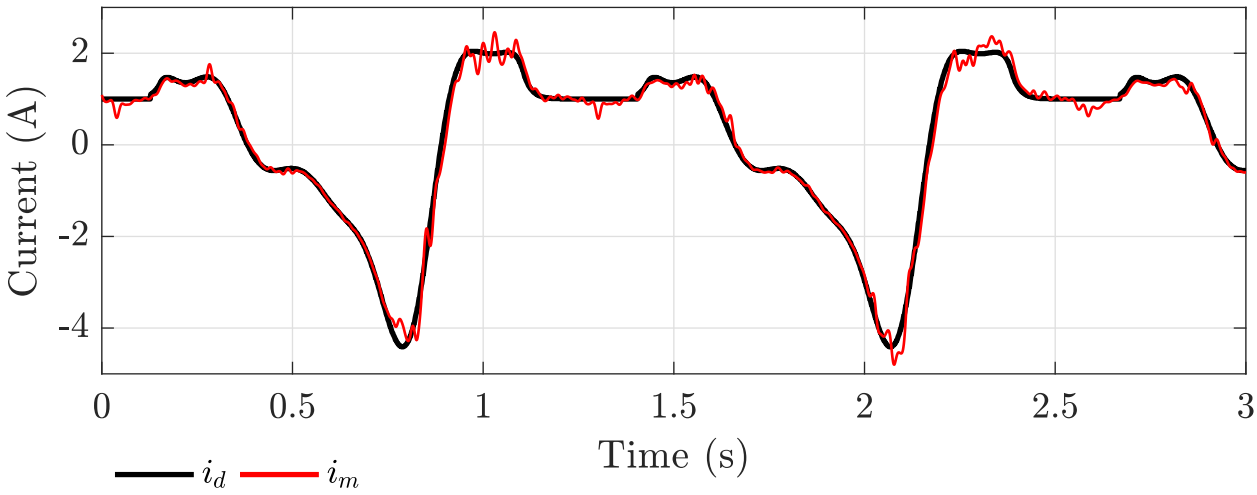


Figure 5.5: Current tracking during human subject experiment. The desired current i_d, defined in (5.4), consists of the active current i_a, defined in (5.2), and the bias current i_b. The actual motor current i_m is sampled from the Maxon motor drive.

is encoded by the embedded system as a pulse-width-modulation (PWM) signal, where the pulse-width encodes the desired current, which is subsequently filtered with an analog first-order low-pass RC-filter, with cutoff frequency f_c chosen as

$$f_c = \frac{1}{2\pi RC} \approx 48 \text{ (Hz)}, \quad R = 33 \text{ k}\Omega, \quad C = 0.1 \ \mu\text{F} \tag{5.5}$$

The servo controller takes as input the analog voltage set point (e.g., the filtered PWM signal) and provides the required motor commutation, using hall-effect feedback from the brushless DC motor. The servo controller implements a PI loop for tracking, which was automatically tuned with Maxon software. The actual motor current i_m is sampled from the Maxon motor drive via the embedded system's ADC. Current tracking during a human subject experiment is shown in Fig 5.5. The filter circuit is detailed in Appendix D.

5.7 Encoder

A 12-bit capacitive encoder, which can be seen in Fig. 5.2 attached to the prototype PAFP, is used to sense the angular position of the ankle joint. The encoder interfaces with the

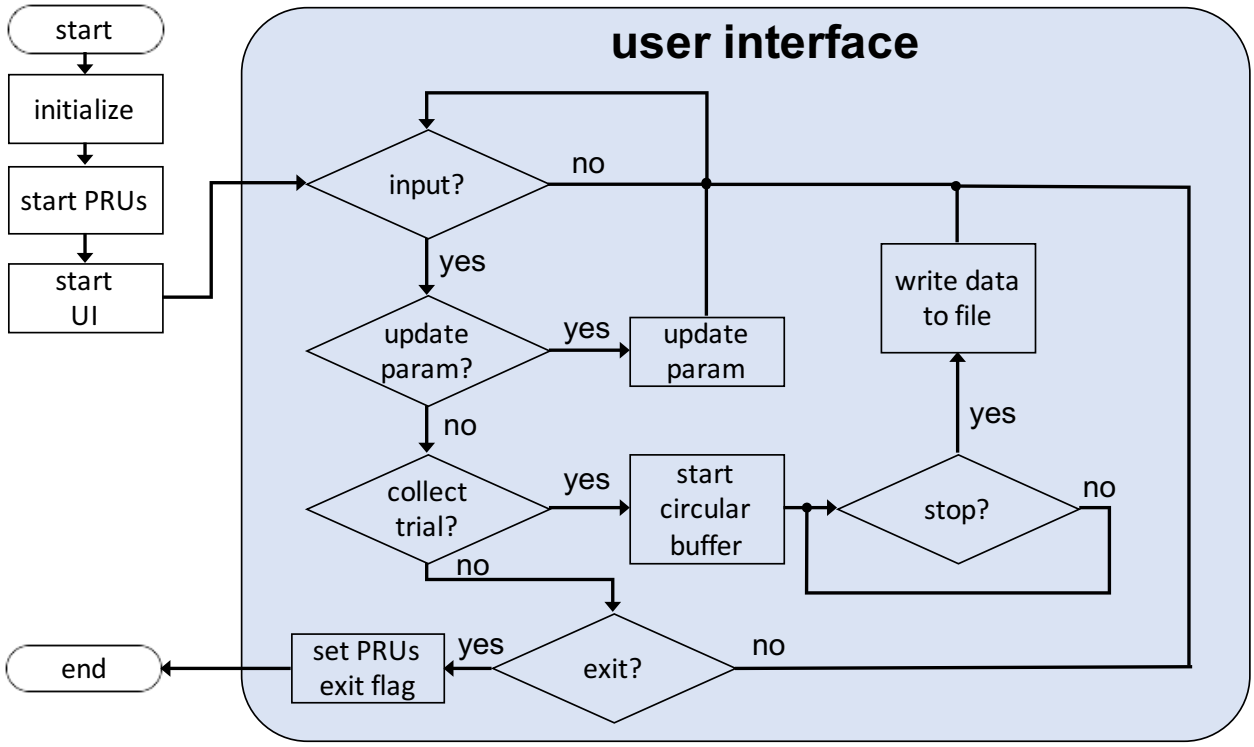


Figure 5.6: Flowchart illustrating the software program running on the main CPU.

embedded system via a serial peripheral interface (SPI).

5.8 Embedded implementation

The embedded system, utilizing a Beaglebone Black (Texas Intruments), integrates a 32-bit CPU, two 32-bit programmable real-time units (PRUs), voltage divider circuitry for the FSRs, and the servo controller and analog filter described above. The CPU and the two PRUs share a block of memory, which can be read and written to by all three processors, depicted in Fig. 5.2. The shared memory space integrates flow control, parameters, the state of the robot (e.g., sensor and control values at each time step n), and the lookup table. In addition, as described above, the system make use of standard peripherals, e.g., ADC, SPI, general purpose input output (GPIO) and PWM. The system is packaged in a small 3D-printed case (shown in Appendix D), which is worn by the user as a backpack. The 32-bit CPU,

running Linux, executes high level commands, such as loading new lookup tables into shared memory, updating parameters, and writing data to files. The two PRUs provide real-time functionality, e.g., ADC sampling and motor control. The real-time system provides a control loop frequency of 1 (kHz) and, excluding the servo controller, is powered by an on-board lithium ion battery. A tether, shown in Fig. 5.2, supplies power to the servo controller and syncs the embedded data with the external motion capture using a pulse generated by the Vicon system.

5.8.1 Embedded software

CPU program

The main program running on the CPU, shown in a flowchart in Fig. 5.6, is responsible for the user interface, which is used to change parameters, load new lookup tables and to initiate data collection. The terminal based user interface (UI) is accessed by the host computer though a secure shell (SSH) connection over WiFi. The terminal UI is shown Fig. 5.7.

```
------------------------------------------------------------------------
hs_delay = 40, u_bias = 1.00, ProsSide = L, FF enabled = 0, FFgain = 0.00
Menu: a - Enter new hs_delay
      s - Enter new u_bias
      d - Change prosthetic limb side
      f - Collect trial
      g - Save parameters
      h - Load parameters
      j - Load Feedforward lookup table
      k - Toggle feedforward control
      l - Enter new FFgain
      p - Test Feedforward
      q - Step Response
      e - exit
------------------------------------------------------------------------
```

Figure 5.7: Terminal user interface for the embedded system.

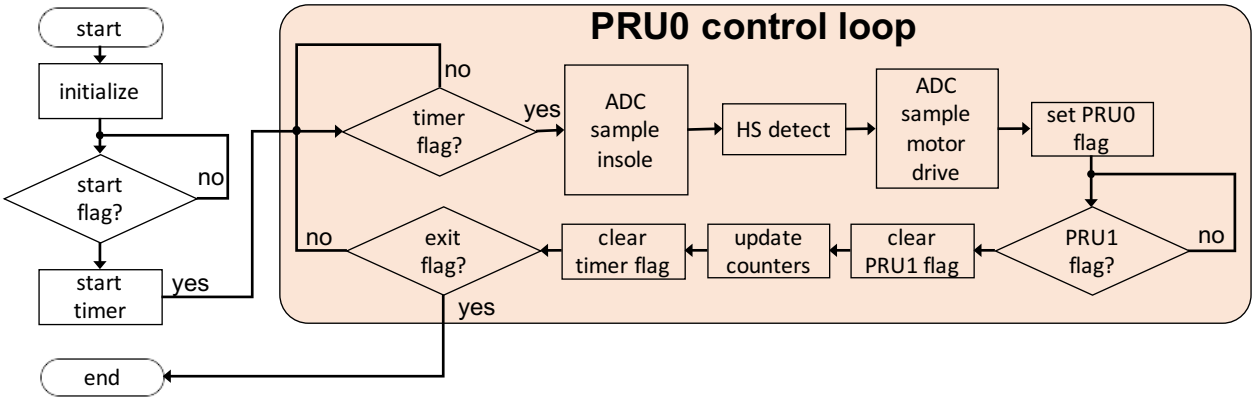


Figure 5.8: Flowchart illustrating the software program running on the PRU0.

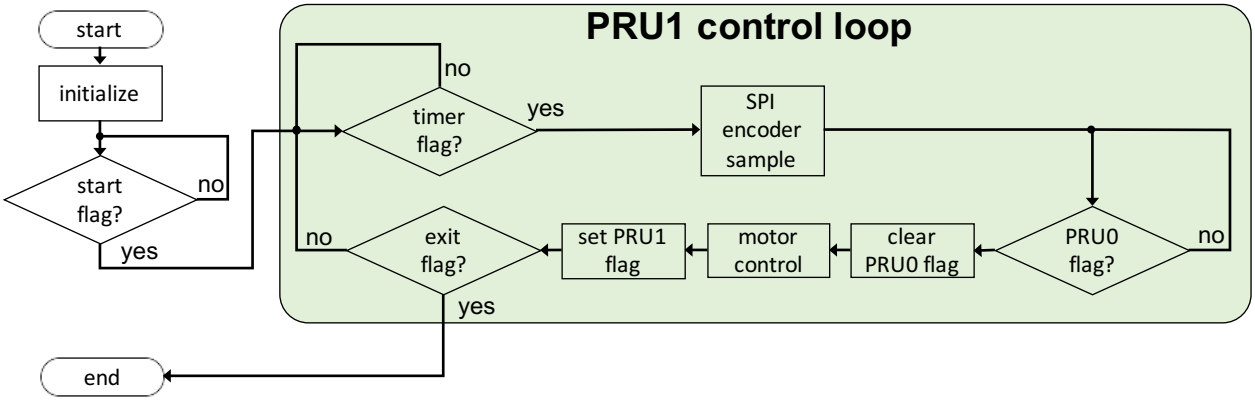


Figure 5.9: Flowchart illustrating the software program running on the PRU1.

PRU0 program

The program on PRU0, shown in Fig. 5.8, is responsible for sampling (e.g., ADC) the insole sensors and the motor velocity and current (transmitted by the Maxon motor drive as analog signals), and stored in the robot state section of the shared memory (see Fig. 5.2). In addition, PRU0 implements the HS detection algorithm outlined in Sect. 5.3 and depicted in Fig. 5.3. Finally, PRU0 maintains system level counters and initiates the timer for the control loop frequency. The code is detailed in Appendix C.

PRU1 program

The program on PRU1, shown in Fig. 5.9, is responsible for sampling the encoder and storing the result in the robot state portion of the shared memory, and sampling the lookup table (e.g., pattern generator), which is also stored in shared memory (see Fig. 5.2). Note since PRU0 implements HS detection, PRU1 must wait until PRU0 is finished sampling the insole sensors and HS detection before executing the pattern generator. This is accomplished through flow control bits, stored in shared memory, and illustrated in Figs 5.8-5.9 as flags.

Chapter 6

HUMAN SUBJECT EXPERIMENTAL EVALUATION

6.1 Experimental methods

The experiment is designed to determine: (1) if the symmetry control method results in symmetrical ankle mechanics, and, (2) if the symmetry control method results in a reduction of peak intact knee adduction moment (KAM) and peak hip intact adduction moment (HAM), e.g., load factors associated with OA progression in the general and amputee populations [62, 63, 65]. Towards this, three experimental conditions are assessed: The first experimental condition, prescribed condition, consists of the subject ambulating with their prescribed passive foot prosthesis. The second condition, passive condition, consists of the subject ambulating with the powered ankle-foot prosthesis in passive mode, e.g., $i_a = 0$ during the passive condition experiment. And the third condition, symmetry control condition, consists of the subject ambulating with the powered ankle-foot prosthesis after the symmetry controller has converged. The hypotheses are formally stated below:

- **H1.1**

  The symmetry controller will reduce ankle moment asymmetry, when compared with the passive condition.

- **H1.2**

  The symmetry controller will reduce ankle angle asymmetry, when compared with the passive condition.

- **H1.3**

  The symmetry controller will reduce ankle power asymmetry, when compared with the passive condition.

- **H2.1**

  The symmetry controller will reduce the intact peak knee adduction moment, when compared with the passive condition.

- **H2.2**

  The symmetry controller will intact reduce the peak hip adduction moment, when compared with the passive condition.

6.1.1 Experimental Setup

The experimental setup, shown in Fig. 6.1 consisted of a split-belt force-sensing treadmill (Bertec), a 12-camera motion capture system (Vicon), and a human subject donning either the powered ankle-foot prosthesis with custom embedded system and tethered power supply, or their prescribed prosthesis. The marker set consisted of 36 reflective markers in an arrangement defined by Vicon's plug-in gait model. The prototype powered ankle-foot prosthesis was fitted to left leg of the subject by a certified prosthetist.

6.1.2 Subject

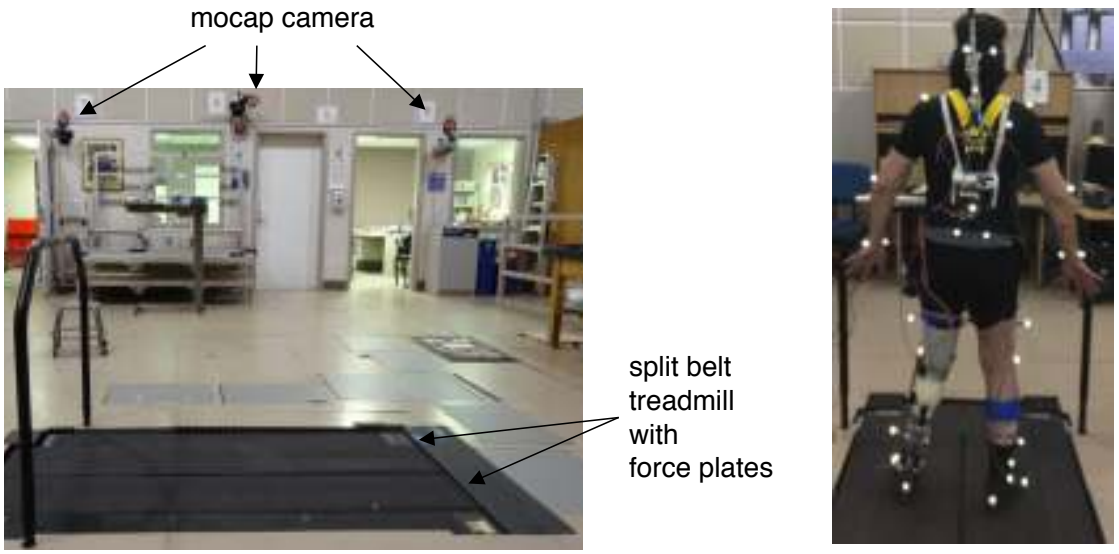
The subject provided written informed consent to participate in the experimental protocol, approved by the VA and UW Institutional Review Boards, a copy of which is provided in Appendix E. The subject was a healthy, active 82 (kg) unilateral below-knee amputee. Table 6.1 provides details of the subject's characteristics.

6.1.3 Protocol

The experimental protocol, illustrated in Fig. 6.2, was divided into three parts:

Setup

After consenting, self-selecting walking speed was identified, with their prescribed prosthesis, by calculating walking speed over 3 trials of overground walking along a 20 (m) hallway while



(a) (b)

Figure 6.1: Experimental setup. (a) Split belt treadmill with force plates. (b) Plug-in gait model reflective maker arrangement.

timed by a study investigator. Next, a certified prosthetist fitted the prototype PAFP to the subject, which include walking on the treadmill to ensure proper alignment. After fitting, anthropometric measurements were taken and reflective markers were placed, shown in 6.2. Total setup time was approximately $1 - 1.5$ hours.

Pretest

Before ambulating, the subject was ask to choose a bias current i_b, through trial and error, which felt comfortable during standing. The bias current i_b, described in Sect. 5.5, effectively adjusts the neutral position of the ankle-foot prosthesis. Next, the subject was allowed to acclimated to the powered ankle-foot prosthesis by walking on the treadmill for approximately 5 minutes. Additionally, the subject was asked to verify their choice of bias current i_b during this period. Total pretest time was approximately 10 minutes.

Experimental trials

After acclimation, the experimental trials commenced. Before each data collection of each iteration k, the control signal was adjusted incrementally over a span of $1 - 2$ minutes (excluding iteration $k = 0$, which had a control signal $u = 0$). After the subject was comfortable with the new control signal, data was collected for 30 (s). After each learning iteration k

Table 6.1: Subject characteristics.

Gender	male
Age (yrs)	50
Height (cm)	180
Intact leg length (mm)	900
Prosthetic leg length (mm)	890
Mass prescribed (kg)	82
Mass prototype (kg)	85
Etiology	traumatic
Prescribed prosthesis	Ossur VP

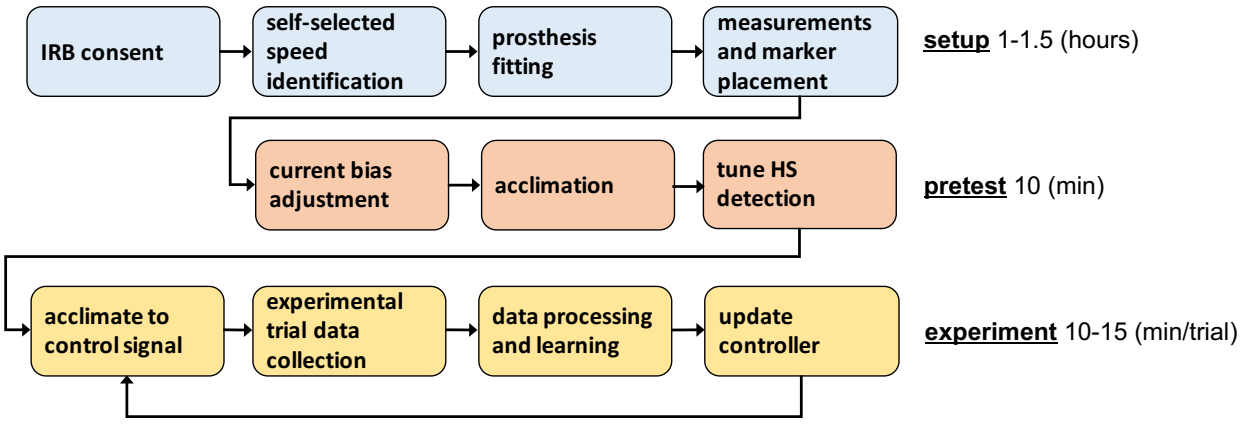


Figure 6.2: Experimental protocol divided into three parts: setup, pretest and experiment.

the subject was allowed to sit down while data was processed, the AB-ILC algorithm was executed and a new control law uploaded to the embedded system. This process was repeated until the measured error signal e_k could no longer be reduced. Each learning trial, including control signal acclimation, data collection, processing and learning took approximately 10-15 minutes.

6.1.4 Data Processing

An overview of the data processing is shown in Fig. 6.3, and implementation details of the biomechanics toolbox (biomechtb), developed as an open source tool, can be found in Appendix B. Details of the processing are as follows. Motion and vGRF data were captured by the Vicon system at 120 (Hz) and 1200 (Hz), respectively. Sensor data, and motor current and velocity were captured by the embedded system, via an on board 12-bit analog-to-digital converter, at 1000 (Hz) and synchronized with the Vicon data. The motion and vGRF were processed with a standard inverse dynamic gait model (plug-in gait model, Vicon), which also used anthropometry data of the subject, resulting in joint mechanics. Then, joint mechanics and vGRFs were low-pass filtered with a zero-phase 4^{th} order Butterworth filter with cutoff frequency of 6 and 50 (Hz), respectively. Additionally, motor current and velocity were low-

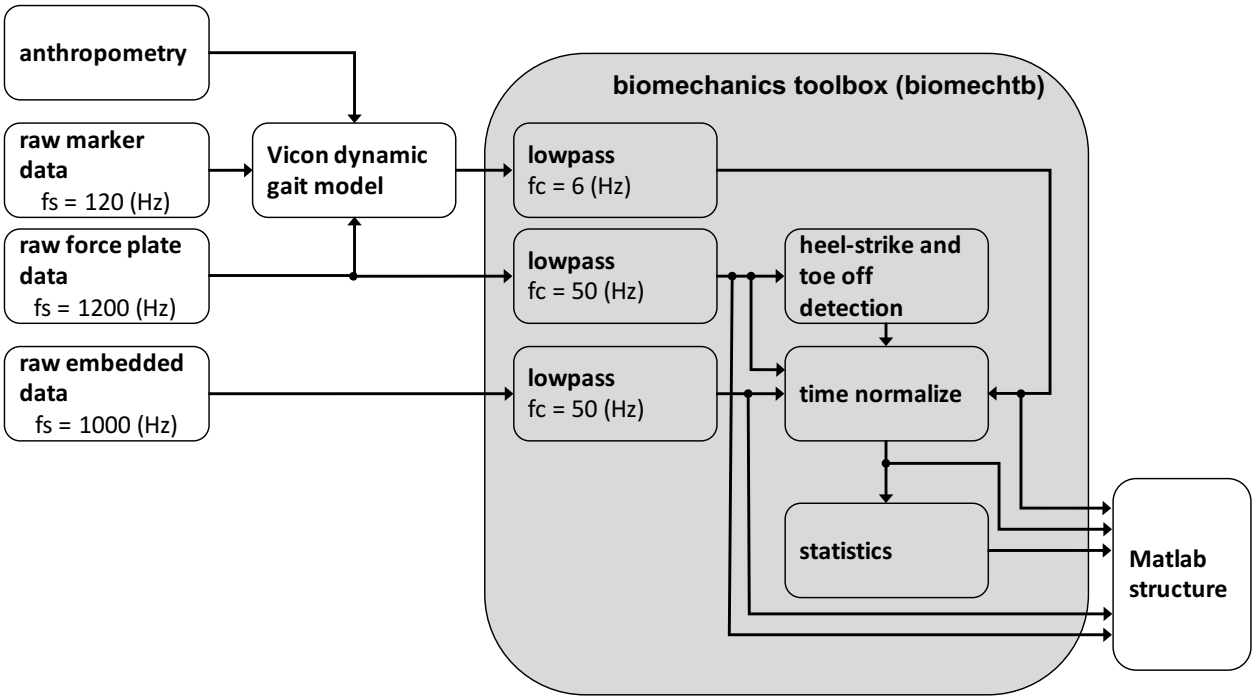


Figure 6.3: Data processing pipeline. The outputs are time series of joint kinematics, kinetics, vGRF and sensor and control data. In addition, time series are also time normalized for analysis.

pass filtered with a cutoff of 50 (Hz). HSs and toe offs were identified post-hoc from the filtered vGRF data as follows: HSs were defined as the frames that transitioned to a vGRF magnitude greater than 50 (N); toe offs were defined as the frames that transitioned to vGRF magnitude of less than 50 (N). All data was then time-normalized, via interpolation, to both 100% of gait and 100% of stance for statistical analysis. The mean intact ankle torque $\bar{\tau}_b$ and mean prosthetic ankle torque $\bar{\tau}_p$ were used to compute the error signal e_k for the AB-ILC algorithm described in Ch. 4 (implementation can be found in Appendix B.2).

6.1.5 Quantifying Asymmetry

In this work, for each gait variable (e.g., ankle torque), each individual step is assigned an asymmetry measure (AM) by computing the average norm difference of the step of interest

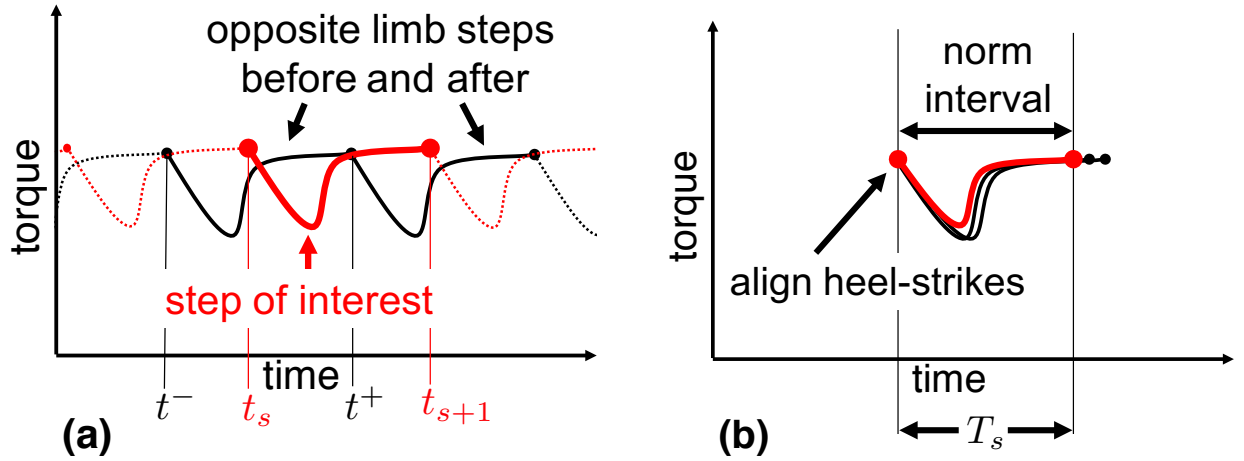


Figure 6.4: Illustrative representation of the asymmetry measure: (a) time series torque data with the step of interest (red) and opposite limb steps (black) and HSs denoted as circles, (b) the opposite limb steps are shifted in time until their HSs align with the step of interest.

and the steps of the opposite limb occurring immediately before and after the step of interest. The procedure is illustrated in Fig. 6.4 and is described in the following. Let t_s denote the time index of the HS that begins the step of interest and t_{s+1} denotes the HS that ends the step of interest, then the step of interests has a period of

$$T_s = t_{s+1} - t_s. \tag{6.1}$$

Let t^- and t^+ denote the time index of the HS that begins the opposite limb's step occurring immediately before and after the step of interest, respectively. Then the AM for step s of some gait variable X (e.g., right ankle torque) can be computed as

$$\mathrm{AM}(X_s) = \frac{1}{2}\|X_s - Y^-\|_2 + \frac{1}{2}\|X_s - Y^+\|_2 \tag{6.2}$$

where X is the time series data of the gait variable of interest (e.g., intact ankle torque), Y is the time series data of the opposite limb (e.g., prosthetic ankle torque) and the vectors

in (6.2) are defined as:

$$\begin{aligned} X_s &= \begin{bmatrix} X(t_s) & \cdots & X(t_{s+1}) \end{bmatrix}^T \\ Y^- &= \begin{bmatrix} Y(t^-) & \cdots & Y(t^- + T_s) \end{bmatrix}^T \\ Y^+ &= \begin{bmatrix} Y(t^+) & \cdots & Y(t^+ + T_s) \end{bmatrix}^T \end{aligned} \tag{6.3}$$

with the 2-norm defined as $\| \cdot \|_2 = \sqrt{(\cdot)^2}$. Note that (6.2) is applied to each step s and the result is a distribution of AMs for each trial, thus it is possible to conduct statistical analysis comparing the AM distributions between trials.

6.1.6 Statistical Analysis

Within each experimental condition, statistical significance between limbs (e.g., prosthetic limb vs. intact limb) was assessed with a two-sampled t-test, and the statistical significance criterion was $p < 0.05$. Similarly, when comparing between experimental conditions, a two-sampled t-test was used to compare results of the same limb (e.g., prescribed prosthetic limb vs. symmetry prosthetic limb), with a statistical significance criterion of $p < 0.05$ Note a two-sampled t-test was used, since it was possible to have a different number of samples within each condition.

6.2 Results

6.2.1 Algorithm and motor parameters

The robot model G_m, defined in (4.4), in (4.15) is taken as the total gain, see (3.35), from the motor current command i_a to the active torque τ_a:

$$G_m = R_T \cdot k_\tau, \tag{6.4}$$

where R_T is the effective transmission, defined in (3.34), and k_τ is the torque constant of the motor. The initial learning gain ρ_0 was initialized as $\rho = 0.5$ and the maximum learning

harmonic $\omega_{max} = 10 \cdot \omega_0$, e.g.,

$$\rho_0(\omega) = \begin{cases} \rho & \text{if } \omega \leq \omega_{max} \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

The filter cutoff, defined in 4.17 was chosen as $\omega_c = 10 \cdot \omega_0$. Prior to filtering, the learned control signal u_{k+1} at each iteration k was eliminated (e.g., set to zero), encoded as t_1 and t_2, defined in 4.16, from $0-5\%$ gait and $\text{TO}+20\%$ (TO = toe off), respectively. The maximum allowable growth factor α, defined in (4.11), was chosen as $\alpha = 2$, and the decay rate γ, defined in (4.12), was chosen as $\gamma = 2$. The frequency dependent padding $\epsilon(\omega)$ in (4.10), is chosen as three times the standard deviation of the magnitude of prosthetic torque during the first learning iteration $k = 0$,

$$\epsilon(\omega) = 3\sigma_M\left(|\tau_{p,0}(\omega)|\right) \tag{6.6}$$

where the standard deviation at each frequencies ω is taken over the total steps M:

$$\sigma_M(|\tau_{p,0}(\omega)|) = \frac{1}{M}\sum_{m=1}^{M}\left(|\tau_{p,0,m}(\omega)| - \mu_M(|\tau_{p,0}(\omega)|)\right)^2, \tag{6.7}$$

$$\mu_M(|\tau_{p,0}(\omega)|) = \frac{1}{M}\sum_{m=1}^{M}|\tau_{p,0,m}(\omega)| \tag{6.8}$$

Table 6.2: Algorithm and motor parameters.

Gearbox transmission	R_G	19.2
Effective transmission	R_T	1800
Torque constant (Nm/A)	k_τ	0.00775
Speed constant (rpm/V)	k_s	1223
Initial gain	ρ	0.5
Pre-smooth time (%)	t_1	10
Post-smooth time (%)	t_2	$\text{TO} + 20\%$
Filter cutoff	ω_c	$10 \cdot \omega_0$
HS delay (ms)	Δ_{HS}	40
Decay rate	γ	2
Allowable reference growth factor	α	2

Note that $\tau_{p,0,m}$ represents the torque at the m$^{\text{th}}$ step during iteration $k = 0$. Table 6.2 summarizes the algorithm and motor parameters used in this study.

6.2.2 Algorithm performance

Time domain

The active torque $\hat{\tau}_{a,k}$, shown in the top plot of Fig. 6.5 and calculated as

$$\hat{\tau}_{a,k} = R_T \cdot k_\tau \cdot u_k \tag{6.9}$$

initially grew, till iteration $k = 6$, then started to shrink towards the first active torque signal $\hat{\tau}_{a,1}$. This can be understood by looking at the peak error e_k, shown in the middle plot of Fig. 6.5, which initially reduced, but then at $k = 6$ began to grow. In addition, note the growth of the error signal e_k during early stance, e.g., $0 - 30\%$. The divergent error e_k was a result of large adaptations in the reference signal, e.g., the intact torque. The bottom plot in Figure 6.5 shows the changes that occurred in the reference signal (intact ankle torque), e.g., $\bar{\tau}_{b,0} - \bar{\tau}_{b,k}$. It can be seen that with each iteration k the reference signal grew, and the growth was concentrated at around 20% gait. Despite the variation in the reference signal, the algorithm was able to avoid divergence of the control signal, apparent from the reduction in $\hat{\tau}_{a,k}$ after iteration $k = 6$. .

Frequency domain

The frequency dependent learning rate ρ_k adapted, as outlined in Sect. 4.3, as the error e_k and the change in reference signal $\bar{\tau}_{b,0} - \bar{\tau}_{b,k}$ increased. To understand the adaptations Fig. 6.6 shows the evolution of each harmonic of the magnitude of error e_k, normalized by the initial error e_0 (left plot), the magnitude of changes in reference signal $\bar{\tau}_{b,0} - \bar{\tau}_{b,k}$, normalized by the initial error e_0 (middle plot), and the learning gain ρ_k, normalized by the initial ρ_0. Initially most error e_k harmonics were reduced (excluding harmonic 10 and 8), and continued reducing till iteration $k = 6$, after which the error e_k plateaued. However, the opposite was true for the

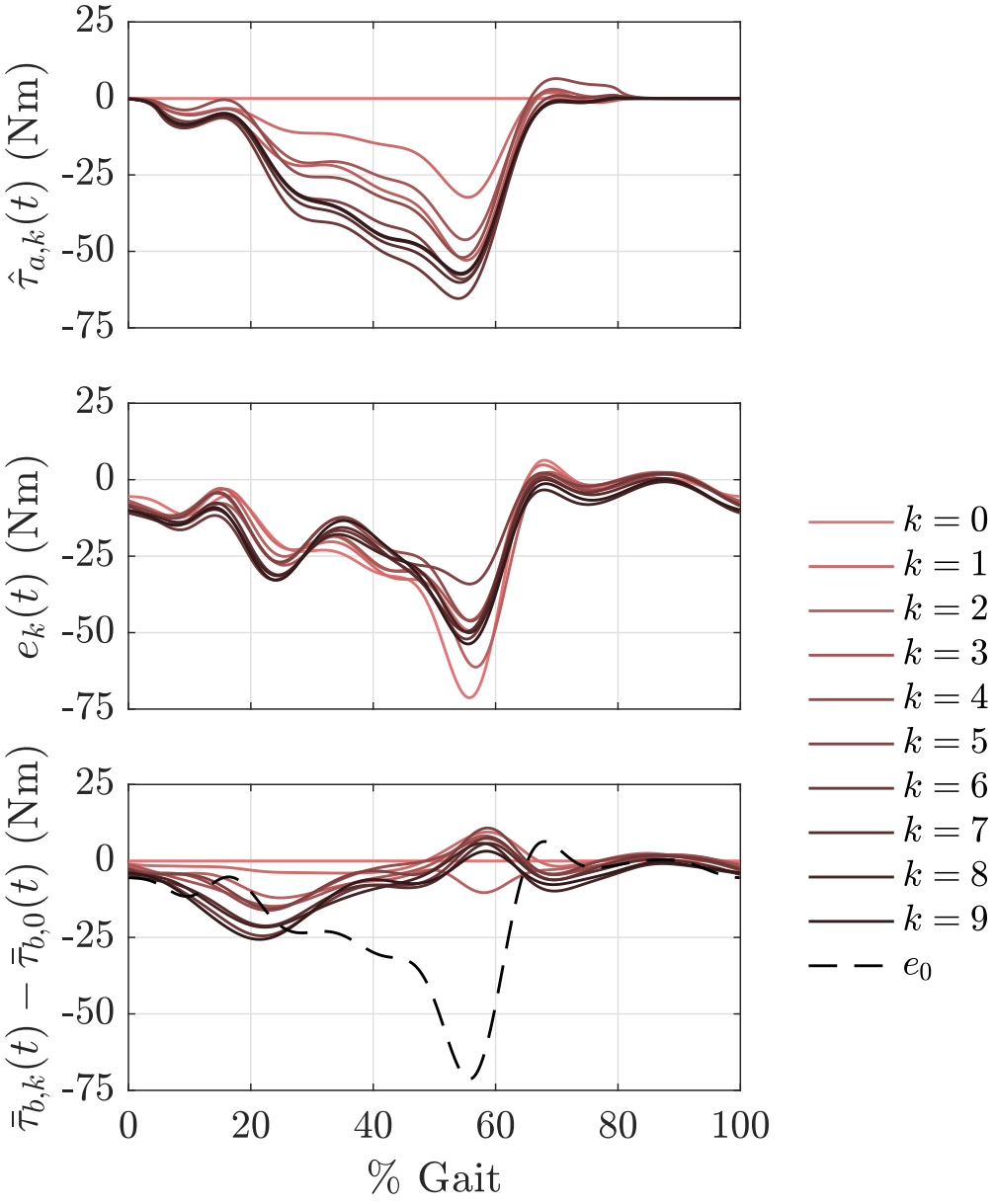


Figure 6.5: Time domain evolution of active torque $\hat{\tau}_{a,k}$, error e_k, and the change in reference signal, e.g., $\bar{\tau}_{b,k} - \bar{\tau}_{b,0}$.

changes is reference signal harmonics, which increased at every iteration up iteration $k = 6$ (notably in the 0-2 harmonics), then again plateauing. The learning gain ρ_k began reducing (e.g., condition 4.12) at iteration $k = 2$ (harmonics 0, 2, 3, 5, 6, 8). Harmonic 3 continued to

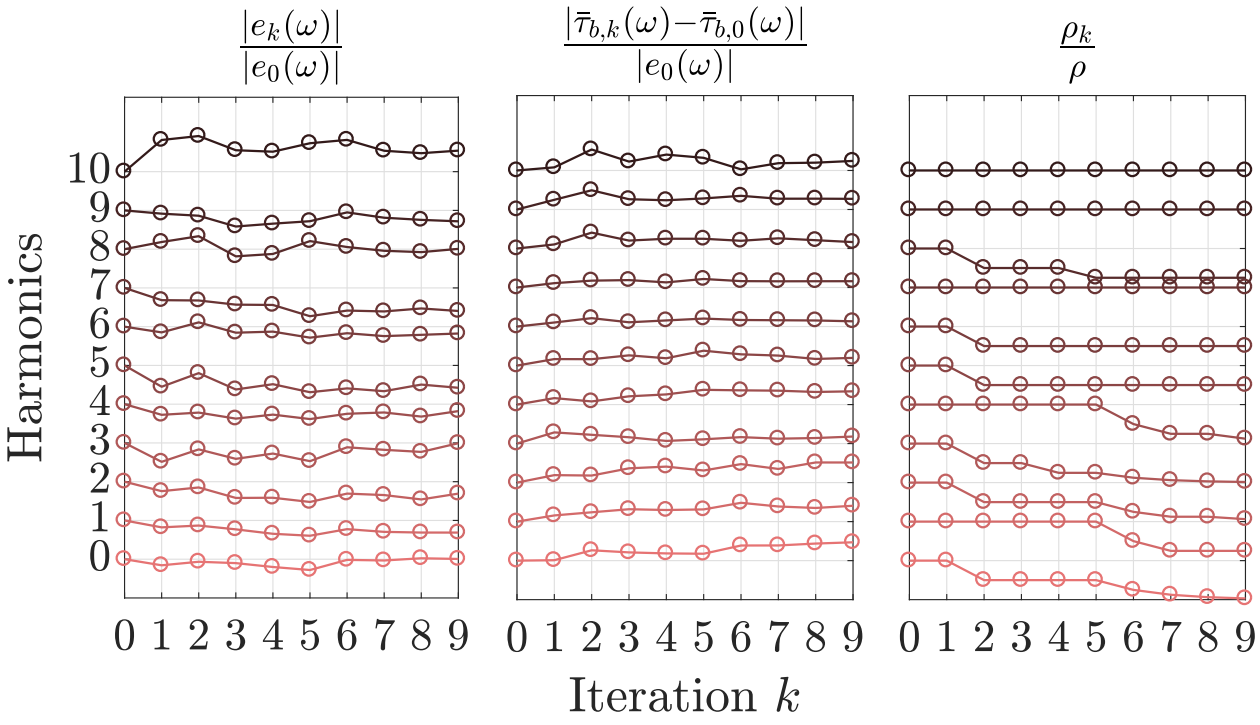


Figure 6.6: Frequency domain evolution of the magnitude of error, normalized by the initial error, $|e_k|/|e_0|$, the magnitude of change in reference signal, normalized by the initial error $|\bar{\tau}_{b,k}-\bar{\tau}_{b,0}|/|e_0|$, and adaptation in ρ_k, normalized by the initial ρ, defined in 6.5.

reduce, effectively removing all new frequency content at that harmonic. The same eventually occurred for harmonics 0, 1, 3, 4, thus, the control signal u_k eventual converged since small learning gains have little affect on the new control signal u_{k+1}. This can be seen in the time domain (Fig. 6.5) as the active torques $\hat{\tau}_{a,k}$ during iteration $k = 8,\ 9$, which have very small differences. These results suggestion the algorithm does work as anticipated, and resulted in a bounded control signal u_k, despite large changes that occurred in the reference signal. From Fig. 6.6 it can be seen that iteration $k = 5$ achieved the lowest error, thereby this iteration will be used in subsequent analysis.

6.2.3 Ankle mechanics

Ankle angle

Both the passive and symmetry conditions had a significant effect on the peak ankle plantarflexion and dorsiflexion of both the intact and prosthetic side, when compared with the prescribed condition. Additionally, the symmetry condition was significantly different than the passive condition. The mean ankle angles for each condition are shown in the top row of Fig 6.7. The most notable changes are as follows. The symmetry condition reduced the peak plantarflexion angle on the prosthetic side from 2.0° during the prescribed condition to 2.3^o in the symmetry condition ($p = 0.005$). Note that the reported angles are in absolute reference, meaning in both cases the ankle never technically plantarflexes (e.g., the angle between the shin and the ankle never became larger than 90°). Additionally, the peak plantarflexion of the prosthetic side during the prescribed condition occurred near heel strike, while during the symmetry condition the peak plantarflexion occurred near toe-off. Plantarflexion on the intact side was not significantly altered. However, dorsiflexion on the prosthetic side was reduced from 24.9° during the prescribed condition to 20.0° during the symmetry condition ($p < 0.000$), and the intact side was reduced from 19.5° during the prescribed condition to

Table 6.3: Joint kinematic outcome measures (standard deviations). Corresponding histograms are provided in F.1

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Plantarflexion (deg)	$1.99(0.19)^a$	$-16.66(1.85)$	$4.13(0.07)^{ab}$	$-14.73(0.93)^b$	$2.30(0.44)^{abc}$	$-16.84(1.07)^c$
Peak Ankle Dorsiflexion (deg)	$24.90(0.13)^a$	$19.52(0.51)$	$24.06(0.17)^{ab}$	$17.77(0.48)^b$	$20.02(0.26)^{abc}$	$17.74(0.59)^b$
Peak Knee Flexion (deg)	$71.34(0.70)^a$	$59.30(1.10)$	$56.00(0.74)^{ab}$	$53.77(0.85)^b$	$57.55(0.69)^{abc}$	$56.29(0.68)^{bc}$
Peak Hip Extension (deg)	$13.29(0.36)^a$	$5.28(0.26)$	$5.46(0.45)^{ab}$	$-0.22(0.31)^b$	$6.27(0.66)^{abc}$	$2.60(0.38)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

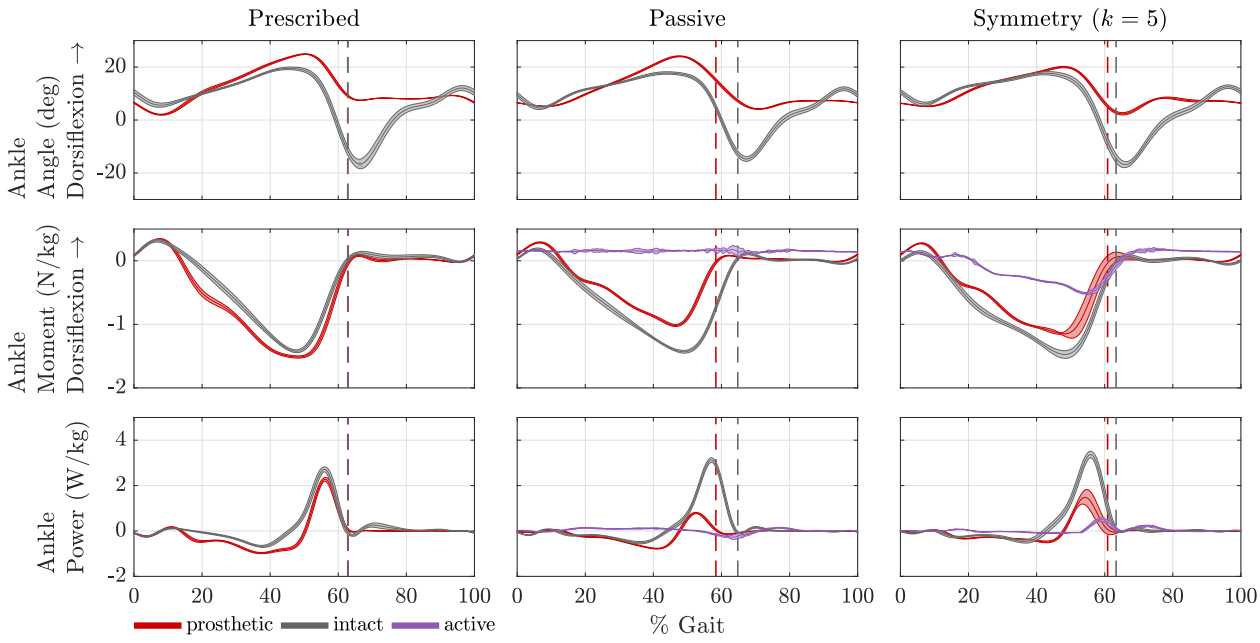


Figure 6.7: Sagittal ankle mechanics for the three experimental conditions. Dashed lines represent corresponding toe offs (during the prescribed condition toe offs nearly coincide). Width of the traces represent ±1 standard deviation.

17.7° during the symmetry condition ($p < 0.000$). Note that increased dorsiflexion on the intact side is a common observation in below-knee amputees [6]. Ankle angle asymmetry, for both intact and prosthetic side, during the prescribed condition, computed as in (6.2), was significantly reduced when compared with both the passive condition ($p < 0.000$) and the symmetry condition ($p = 0.000$), however, significance was not achieved when comparing the passive and symmetry conditions ($p = 0.9$). Ankle angle outcomes are compiled in Table 6.3 and asymmetry outcomes for the ankle angle are recorded in Table 6.7.

Ankle moment

Ankle moment, shown in the middle row of Fig. 6.7, was significantly altered during the passive and symmetry conditions. During the prescribed condition the prosthetic side ankle moment was significantly larger than the intact ankle peak moment (−1.52 (Nm/kg) for

prosthetic side and -1.42 (Nm/kg) for intact side ($p < 0.000$)), shown in the left column middle row in Fig. 6.7. The prosthetic side peak ankle moment was largely reduced during the passive condition (middle column middle row in Fig. 6.7), from -1.52 (Nm/kg) to -1.03 (Nm/kg) ($p < 0.000$), while the intact side showed no significant difference ($p = 0.242$). Prosthetic side peak ankle moments reduced from -1.52 (Nm/kg) during the prescribed condition to -1.19 (Nm/kg) during the symmetry condition ($p < 0.000$), while intact side increased from -1.42 (Nm/kg) to -1.48 (Nm/kg) ($p < 0.000$). Note that the intact moment during the prescribed condition didn't significantly differ from intact moment during the passive condition, confirming that the added active torque significantly alters the intact moment e.g., as described in Sect. 6.2.2, adaptations occurred on the intact side during learning. During the symmetry condition, the prosthetic side ankle peak moment increased when compared to the passive condition, however, large variations in the ankle moment during the symmetry condition are apparent (right column middle row in Fig. 6.7) by the thickness of the trace (± 1 standard deviation), and were caused by small changes in toe off timing. Note the peak active torque does not correspond to the peak prosthetic side ankle moment, which means the subject was not receiving the full benefit of the active torque. If the subject "caught" the push-off, then the subject achieved the largest amount of ankle moment. These strides are represented by the standard deviation limits closest to that of the intact moment curve (Fig. 6.7). The timing discrepancy is an inherent limitation of the error e_k calculation: by examining the passive condition (center plot Fig. 6.7), which is used to calculate the initial error e_0, it can be seen that a substantial portion of the error occurred very close to toe off, thus, the peak error was nearly aligned with toe off. However, the peak intact ankle moment occurred prior to toe off resulting in the peak torque being delivered later than appropriate. Asymmetries in ankle moment, computed as in (6.2), significantly increased for the prosthetic and intact sides when comparing the prescribed and symmetry condition ($p = 0.03$, $p = 0.04$ for prescribed and intact respectively). However, when comparing the passive and symmetry conditions, a large reduction in asymmetry was observed in both the prosthetic and intact sides ($p < 0.000$ for both). Ankle moment asymmetry outcomes are

recorded in Table 6.7 and peak ankle moment outcomes are recorded in Table 6.4.

Ankle power

Ankle power, shown in the bottom row Fig. 6.7 for all conditions, was significantly altered during the passive and symmetry conditions. The most apparent modification was a significant increase in intact peak positive power, from 2.7 (W/kg) during the prescribed condition, to 3.13 (W/kg) during the passive condition, to 3.38 (W/kg) during the symmetry condition. Negative power also significantly decreased in each subsequent condition. In terms of asymmetry, similar findings of ankle angle moments were observed: asymmetry was significantly reduced ($p < 0.000$ for both intact and prosthetic side), when comparing the passive and symmetry conditions, but significantly increased when comparing the prescribed and symmetry conditions. Table 6.6 contains the peak ankle power outcomes and Table 6.7 contains the ankle power asymmetry outcomes.

Table 6.4: Joint kinetics and vGRF outcome measures (standard deviations). Corresponding histograms are provided in F.2

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Mom. (Nm/kg)	$-1.52(0.02)^{a}$	$-1.42(0.02)$	$-1.02(0.02)^{ab}$	$-1.43(0.02)$	$-1.19(0.08)^{abc}$	$-1.48(0.06)^{bc}$
Peak Knee Flex. Mom. (Nm/kg)	$0.32(0.01)^{a}$	$0.28(0.02)$	$0.21(0.01)^{ab}$	$0.33(0.02)^{b}$	$0.22(0.01)^{abc}$	$0.30(0.02)^{bc}$
Peak Knee Ext. Mom. (Nm/kg)	$-0.70(0.02)^{a}$	$-1.31(0.06)$	$-0.70(0.03)^{a}$	$-1.12(0.04)^{b}$	$-0.65(0.02)^{abc}$	$-1.04(0.06)^{bc}$
Peak Hip Flex. Mom. (Nm/kg)	$0.45(0.02)^{a}$	$0.53(0.05)$	$0.70(0.03)^{ab}$	$0.75(0.05)^{b}$	$0.61(0.04)^{bc}$	$0.65(0.13)^{bc}$
Peak Hip Ext. Mom. (Nm/kg)	$-0.59(0.03)^{a}$	$-0.38(0.05)$	$-0.33(0.01)^{ab}$	$-0.57(0.03)^{b}$	$-0.36(0.02)^{abc}$	$-0.51(0.03)^{bc}$
Peak vGRF (BW)	$1.05(0.01)^{a}$	$1.02(0.02)$	$0.99(0.01)^{ab}$	$1.08(0.01)^{b}$	$1.01(0.02)^{abc}$	$1.04(0.02)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

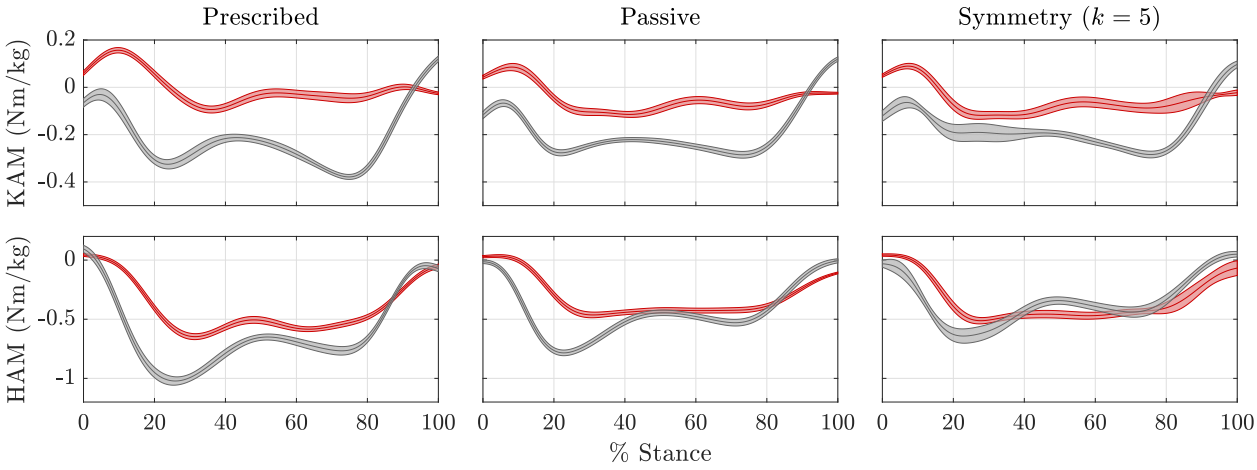


Figure 6.8: Frontal plane hip moments (KAM and HAM), during stance, for the three experimental conditions. Width of the traces represent ±1 standard deviation.

6.2.4 OA loading factors: KAM and HAM

The peak KAM and HAM are shown in Fig. 6.8. The intact peak KAM was reduced from −0.38 (Nm/kg) during the prescribed condition to −0.29 (Nm/kg) during the passive condition ($p = 0.000$). However, comparing the passive and the symmetry condition result in marginal improvements not reaching significance. A larger effect was apparent for the peak HAM, where the prosthetic side reduce from −0.65 (Nm/kg) during the prescribed condi-

Table 6.5: OA loading factors outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak KAM (Nm/kg)	$-0.09(0.02)^{a}$	$-0.38(0.01)$	$-0.12(0.01)^{ab}$	$-0.29(0.01)^{b}$	$-0.12(0.02)^{ab}$	$-0.28(0.01)^{b}$
Peak HAM (Nm/kg)	$-0.65(0.03)^{a}$	$-1.03(0.03)$	$-0.47(0.02)^{ab}$	$-0.79(0.02)^{b}$	$-0.51(0.03)^{abc}$	$-0.65(0.05)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

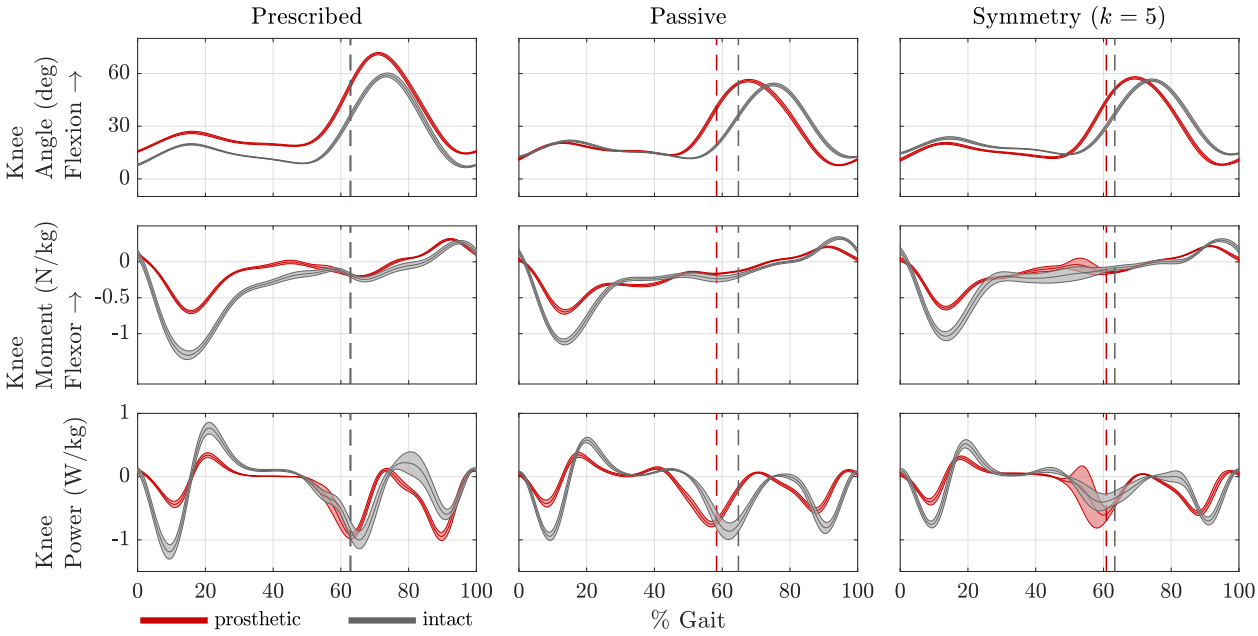


Figure 6.9: Sagittal plane knee mechanics for the three experimental conditions. Dashed lines represent corresponding toe offs (during the prescribed condition toe offs lines nearly coincide). Width of the traces represent ±1 standard deviation.

tion, to −0.47 (Nm/kg) during the passive condition, and −0.51 (Nm/kg) in the symmetry condition. Finally, the largest effect occurred for the intact peak HAM, which was reduced from −1.03 (Nm/kg) during the prescribed condition to −0.65 (Nm/kg) during the symmetry condition ($p = 0.000$). It should be note that although the peak magnitude of the KAM is important, its thought that the initial peak (e.g., the first hump in Fig. 6.8) is more important, due to usually being larger in magnitude and a steeper rate of rise [23, 64, 98]. By examining Fig. 6.8, in can be seen that during the symmetry condition, the first KAM peak is largely reduced, suggesting a beneficial effect during the symmetry condition. KAM and HAM outcome measures are recorded in Table 6.5.

6.2.5 Knee mechanics

Knee angle

Knee angle, shown in the top row of Fig. 6.9, was significantly alter during the passive and symmetry condition when compared with the prescribed condition. A significant effect was observed in the peak knee flexion angle of the prosthetic and intact sides. For the prosthetic side, the peak knee flexion decreased from 24.9° during the prescribed condition, to 20.0° during the symmetry condition ($p < 0.000$). Additionally, peak intact knee flexion was reduced from 19.5° during the prescribed condition to 17.7° during the symmetry condition ($p < 0.000$), which was not significantly different than the passive condition. Note that below-knee amputees have been observed to have increased intact and prosthetic side knee flexion [6]. Indeed, the large flexion angles are apparent during the prescribed condition in Fig. 6.9. However, knee angle asymmetry during the prescribed condition was significantly reduced when compared to the passive condition for both intact and prosthetic sides ($p < 0.000$, for both intact and prosthetic sides), and an even larger reduction when compared to the symmetry condition ($p < 0.000$, for both intact and prosthetic sides). In particular, com-

Table 6.6: Joint power outcome measures (standard deviations). Corresponding histograms are provided in F.3

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Positive Ankle Power (W/kg)	$2.29(0.09)^{a}$	2.74(0.11)	$0.79(0.02)^{ab}$	$3.13(0.10)^{b}$	$1.51(0.36)^{abc}$	$3.38(0.15)^{bc}$
Peak Negative Ankle Power (W/kg)	$0.98(0.02)^{a}$	0.68(0.04)	$0.78(0.02)^{ab}$	$0.52(0.05)^{b}$	$0.49(0.04)^{bc}$	$0.48(0.06)^{bc}$
Peak Positive Knee Power (W/kg)	$0.34(0.04)^{a}$	0.77(0.09)	$0.35(0.04)^{a}$	$0.58(0.04)^{b}$	$0.35(0.16)^{a}$	$0.53(0.07)^{bc}$
Peak Negative Knee Power (W/kg)	$0.97(0.06)^{a}$	1.22(0.11)	$0.76(0.04)^{ab}$	$0.97(0.07)^{b}$	$0.66(0.04)^{abc}$	$0.77(0.05)^{bc}$
Peak Positive Hip Power (W/kg)	$1.11(0.05)^{a}$	1.22(0.13)	$0.77(0.04)^{ab}$	$0.93(0.07)^{b}$	$0.73(0.05)^{abc}$	$0.63(0.08)^{bc}$
Peak Negative Hip Power (W/kg)	$0.39(0.03)^{a}$	0.69(0.10)	$0.99(0.05)^{b}$	$0.96(0.06)^{b}$	$0.72(0.06)^{abc}$	$0.92(0.18)^{b}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

paring the prescribed and symmetry conditions (left and right column top row in Fig. 6.9), prosthetic side knee angle during stance was shifted towards extension and nearly overlapped the intact side knee angle, and the peak knee flexion angle of the intact and prosthetic sides were closer in magnitude. Knee angle outcomes are recorded in Table 6.3 and knee ankle asymmetry measures are recorded in Table 6.7.

Knee moments

Knee moments are shown in the middle row of Fig. 6.9 for all conditions. The symmetry condition resulted in a significant reduction in the peak intact knee extension moment, when compared to the prescribed condition, from -1.3 (Nm/kg) to -1.0 ($p < 0.000$), which occurs early in stance. Prosthetic side peak extension moment during the prescribed condition was also significantly reduce when compared to the symmetry condition, from -0.7 (Nm/kg) to -0.6 (Nm/kg). In addition, prosthetic side knee flexion moment during the prescribed condition was reduced from 0.3 (Nm/kg) to 0.2 (Nm/kg) during the symmetry condition ($p < 0.000$). Lastly, asymmetry, calculated from (6.2), was significantly reduced when comparing the prescribed condition to the to symmetry condition ($p = 0.033$, $p = 0.043$, for prescribed and intact, respectively). This is clearly seen in the middle row of Fig. 6.9. However, asymmetry didn't reach significance when comparing the passive and symmetry conditions ($p = 0.763$, $p = 0.737$, for prosthetic and intact, respectively). Symmetry outcomes are recorded in Table 6.7 and peak knee moment outcomes are recorded in Table 6.4.

Knee power

Out all joint variables, knee power was altered the most by the symmetry condition, which is shown in the bottom row of Fig. 6.9. Intact side positive knee power during the prescribed condition was significantly reduced from 0.8 (W/kg) to 0.5 (W/kg) during the symmetry condition ($p = 0.000$). A small increase in prosthetic side positive power was observed, but didn't reach significance ($p = 0.626$). Intact negative peak power during the prescribed condition was reduced from 1.2 (W/kg) to 0.77 (W/kg). The large effects are apparent from

examining the bottom row of Fig. 6.9. Note the large reduction of intact negative power near toe off. Additionally, asymmetry was greatly reduced during the prescribed condition, when compared to the passive condition, which was subsequently reduced during the symmetry condition, all of which were significant at $p < 0.000$ levels. These results suggest than the symmetry control had a large effect on knee mechanics. Knee power asymmetry outcomes are recorded in Table 6.7 and peak knee power outcomes are recorded in Table 6.6.

6.2.6 Hip mechanics

Hip angle

The hip angle, shown for all three condition in Fig. 6.10, was significantly altered from the prescribed condition when compared to both the passive and symmetry conditions. The peak hip angle significantly decreased, towards extension, when comparing the prescribed and symmetry conditions for both intact and prosthetics sides. In addition, asymmetry,

Table 6.7: Sagittal plane AM outcome measures (standard deviations), defined in (6.2).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
AM GRF (BW)	2.42(0.37)	2.43(0.38)	$5.46(0.14)^{a\,b}$	$6.15(0.29)^{b}$	$3.47(0.63)^{b\,c}$	$3.56(0.69)^{b\,c}$
AM Ankle Angle (rad)	1.93(0.10)	1.93(0.13)	$1.63(0.06)^{b}$	$1.63(0.07)^{b}$	$1.64(0.07)^{b}$	$1.64(0.09)^{b}$
AM Ankle Moment (Nm/kg)	2.29(0.21)	2.28(0.26)	$3.64(0.12)^{b}$	$3.63(0.18)^{b}$	$2.54(0.48)^{b\,c}$	$2.54(0.49)^{b\,c}$
AM Ankle Power (W/kg)	4.38(0.43)	4.39(0.46)	$8.77(0.20)^{b}$	$8.77(0.26)^{b}$	$7.36(1.33)^{b\,c}$	$7.36(1.19)^{b\,c}$
AM Knee Angle (rad)	2.19(0.12)	2.18(0.12)	$1.95(0.18)^{b}$	$1.95(0.17)^{b}$	$1.54(0.16)^{b\,c}$	$1.53(0.20)^{b\,c}$
AM Knee Moment (Nm/kg)	3.88(0.24)	3.88(0.30)	$2.17(0.15)^{b}$	$2.17(0.18)^{b}$	$2.14(0.37)^{b}$	$2.14(0.38)^{b}$
AM Knee Power (W/kg)	4.16(0.41)	4.18(0.57)	$3.67(0.33)^{b}$	$3.65(0.34)^{b}$	$2.66(0.64)^{b\,c}$	$2.66(0.49)^{b\,c}$
AM Hip Angle (rad)	1.29(0.08)	1.28(0.08)	$1.58(0.11)^{b}$	$1.57(0.12)^{b}$	$1.08(0.12)^{b\,c}$	$1.07(0.15)^{b\,c}$
AM Hip Moment (Nm/kg)	4.50(0.31)	4.50(0.39)	$2.51(0.13)^{b}$	$2.51(0.15)^{b}$	$2.28(0.34)^{b\,c}$	$2.27(0.35)^{b\,c}$
AM Hip Power (W/kg)	5.50(0.42)	5.49(0.54)	$3.96(0.34)^{b}$	$3.95(0.31)^{b}$	$3.06(0.39)^{b\,c}$	$3.05(0.43)^{b\,c}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

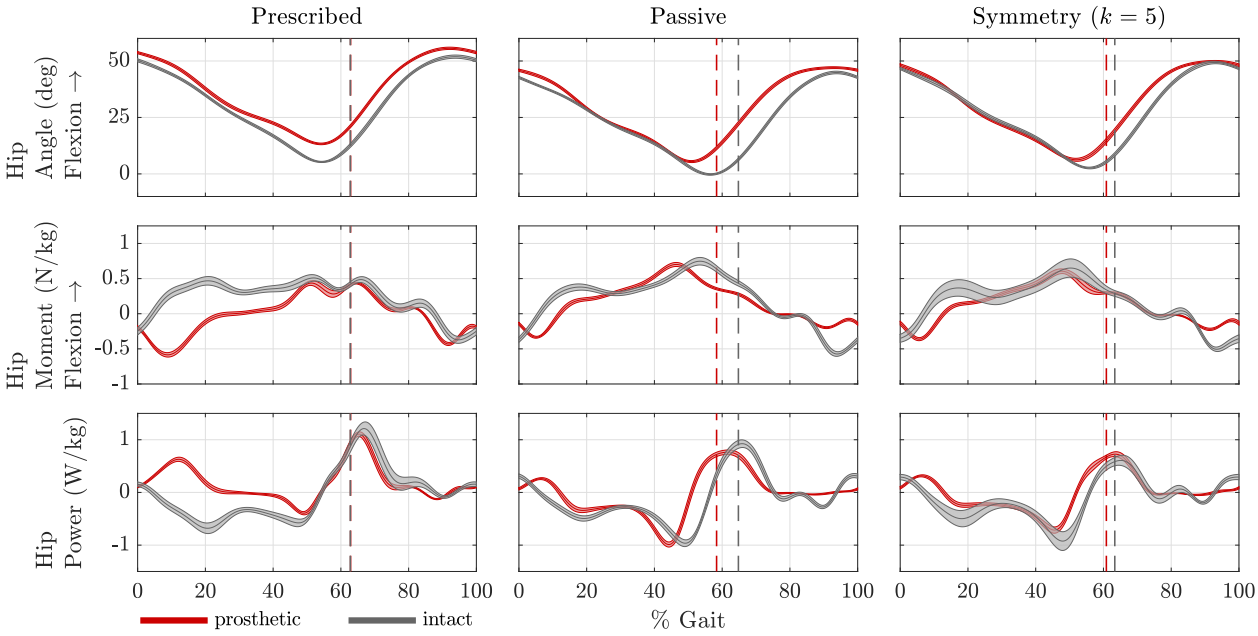


Figure 6.10: Sagittal plane hip mechanics for the three experimental conditions. Dashed lines represent corresponding toe offs (during the prescribed condition toe offs nearly coincide). Width of the traces represent ±1 standard deviation.

calculated as in (6.2), significantly decreased during the prescribed condition when compared with the symmetry condition for both intact and prosthetic sides ($p < 0.000$ for both), and can be seen by comparing the middle row of Fig. 6.10. Note early in stance, the hip angle for the intact and prosthetic sides nearly overlapped and the peak hip angles are closer in magnitude during the symmetry condition, when compared with the prescribed condition. Hip angle asymmetry outcomes are recorded in Table 6.7 and peak hip angle outcomes are recorded in Table 6.3.

6.2.7 Hip moment

The hip moments, shown for all conditions in the middle row of Fig. 6.10, was altered substantially. Qualitatively, in can be seen that the prosthetic side moment early in stance nearly reverses from extension moments to flexion moments, for both the passive and symmetry con-

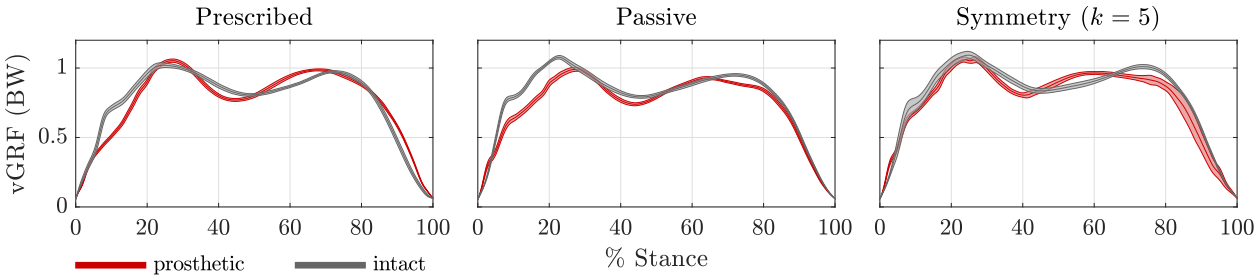


Figure 6.11: Stance phase vertical ground reaction force (vGRF), normalized by body weight, for the three experimental conditions. Width of the traces represent ±1 standard deviation.

ditions when compared to the prescribed condition. Peak hip flexion moment increased for both intact and prosthetic sides, however, asymmetry, computed as in (6.2), was significantly reduced when comparing the prescribed condition to the passive and symmetry conditions, for both intact and prosthetic sides. Asymmetry comparison didn't reach significane when comparing the passive and symmetry conditions. Hip moment asymmetry outcomes are recorded in Table 6.7 and peak hip moment outcomes are recorded in Table 6.4.

6.2.8 Hip power

Similar to the knee power adaptations during symmetry control, the hip power exhibit substantial alterations. Most notable is the reversal in power, from positive to negative, that occurred early in stance, and can be seen in the bottom row of Fig. 6.10. Intact side peak positive hip power, which coincided with toe off, was significantly reduce from 1.2 (W/kg) in the prescribed condition, to 0.6 (W/kg) during the symmetry condition ($p < 0.000$). Similarly, prosthetic side peak positive hip power was significantly reduce from 1.1 (W/kg) during the prescribed condition to 0.7 (W/kg) during the symmetry condition ($p < 0.000$). On the other hand, negative power significantly increased for intact side, 0.7 (W/kg) during the prescribed condition to 0.9 (W/kg) during the symmetry condition ($p < 0.000$). A similar effect was observed for the prosthetic side. Lastly, hip power asymmetry was significantly reduce for both intact and prosthetic sides, when comparing the prescribed and symmetry

conditions ($p < 0.000$ for both). Hip power asymmetry outcomes are recorded in Table 6.7 and peak hip power outcomes are recorded in Table 6.6.

6.2.9 Vertical ground reaction force

The vGRF, shown during the stance phase in 6.11, was significantly affected by the passive and symmetry conditions when compared to the prescribed condition. Peak vGRF on the prosthetic side decreased, from 1.05 (BW) during the prescribed condition to 1.01 (BW) during the symmetry condition ($p = 0.000$), while intact side peak vGRF increased, from 1.02 (BW) to during the prescribed condition to 1.04 (BW) during the symmetry condition ($p = 0.042$). Asymmetry, computed as in (6.2) over the stance phase, increased significantly ($p = 0.000$ for both intact and prosthetic sides). Asymmetry for vGRFs are recorded in Table 6.7 and peak vGRF outcomes are recorded in Table 6.4.

6.2.10 Temporal symmetry

The period of stance changed significantly when comparing the prescribed, passive and symmetry conditions. During the prescribed condition the period of stance of the prosthetic and intact sides showed no significant difference, 62.9% and 62.8%, respectively ($p = 0.446$). On the other hand, the passive condition resulted in a reduced prosthetic side stance (58.4% ($p < 0.000$)) and an increased intact stance (64.8% ($p < 0.000$)). The symmetry condition

Table 6.8: Temporal symmetry outcomes measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Step Period (s)	1.30(0.01)		1.24(0.01)^{b}		1.29(0.01)^{bc}	
Stance	62.87(0.28)	62.78(0.45)	58.35(0.33)^{ab}	64.79(0.23)^{b}	60.86(1.59)^{abc}	63.34(0.35)^{bc}

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

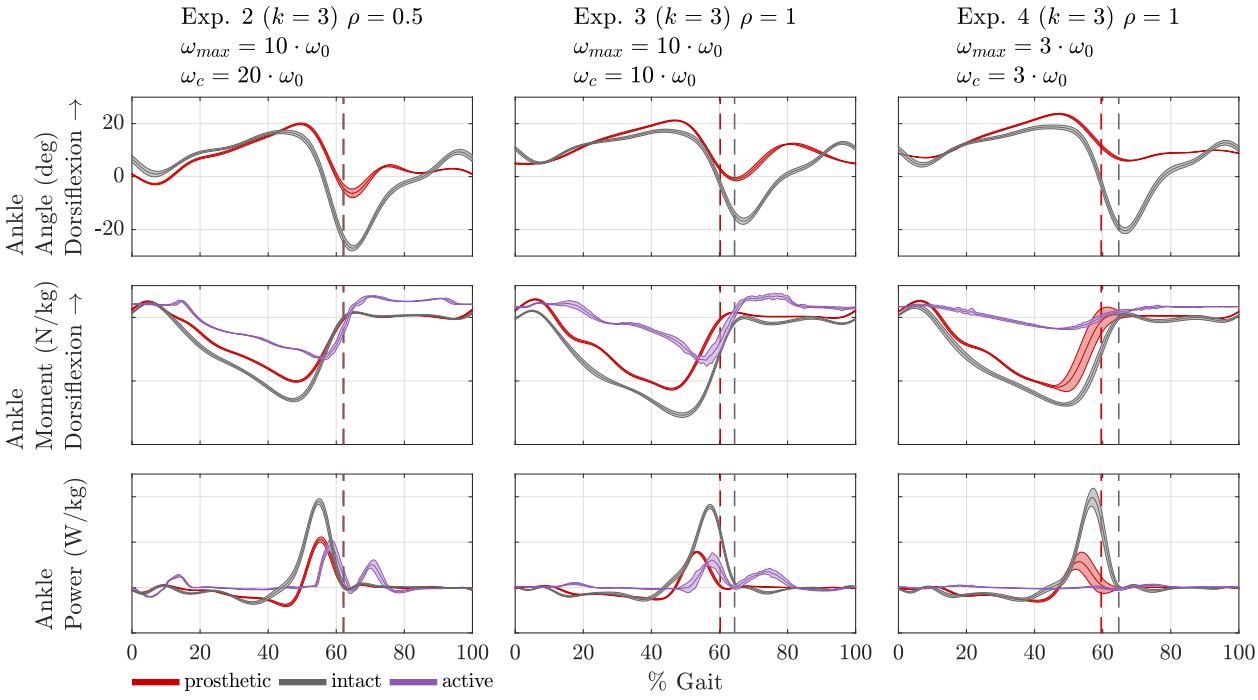


Figure 6.12: Selected results from different parameter settings. Full experimental results for each experiment are in included in Appendix G-K.

subsequently decreased intact side stance (63.3% ($p < 0.000$)) and increased prosthetic side stance (60.9% ($p < 0.000$)), yet not to the original levels observed during the prescribed condition. Similar observations occurred for the step period: a decrease from 1.30 (s) during the prescribed condition, to 1.24 (s) during the passive condition ($p < 0.000$) and 1.29 (s) ($p < 0.000$) during the symmetry condition. The temporal outcomes are recorded in Table 6.8.

6.2.11 Alternative algorithm parameters

Figure 6.12 shows ankle mechanics results with different parameters, e.g., different learning gain ρ, and maximum learning harmonics ω_{max}. Note that in Experiment 2 (left column) time symmetry was achieved, e.g., stance times coincided evident from the overlapping toe off lines. Experiment 2 was an early experiment, with a different motor configuration, detailed

in Appendix G. However, the motor rating was slightly to small, and thus would saturate with any larger torques than the ones shown in Fig. 6.12. Experiment 2 shared the same parameters as Experiment 1 (e.g., the one detailed above), with the only difference $\rho = 1$. It can be seen that with this configuration, the reference signal grew rapidly (see Appendix H). Note the deviation in intact ankle moment early in stance. In Experiment 4 the maximum learning harmonic was reduced to $\omega_{max} = 3$. Again, this largely affected the intact ankle response, evident from the large intact power. It should be noted, to a large degree, most increased symmetry at the knee and hip, and do varying degrees at the ankle. A total of 6 experiments were conducted, each with different control parameters. All other experimental results, which in general support the same conclusions, can be found in Appendix G-K.

6.3 Summary

The hypotheses were: the symmetry control method would result in (1) a reduction in ankle kinetics and kinematic asymmetries, and (2) a reduction in peak knee and hip adduction moments. Additionally, hip and knee mechanics, vGRF and temporal outcomes were statistically analyzed. Hypotheses (1) was partially supported: asymmetries in ankle moment and ankle power were reduced, when comparing the passive and symmetry conditions. Hypothesis (2) was partially supported, with large reductions in KAM and HAM, however, statistical significance between the passive and symmetry condition for KAM weren't achieved. It should be noted that the first peak of the KAM was greatly diminished during the symmetry condition when compared with the passive condition. Additionally, secondary outcomes indicate significant hip and knee adaptations. The conclusions are summarized as follows:

6.3.1 Primary outcomes

- **avoiding divergent control signal growth**

  Despite large variations in the intact torque (e.g., the reference signal), the AB-ILC algorithm maintained boundedness. This can be seen in Fig. 6.5, which shows time domain adaptations and Fig. 6.6, adaptations in the frequency domain. The frequency

dependent learning gait ρ_k adapted in the error e_k increases, well as large changes in the reference signal (intact ankle moment), therefore, the approach performed as intended.

- **ankle moment asymmetry**

  (**H1.1**) The symmetry controller resulted in a reduction of asymmetries in ankle moment, when compared to the passive condition. ($p < 0.000$), which support the hypothesis (middle row in Fig. 6.7 Table 6.7).

- **ankle angle asymmetry**

  (**H1.2**) The symmetry controller resulted in insignificant changes in in ankle angle asymmetries, when compared to the passive condition, which rejects the hypothesis (top row in Fig. 6.7 Table 6.7).

- **ankle power asymmetry**

  (**H1.3**) The symmetry controller resulted in a reduction of asymmetries in ankle power, when compared to the passive condition, supporting the hypothesis ($p < 0.000$) (bottom row Fig. 6.7 Table 6.7).

- **peak KAM**

  (**H2.1**) The symmetry controller resulted in insignificant changes in peak intact KAM, when compared to the passive condition, which rejects the hypothesis (top row in Fig. 6.8, Table 6.5).

- **peak HAM**

  (**H2.1**) The symmetry controller resulted in a reduction of peak intact HAM (bottom row in Fig. 6.8, Table 6.5), from -0.79 (Nm/kg) during the passive condition to -0.65 (Nm/kg) during the symmetry condition ($p < 0.000$), supporting the hypothesis.

6.3.2 Secondary outcomes

- **ankle mechanics adaptations**

  The symmetry controller resulted in: (i) a reduction in peak dorsiflexion on the pros-

thetic side (top row in Fig. 6.7, Table 6.3) from 24.9° during the prescribed condition to 20.0° during the symmetry condition ($p < 0.000$); (ii) a reduction in prosthetic side peak ankle moment and an increase in intact side peak ankle moment (middle row in Fig. 6.7, Table 6.4), from 1.52 (Nm/kg) for the prosthetic side during the prescribed condition to 1.19 (Nm/kg) for the prosthetic side during the symmetry condition, and 1.42 (Nm/kg) for the intact side during the prescribed condition to 1.48 (Nm/kg) for the intact side during the symmetry condition ($p < 0.000$ for all); (iii) a reduction in prosthetic side peak ankle power and an increase in intact side peak ankle power (bottom row in Fig. 6.7, Table 6.6), from 2.7 (W/kg) for the intact side during the prescribed condition to 3.38 (W/kg) for the intact side during the symmetry condition, and from 2.29 (W/kg) for the prosthetic side during the prescribed condition to 1.51 (W/kg) for the prosthetic side during the symmetry condition ($p < 0.000$ for all). Asymmetry was increased when comparing the prescribed and symmetry conditions (see Table 6.7).

- **knee mechanics adaptations**

  The symmetry controller resulted in: (i) reduced peak intact knee flexion angle (top row in Fig. 6.9, Table 6.3), from 24.9° during the prescribed condition, to 20.0° during the symmetry condition ($p < 0.000$); (ii) reduced intact knee extension moment (middle row in Fig. 6.9, Table 6.4), from 1.3 (Nm/kg) during the prescribed condition to 1.0 (Nm/kg) during the symmetry condition ($p < 0.000$); (iii) reduced intact positive knee power (bottom row in Fig. 6.9, Table 6.6), from 0.8 (W/kg) during the prescribed condition, to 0.5 (W/kg) during the symmetry condition ($p < 0.000$) and reduced intact negative peak knee power, 1.2 (W/kg) during the prescribed condition to 0.77 (W/kg) during the symmetry condition. Additionally, asymmetries at the knee were significantly reduced (see Table 6.7). The adaptations outlined above, suggest that the symmetry control significantly affected knee joint behavior.

- **hip mechanics adaptations**

The symmetry controller resulted in: (i) a decrease in peak intact and prosthetic side hip angle, shifting towards extension, (top row in Fig. 6.10, Table 6.3), from 13.3° for the prosthetic side during the prescribed condition to 6.3° for the prosthetic side during the symmetry condition, and from 5.3° for the intact side during the prescribed condition to 2.6° for the intact side during the symmetry condition; (ii) an increase in hip flexion moment for the both the prosthetic and intact sides (middle row in Fig. 6.10, Table 6.3), from 0.45 (Nm/kg) for the prosthetic side during the prescribed condition to 0.61 (Nm/kg) for the prosthetic side during the symmetry condition, and from 0.53 (Nm/kg) for the intact side during the prescribed condition to 0.65 (Nm/kg) for the intact side during the symmetry condition; (iii) a decrease in peak hip extension moment (magnitude) for the prosthetic side, from -0.59 (Nm/kg) during the prescribed condition to to -0.36 (Nm/kg) during the symmetry condition, and an increase in peak hip extension moment on the intact side, from -0.38 (Nm/kg) during the prescribed condition to to -0.51 (Nm/kg) during the symmetry condition; (iv) a decrease in peak positive hip power for both prosthetic and intact sides, (bottom row in Fig. 6.10, Table 6.6), from 1.11 (W/kg) for the prosthetic side during the prescribed condition to 0.73 (W/kg) for the prosthetic side during the symmetry condition, and 1.22 (W/kg) for the intact side during prescribed condition to 0.63 (W/kg) for the intact side during the symmetry condition ($p < 0.000$ for all hip statistics listed above). Additionally, asymmetries at the hip were largely reduced (see Table 6.7).

- **KAM and HAM adaptations** The symmetry controller substantially altered the KAM and HAM, as follows: (i) qualitatively, the first KAM peak was largely reduced (top row in Fig. 6.8); (ii) peak intact HAM was reduced (bottom row in Fig. 6.8), from -1.03 (Nm/kg) during the prescribed condition to -0.65 (Nm/kg) in the symmetry condition ($p < 0.000$), which provides some evidence of clinical benefits.

6.4 Discussion

6.4.1 Joint mechanics

All lower limb joints were affected by the symmetry learning controller. Although originally targeting ankle torque asymmetries, the most significant adaptations were observed in the knee and hip. This suggests that lower limb mechanics are strongly coupled: affecting the ankle had a large effect on knee and hip. This is as expected, since the, knee and hip compensate for the loss of ankle function, and by increasing the prosthetic ankle moment, e.g., though active compensation, towards that of the intact limb, the knee and hip compensations were reduced, as indicated by the results.

6.4.2 The effect of human adaptations

The intact ankle moment adapted significantly during learning. In particular, as the active ankle torque increased, so did the intact ankle moment (Fig. 6.5). It is possible that not enough time was given, between each learning iteration, in order for the subject to become comfortable with the new device behavior. This view is supported by the apparent large variation in joint mechanics during the symmetry condition (left column of Fig. 6.7-6.10). A longer time frame of acclimation, for each learning trial, could allow the subject to adapt more adequately.

6.4.3 Algorithm limitations

The major limitations of the symmetry learning control are as follows. The algorithm can not positively adapt to large variations in the reference signal, but instead backsteps and reduces the learning gains, thus eliminating the potential for more learning. A better approach would incorporate the new reference adaptations. For example, by building a dynamic model using the new data [87]. In addition, a time based approach, e.g., encoding the control signal as a function in time, has limitations. A time based approach isn't robust to small perturbations, e.g., step period. This was observed, and can be seen in the large variations in the ankle

moment (Fig. 6.7). If the user was able to "catch" the peak moment the received substantially more push-off. A phase based approach, where a time-invariant variable is chosen to encode gait cycle [99, 100], could address this limitation. Learning could be done in the "phase" domain, resulting in a time invariant control law. Another approach is to learn the signal in the time or frequency domain, e.g., as proposed in this work, then encode the learned torque-angle relationship as an impedance control law [35].

Chapter 7
FUTURE WORK AND CONCLUSIONS

7.1 Future work

7.1.1 Mechanical design and modeling

Improvements to the mechanical design of the prototype PAFP are possible. For example, including the motor dynamics in the optimization routine, outline in Ch. 3, could further reduce the active demand. Reduction in overall device mass can be achieved by using composite materials, including the cam spring [101]. More complex cam shapes are possible, e.g., cams that encode multiple trajectories and can be switch automatically. Lastly, a more through investigation into tailoring the passive response and the resulting effects on joint mechanics, would allow for a more informed procedure for customization.

7.1.2 Embedded system

Improvements to the heel-strike detection are possible, for example, by utilizing more FSR sensors. This would address issues of false heel-strikes or double heel-strikes, which occurred during human subject evaluations. Real-time estimation of ankle mechanics would allow the system to be utilized outside the lab. Towards this, an artificial neural network could be trained to utilize insoles, equipped with inertial measurement units and FSRs, to adequately estimate the ankle torques.

7.1.3 Human-in-the-loop optimization

Advancements in the learning algorithm, presented in Ch. 4, are possible. A higher fidelity dynamic model of the prosthesis would allow for increased learning, especially with respect

to phase errors. A model would allow for phase errors greater than 90^o to be accommodated, since the model would encode the frequency response of the actuator. Moreover, building a model of the human-robot response dynamics, using the achieved torques at each iteration, would allow the algorithm to account and adapt to the human contributions [87]. Additionally, learning a time-invariant control law, e.g., phase angle based [99, 100] or impedance based [35], would increase robustness with respect to time variations. Lastly, a method for interpolating learned trajectories would reduce the need for learning trials and allow for a more comprehensive trajectory library, e.g., interpolation for different variations in the environment (e.g., speed, ramp, load carriage).

7.2 Conclusions

The work accomplished in this thesis has demonstrated the potential for personalized power ankle-foot prostheses. The contributions are as follows:

- **a cam-based approach to personalizing the passive response**
  The elastic response of a biological ankle, at self-selected walking gait, is well approximated using nonlinear elements. As a result, a nonlinear element can substantially decrease the requirements on the active component when compared with a linear element. In particular, an optimal nonlinear element can substantially reduce the positive energy ($\sim 40\%$), peak positive power ($\sim 29\%$), and peak torque ($\sim 21\%$) when compared to an optimal linear element. A cam-based passive element was designed to realize the nonlinear element, and subsequently used for the prototype powered ankle-foot prostheses.

- **a learning algorithm that can avoid divergence**
  This work proposed a human-in-the-loop control strategy for powered ankle-foot prostheses that targets gait asymmetry. The main challenge with human-in-the-loop approaches is to avoid control signal divergence in the presence of time-varying human response dynamics. Towards this, an algorithm was develop that can iteratively adapt

the active ankle torque of the prosthetic limb towards matching the achieved ankle toque of the intact limb. The algorithm utilized an adaptive backstepping iterative learning control algorithm, which effectively maintains boundedness of the control signal, even in the presences of large variations of the reference signal. However, boundedness comes at a cost of exploration: since any growth in the error causes the algorithm to backstep, this new information is effectively lost. The algorithm could be potential adapted to address this limitation, by, for example, incorporating the data into a model of the human-robot dynamics.

- **clinical evaluation and hypothesis testing**

  An experimental pilot study was conducted to test the hypotheses that the learning symmetry controller would reduce asymmetries and loading factors known to be associated with OA. The results demonstrated that the proposed approach was able to significantly reduce ankle moment asymmetries when compared with the passive (no control) condition, which was the primary goal. Additionally, a significant reduction in peak hip adduction moment was observed, e.g., a load factor associated with osteoarthritis. Knee adduction moment was also decreased, however, when comparing the passive and symmetry conditions significants was not reach. Qualitatively, the first peak of the knee adduction moment was largely reduced. Secondary analysis revealed large reduction in sagittal plane knee and hip kinetic and kinematic asymmetries. The results indicate that symmerty control can reduce asymmetries, especially with respect to ankle moment. Addiontally, explicitly targeting symmetries at the ankle, resulted in a reduction of other asymmetries at the hip and knee, thus providing a starting point for future personalized approaches that seek to target the altered body mechanics of amputees.

BIBLIOGRAPHY

[1] David C Morgenroth, Alfred C Gellhorn, and Pradeep Suri. Osteoarthritis in the disabled population: a mechanical perspective. *PM&R*, 4(5):S20–S27, 2012.

[2] Robert Gailey, Kerry Allen, Julie Castles, Jennifer Kucharik, and Mariah Roeder. Review of secondary physical conditions associated with lower-limb amputation and long-term prosthesis use. *Journal of Rehabilitation Research and Development*, 45(1):15, 2008.

[3] David A Winter. Energy generation and absorption at the ankle and knee during fast, natural, and slow cadences. *Clinical Orthopaedics and Related Research*, 175:147–154, 1983.

[4] David A Winter and Susan E Sienko. Biomechanics of below-knee amputee gait. *Journal of Biomechanics*, 21(5):361–367, 1988.

[5] Harry B Skinner and David J Effeney. Gait analysis in amputees. *American Journal of Physical Medicine & Rehabilitation*, 64(2):82–89, 1985.

[6] Hamid Bateni and Sandra J Olney. Kinematic and kinetic variations of below-knee amputee gait. *JPO: Journal of Prosthetics and Orthotics*, 14(1):2–10, 2002.

[7] P.G. Adamczyk and A.D. Kuo. Mechanisms of gait asymmetry due to push-off deficiency in unilateral amputees. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 23(5):776–785, Sept 2015.

[8] David J Sanderson and Philip E Martin. Lower extremity kinematic and kinetic adaptations in unilateral below-knee amputees during walking. *Gait & posture*, 6(2):126–136, 1997.

[9] E Isakov, O Keren, and N Benjuya. Trans–tibial amputee gait: Time–distance parameters and emg activity. *Prosthetics and Orthotics International*, 24(3):216–220, 2000.

[10] Sarah J Mattes, Philip E Martin, and Todd D Royer. Walking symmetry and energy cost in persons with unilateral transtibial amputations: matching prosthetic and intact limb inertial properties. *Archives of Physical Medicine and Rehabilitation*, 81(5):561–568, 2000.

[11] Ronald D Snyder, Christopher M Powers, Catherine Fontaine, and Jacquelin Perry. The effect of five prosthetic feet on the gait and loading of the sound limb in dysvascular below-knee amputees. *Journal of Rehabilitation Research and Development*, 32(4):309, 1995.

[12] Michael S Pinzur, William Cox, James Kaiser, Ted Morris, et al. The effect of prosthetic alignment on relative limb loading in persons with trans-tibial amputation: a preliminary report. *Journal of Rehabilitation Research and Development*, 32(4):373, 1995.

[13] AP Arya, A Lees, HC Nerula, and L Klenerman. A biomechanical comparison of the sach, seattle and jaipur feet using ground reaction forces. *Prosthetics and Orthotics International*, 19(1):37–45, 1995.

[14] Robert J Zmitrewicz, Richard R Neptune, Judith G Walden, William E Rogers, and Gordon W Bosker. The effect of foot and ankle prosthetic components on braking and propulsive impulses during transtibial amputee gait. *Archives of Physical Medicine and Rehabilitation*, 87(10):1334–1339, 2006.

[15] JR Engsberg, AG Lee, KG Tedford, and JA Harder. Normative ground reaction force data for able-bodied and below-knee-amputee children during walking. *Journal of Pediatric Orthopedics*, 13(2):169–173, 1992.

[16] C Beyaert, C Grumillier, N Martinet, J Paysant, and J-M André. Compensatory mechanism involving the knee joint of the intact limb during gait in unilateral below-knee amputees. *Gait & posture*, 28(2):278–284, 2008.

[17] NH Molen. Energy/speed relation of below-knee amputees walking on a motor-driven treadmill. *European Journal of Applied Physiology and Occupational Physiology*, 31(3):173–185, 1973.

[18] EG Gonzalez, PJ Corcoran, and RL Reyes. Energy expenditure in below-knee amputees: correlation with stump length. *Archives of physical medicine and rehabilitation*, 55(3):111, 1974.

[19] G Robert Colborne, Stephen Naumann, Patricia E Longmuir, and David Berbrayer. Analysis of mechanical and metabolic factors in the gait of congenital below knee amputees: A comparison of the sach and seattle feet. *American Journal of Physical Medicine & Rehabilitation*, 71(5):272–278, 1992.

[20] Elizabeth Russell Esposito and Jason M Wilken. Biomechanical risk factors for knee osteoarthritis when using passive and powered ankle–foot prostheses. *Clinical Biomechanics*, 29(10):1186–1192, 2014.

[21] Leslie Torburn, Christopher M Powers, Robert Guiterrez, and Jacquelin Perry. Energy expenditure during ambulation in dysvascular and traumatic below-knee amputees: a comparison of five prosthetic feet. *Journal of rehabilitation research and development*, 32(2):111, 1995.

[22] Miao-Ju Hsu, David H Nielsen, Suh-Jen Lin-Chan, and Donald Shurr. The effects of prosthetic foot design on physiologic measurements, self-selected walking velocity, and physical activity in people with transtibial amputation. *Archives of physical medicine and rehabilitation*, 87(1):123–129, 2006.

[23] David C Morgenroth, Ava D Segal, Karl E Zelik, Joseph M Czerniecki, Glenn K Klute, Peter G Adamczyk, Michael S Orendurff, Michael E Hahn, Steven H Collins, and Art D Kuo. The effect of prosthetic foot push-off on mechanical loading associated with knee osteoarthritis in lower extremity amputees. *Gait & posture*, 34(4):502–507, 2011.

[24] Hugh M Herr and Alena M Grabowski. Bionic ankle–foot prosthesis normalizes walking gait for persons with leg amputation. In *Proc. R. Soc. B*, volume 279, pages 457–464. The Royal Society, 2012.

[25] Rino Versluys, Pieter Beyl, Michael Van Damme, Anja Desomer, Ronald Van Ham, and Dirk Lefeber. Prosthetic feet: State-of-the-art review and the importance of mimicking human ankle–foot biomechanics. *Disability and Rehabilitation: Assistive Technology*, 4(2):65–75, 2009.

[26] Joseph K Hitt, Thomas G Sugar, Matthew Holgate, and Ryan Bellman. An active foot-ankle prosthesis with biomechanical energy regeneration. *Journal of medical devices*, 4(1):011003, 2010.

[27] Bryan J Bergelin and Philip A Voglewede. Design of an active ankle-foot prosthesis utilizing a four-bar mechanism. *Journal of Mechanical Design*, 134(6):061004, 2012.

[28] GK Klute, J Czerniecki, and B Hannaford. Muscle-like pneumatic actuators for below-knee prostheses. In *Proceedings the 7th International Conference on New Actuators*, pages 289–292, 2000.

[29] Jinying Zhu, Qining Wang, and Long Wang. Pantoe 1: Biomechanical design of powered ankle-foot prosthesis with compliant joints and segmented foot. In *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*, pages 31–36. IEEE, 2010.

[30] Samuel Au and H Herr. Powered Ankle-foot Prosthesis. *IEEE Robotics & Automation Magazine*, 15(3):52–59, 2008.

[31] Pierre Cherelle, Victor Grosu, Louis Flynn, Karen Junius, Marta Moltedo, Bram Vanderborght, and Dirk Lefeber. The ankle mimicking prosthetic foot 3locking mechanisms, actuator design, control and experiments with an amputee. *Robotics and Autonomous Systems*, 91:327–336, 2017.

[32] Elliott J Rouse, Luke M Mooney, Ernesto C Martinez-Villalpando, and Hugh M Herr. Clutchable Series-Elastic Actuator: Design of a Robotic Knee Prosthesis for Minimum Energy Consumption. In *2013 IEEE International Conference on Rehabilitation Robotics (ICORR 2013)*, pages 1–6. IEEE, 2013.

[33] Jinming Sun, Jessica M Fritz, David R Del Toro, and Philip A Voglewede. Amputee subject testing protocol, results, and analysis of a powered transtibial prosthetic device. *Journal of medical devices*, 8(4):041007, 2014.

[34] W. Robert Brown and A. Galip Ulsoy. A Maneuver Based Design of a Passive-Assist Device for Augmenting Active Joints. *Journal of Mechanisms and Robotics*, 5(3):031003, June 2013.

[35] Amanda H Shultz, Brian E Lawson, and Michael Goldfarb. Variable cadence walking and ground adaptive standing with a powered ankle prosthesis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(4):495–505, 2016.

[36] Joshua M Caputo and Steven H Collins. Prosthetic ankle push-off work reduces metabolic rate but not collision work in non-amputee walking. *Scientific reports*, 4, 2014.

[37] Samuel K Au, Jeff Weber, and Hugh Herr. Powered ankle–foot prosthesis improves walking metabolic economy. *IEEE Transactions on Robotics*, 25(1):51–66, 2009.

[38] Emily S Gardinier, Brian M Kelly, Jeffrey Wensman, and Deanna H Gates. A controlled clinical trial of a clinically-tuned powered ankle prosthesis in people with transtibial amputation. *Clinical Rehabilitation*, page 0269215517723054, 2017.

[39] Michael F Eilenberg, Hartmut Geyer, and Hugh Herr. Control of a powered ankle–foot prosthesis based on a neuromuscular model. *IEEE transactions on neural systems and rehabilitation engineering*, 18(2):164–173, 2010.

[40] Deepak Gopinath, Siddarth Jain, and Brenna D Argall. Human-in-the-loop optimization of shared autonomy in assistive robotics. *IEEE Robotics and Automation Letters*, 2(1):247–254, 2017.

[41] David A Winter. Kinematic and kinetic patterns in human gait: variability and compensating effects. *Human Movement Science*, 3(1):51–76, 1984.

[42] Joseph M Czerniecki and David C Morgenroth. Metabolic energy expenditure of ambulation in lower extremity amputees: what have we learned and what are the next steps? *Disability and rehabilitation*, 39(2):143–151, 2017.

[43] Konstantinos Dermitzakis, Juan Pablo Carbajal, and James H Marden. Scaling laws in robotics. *Procedia Computer Science*, 7:250–252, 2011.

[44] Verne T Inman. Human locomotion. *Canadian Medical Association Journal*, 94(20):1047, 1966.

[45] David A Winter. Overall principle of lower limb support during stance phase of gait. *Journal of biomechanics*, 13(11):923–927, 1980.

[46] David A Winter. Biomechanical motor patterns in normal walking. *Journal of motor behavior*, 15(4):302–330, 1983.

[47] Janice J Eng and David A Winter. Kinetic analysis of the lower limbs during walking: what information can be gained from a three-dimensional model? *Journal of biomechanics*, 28(6):753–758, 1995.

[48] Roy B Davis and Peter A DeLuca. Gait characterization via dynamic joint stiffness. *Gait & Posture*, 4(3):224–231, 1996.

[49] Andrew H Hansen, Dudley S Childress, Steve C Miff, Steven A Gard, and Kent P Mesplay. The human ankle during walking: implications for design of biomimetic ankle prostheses. *Journal of biomechanics*, 37(10):1467–1474, 2004.

[50] Michael Lars Palmer. Sagittal Plane Characterization of Normal Human Ankle Function Across a Range of Walking Gait Speeds. Master's thesis, Dept. Mech. Eng., Massachusetts Institute of Technology, Boston, 2002.

[51] Justus F Lehmann, Robert Price, Sherlyn Boswell-Bessette, Al Dralle, and Kent Questad. Comprehensive analysis of dynamic elastic response feet: Seattle ankle/lite foot versus sach foot. *Archives of physical medicine and rehabilitation*, 74(8):853–861, 1993.

[52] Michael R Menard and D Duncan Murray. Subjective and objective analysis of an energy-storing prosthetic foot. *JPO: Journal of Prosthetics and Orthotics*, 1(4):220–230, 1989.

[53] David H Nielsen, Donald G Shurr, Jane C Golden, and Kenneth Meier. Comparison of energy cost and gait efficiency during ambulation in below-knee amputees using different prosthetic feet-a preliminary report. *JPO: Journal of Prosthetics and Orthotics*, 1(1):24–31, 1988.

[54] Laetitia Fradet, Merkur Alimusaj, Frank Braatz, and Sebastian I Wolf. Biomechanical analysis of ramp ambulation of transtibial amputees with an adaptive ankle foot system. *Gait & posture*, 32(2):191–198, 2010.

[55] Steven H Collins and Arthur D Kuo. Recycling energy to restore impaired ankle function during human walking. *PLoS one*, 5(2):e9307, 2010.

[56] Ava D Segal, Karl E Zelik, Glenn K Klute, David C Morgenroth, Michael E Hahn, Michael S Orendurff, Peter G Adamczyk, Steven H Collins, Arthur D Kuo, and Joseph M Czerniecki. The effects of a controlled energy storage and return prototype prosthetic foot on transtibial amputee ambulation. *Human movement science*, 31(4):918–931, 2012.

[57] M Mussman, W Altwerger, J Eisenstein, A Turturro, A Glockenberg, and L Bubbers. Contralateral lower extremity evaluation with a lower limb prosthesis. *Journal of the American Podiatry Association*, 73(7):344–346, 1983.

[58] Daniel C Norvell, Joseph M Czerniecki, Gayle E Reiber, Charles Maynard, Janice A Pecoraro, and Noel S Weiss. The prevalence of knee pain and symptomatic knee osteoarthritis among veteran traumatic amputees and nonamputees. *Archives of physical medicine and rehabilitation*, 86(3):487–493, 2005.

[59] Pieter A Struyf, Caroline M van Heugten, Minou W Hitters, and Rob J Smeets. The prevalence of osteoarthritis of the intact hip and knee among traumatic leg amputees. *Archives of physical medicine and rehabilitation*, 90(3):440–446, 2009.

[60] Jai Kulkarni, Judith Adams, Elaine Thomas, and Alan Silman. Association between amputation, arthritis and osteopenia in british male war veterans with major lower limb amputations. *Clinical Rehabilitation*, 12(4):348–353, 1998.

[61] MJ Burke, V Roman, and V Wright. Bone and joint changes in lower limb amputees. *Annals of the rheumatic diseases*, 37(3):252–254, 1978.

[62] AJ Baliunas, DE Hurwitz, AB Ryals, A Karrar, JP Case, JA Block, and TP Andriacchi. Increased knee joint loads during walking are present in subjects with knee osteoarthritis. *Osteoarthritis and cartilage*, 10(7):573–579, 2002.

[63] Debra E Hurwitz, Kharma C Foucher, Dale R Sumner, Thomas P Andriacchi, Aaron G Rosenberg, and Jorge O Galante. Hip motion and moments during gait relate directly to proximal femoral bone mineral density in patients with hip osteoarthritis. *Journal of biomechanics*, 31(10):919–925, 1998.

[64] Chandra H Lloyd, Steven J Stanhope, Irene S Davis, and Todd D Royer. Strength asymmetry and osteoarthritis risk factors in unilateral trans-tibial, amputee gait. *Gait & posture*, 32(3):296–300, 2010.

[65] Todd D Royer and Carolyn A Wasilewski. Hip and knee frontal plane moments in persons with unilateral, trans-tibial amputation. *Gait & posture*, 23(3):303–306, 2006.

[66] Jinming Sun and Philip A. Voglewede. Powered Transtibial Prosthetic Device Control System Design, Implementation, and Bench Testing. *Journal of Medical Devices*, 8(1):011004, December 2013.

[67] Rino Versluys, Arnout Matthys, Ronald Van Ham, Innes Vanderniepen, and Dirk Lefeber. Powered Ankle-Foot System That Mimics Intact Human Ankle Behavior: Proposal of a New Concept. *2009 IEEE International Conference on Rehabilitation Robotics (ICORR 2009)*, pages 658–662, June 2009.

[68] Ava D Segal, Karl E Zelik, Glenn K Klute, David C Morgenroth, Michael E Hahn, Michael S Orendurff, Peter G Adamczyk, Steven H Collins, Arthur D Kuo, and Joseph M Czerniecki. The Effects of a Controlled Energy Storage and Return Prototype Prosthetic Foot on Transtibial Amputee Ambulation. *Human Movement Science*, 31(4):918–931, 2012.

[69] Frank Sup, Amit Bohara, and Michael Goldfarb. Design and Control of a Powered Transfemoral Prosthesis. *The International Journal of Robotics Research*, 27(2):263–273, February 2008.

[70] B. Vanderborght, A. Albu-Schaeffer, A. Bicchi, E. Burdet, D.G. Caldwell, R. Carloni, M. Catalano, O. Eiberger, W. Friedl, G. Ganesh, M. Garabini, M. Grebenstein, G. Grioli, S. Haddadin, H. Hoppner, A. Jafari, M. Laffranchi, D. Lefeber, F. Petit, S. Stramigioli, N. Tsagarakis, M. Van Damme, R. Van Ham, L.C. Visser, and S. Wolf. Variable Impedance Actuators: A Review. *Robotics and Autonomous Systems*, 61(12):1601 – 1614, 2013.

[71] Christine Vehar Jutte and Sridhar Kota. Design of Nonlinear Springs for Prescribed Load-Displacement Functions. *Journal of Mechanical Design*, 130(8):081403, 2008.

[72] DFB Haeufle, MD Taylor, S Schmitt, and H Geyer. A Clutched Parallel Elastic Actuator Concept: Towards Energy Efficient Powered Legs in Prosthetics and Robotics. In *4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob 2012)*, pages 1614–1619. IEEE, 2012.

[73] Ryan J Williams, Andrew H Hansen, and Steven A Gard. Prosthetic Ankle-Foot Mechanism Capable of Automatic Adaptation to the Walking Surface. *Journal of Biomechanical Engineering*, 131(3):035002, March 2009.

[74] Eric A. Nickel, Andrew H. Hansen, and Steven A. Gard. Prosthetic Ankle-Foot System That Adapts to Sloped Surfaces. *Journal of Medical Devices*, 6(1):011006, 2012.

[75] Ivar Thorson and Darwin Caldwell. A Nonlinear Series Elastic Actuator for Highly Dynamic Motions. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 390–394, September 2011.

[76] Sebastian Wolf and Gerd Hirzinger. A New Variable Stiffness Design: Matching Requirements of the Next Robot Generation. *2008 IEEE International Conference on Robotics and Automation*, pages 1741–1746, May 2008.

[77] Bram Vanderborght, Nikos G. Tsagarakis, Ronald Ham, Ivar Thorson, and Darwin G. Caldwell. MACCEPA 2.0: Compliant Actuator Used for Energy Efficient Hopping Robot Chobino1D. *Autonomous Robots*, 31(1):55–65, April 2011.

[78] Shane A. Migliore, Edgar A. Brown, and Stephen P. DeWeerth. Novel Nonlinear Elastic Actuators for Passively Controlling Robotic Joint Compliance. *Journal of Mechanical Design*, 129(4):406, 2007.

[79] René Jimenez-Fabian and Olivier Verlinden. Review of control algorithms for robotic ankle systems in lower-limb orthoses, prostheses, and exoskeletons. *Medical Engineering & Physics*, 34(4):397–408, 2012.

[80] Stephanie Huang, Jeffrey P Wensman, and Daniel P Ferris. Locomotor adaptation by transtibial amputees walking with an experimental powered prosthesis under continuous myoelectric control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(5):573–581, 2016.

[81] Jeffrey R Koller, Deanna H Gates, Daniel P Ferris, and C David Remy. 'body-in-the-loop' optimization of assistive robotic devices: A validation study. In *Robotics: Science and Systems*, 2016.

[82] Juanjuan Zhang, Pieter Fiers, Kirby A Witte, Rachel W Jackson, Katherine L Poggensee, Christopher G Atkeson, and Steven H Collins. Human-in-the-loop optimization of exoskeleton assistance during walking. *Science*, 356(6344):1280–1284, 2017.

[83] Jonathan Realmuto, Rahul B Warrier, and Santosh Devasia. Iterative learning control for human-robot collaborative output tracking. In *Mechatronic and Embedded Systems and Applications (MESA), 2016 12th IEEE/ASME International Conference on*, pages 1–6. IEEE, 2016.

[84] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Bettering operation of robots by learning. *Journal of Field Robotics*, 1(2):123–140, 1984.

[85] Ch Atkeson and Joe McIntyre. Robot trajectory learning through practice. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1737–1742. IEEE, 1986.

[86] Mikael Norrlof. An adaptive iterative learning control algorithm with experiments on an industrial robot. *IEEE Transactions on robotics and automation*, 18(2):245–251, 2002.

[87] Jonathan Realmuto, Rahul B. Warrier, and Santosh Devasia. Data-inferred personalized human-robot models for iterative collaborative output tracking. *Journal of Intelligent and Robotic Systems*, Accepted June 6, 2017.

[88] M.W. Legro, G. Reiber, M. del Aguila, M.J. Ajax, D.A. Boone, J.A. Larsen, D.G. Smith, and B. Sangeorzan. Issues of Importance Reported by Persons With Lower Limb Amputations and Prostheses. *Journal of Rehabilitation Research and Development*, 36(3):155–163, 1999.

[89] Michael W Whittle. *Gait Analysis: An Introduction*. 2003.

[90] U. Mettin, P. X. La Hera, L. B. Freidovich, and A. S. Shiriaev. Parallel Elastic Actuators as a Control Tool for Preplanned Trajectories of Underactuated Mechanical Systems. *The International Journal of Robotics Research*, 29(9):1186–1198, August 2009.

[91] Martin Grimmer, Mahdy Eslamy, Stefan Gliech, and André Seyfarth. A Comparison of Parallel-and Series Elastic Elements in an Actuator for Mimicking Human Ankle Joint in Walking and Running. In *IEEE 2012 International Conference on Robotics and Automation (ICRA 2012)*, pages 2463–2470. IEEE, 2012.

[92] Glenn Mathijssen, Pierre Cherelle, Dirk Lefeber, and Bram Vanderborght. Concept of a Series-Parallel Elastic Actuator for a Powered Transtibial Prosthesis. *Actuators*, 2:59–73, 2013.

[93] Kevin W Hollander, Robert Ilg, Thomas G Sugar, and Donald Herring. An Efficient Robotic Tendon for Gait Assistance. *Journal of Biomechanical Engineering*, 128(5):788–791, 2006.

[94] G.A. Pratt and M.M. Williamson. Series elastic actuators. In *1995 IEEE/RSJ International Conference on Human Robot Interaction and Cooperative Robots (Intelligent Robots and Systems 95)*, volume 1, pages 399–406 vol.1, Aug 1995.

[95] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[96] Robert L Norton. *Cam Design and Manufacturing Handbook*. Industrial Press, 2009.

[97] Szuchi Tien, Qingze Zou, and Santosh Devasia. Iterative control of dynamics-coupling-caused errors in piezoscanners during high-speed afm operation. *IEEE Transactions on Control Systems Technology*, 13(6):921–931, 2005.

[98] Annegret Mündermann, Chris O Dyrby, and Thomas P Andriacchi. Secondary gait changes in patients with medial compartment knee osteoarthritis: increased load at the ankle, knee, and hip during walking. *Arthritis & Rheumatology*, 52(9):2835–2844, 2005.

[99] David Quintero, Anne E Martin, and Robert D Gregg. Toward unified control of a powered prosthetic leg: A simulation study. *IEEE Transactions on Control Systems Technology*, 2017.

[100] Juan De la Fuente, Thomas G Sugar, and Sangram Redkar. Nonlinear, phase-based oscillator to generate and assist periodic motions. *Journal of Mechanisms and Robotics*, 9(2):024502, 2017.

[101] Max K Shepherd and Elliott J Rouse. Design of a quasi-passive ankle-foot prosthesis with biomimetic, variable stiffness. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 6672–6678. IEEE, 2017.

Appendix A

SUPPLEMENTARY HARDWARE INFORMATION

A.1 Cam spring stiffness verification

To verify the elastic response achieved by the cam-follower mechanism, the powered ankle prosthesis (with the drive train and composite foot removed) was subjected to static loading experiments using an Instron 5585H electromechanical materials testing machine. The experimental setup to angularly deflect the ankle joint and measure the resulting torque is shown in Fig. A.1.

A.1.1 Obtaining the elastic response

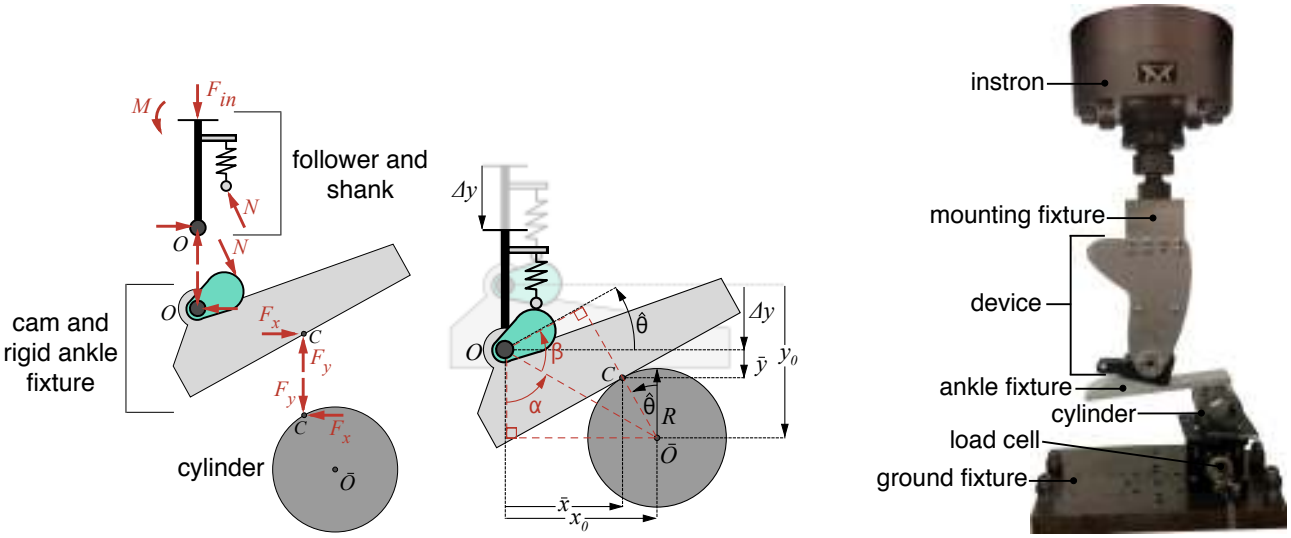
The experimental cam torque $\hat{T}_c$ can be calculated in terms of the reaction forces (F_x and F_z measured with a 6-DOF load cell (AMTI MC3A), see Fig. A.1a) by taking the moment about the ankle-joint center O and using Eq. (3.18) as

$$\hat{T}_c = -\bar{y}\ F_x\ -\bar{x}\ F_y \tag{A.1}$$

where $\bar{y}$ and $\bar{x}$ are the vertical and horizontal distances from ankle-joint center O to the contact point C between the ankle fixture and the cylinder, as shown in Fig A.1b. From the geometrical constraints, $\bar{y}$ and $\bar{x}$ are calculated as

$$\begin{aligned} \bar{x} &= x_0 - R\sin\hat{\theta}, \\ \bar{y} &= y_0 - \Delta y - R(1-\cos\hat{\theta}), \end{aligned} \tag{A.2}$$

where $\hat{\theta}$ is the experimentally measured rotational displacement of the ankle joint, and y_0 and x_0 are the initial vertical and horizontal distances from the ankle joint center O to the cylinder center $\bar{O}$, respectively (see Fig. A.1b), Δy is the measured vertical displacement



(a) (b) (c)

Figure A.1: Images detailing the static loading experiment. (a) Free body diagram of static loading experiment. (b) Schematic of static loading experiment illustrating the geometry and the angular displacement $\hat{\theta}$, caused by input displacement Δy. (c) Image of experimental setup.

imposed by the Instron machine, and R is the diameter of the cylinder in Fig. A.1b. The distances y_0 and x_0 are known from the geometry of the system. The rotational displacement $\hat{\theta}$ of the ankle joint, due to the vertical displacement Δy, can be calculated as

$$\hat{\theta} = \frac{\pi}{2} - [\pi - (\alpha + \beta)] \quad = \alpha + \beta - \frac{\pi}{2}, \tag{A.3}$$

where

$$\sin\alpha = \frac{x_0}{h}, \quad \sin\beta = \frac{y_0}{h}, \quad h = \sqrt{(y_0 - \Delta y)^2 + x_0^2}. \tag{A.4}$$

A.1.2 Nominal results

The experimentally obtained elastic response, e.g., the cam torque $\hat{T}_c$ from Eq. (A.1) versus angular displacement $\hat{\theta}$ from Eq. (A.3), for three loading and unloading cycles is compared

with the desired elastic response (from Eq. (3.17)) in Fig. A.2. Significant deviation ($\%E_{max} = 33.8\%$) from the optimal profile is seen where the max error $\%E_{max}$ is calculated as

$$\%E_{max} = \frac{\max\limits_{i\in\{1,\cdots,p\}} \left|T_p^*[i] - \hat{T}_c[i]\right|}{\max\limits_{i\in\{1,\cdots,p\}} \left|T_p^*[i]\right|} \cdot 100\%, \tag{A.5}$$

where $[i]$ represents the i^{th} data sample, p is the total number of samples, $\hat{T}_c[i]$ are the experimentally determined torque values given by Eq. (A.1), and $T_p^*[i]$ correspond to the desired (optimal) torque at the measured angular displacement $\hat{\theta}[i]$.

A.1.3 Corrected results

To correct the cam response, shims (spacers) were placed between the top of the spring and the support structure (see Fig. 3.14a) to adjust deviations in the follower-spring preload. The torque $\tilde{T}_c$ with a shim thickness of $\bar{s}$ can be found from Eq. (3.21) as

$$\tilde{T}_{c,\bar{s}} = -\tilde{k}_c(s + \bar{s})s' \tag{A.6}$$

where $\tilde{k}_c = 722$ kN/m is the measured spring constant found using a static load test, which is different from the manufacturer-reported nominal value of $k_c = 817$ kN/m used in the design. First, the potential error in the preload is estimated in terms of a shim thickness $\bar{s}_1$ by fitting the elastic response with a shim in Eq. (A.6) to the experimental elastic response through the following minimization

$$\min_{\bar{s}} \sum_{i=1}^{p} \left(\hat{T}_c[i] - \tilde{T}_{c,\bar{s}}[i]\right)^2. \tag{A.7}$$

where $[i]$ represents the i^{th} data sample, and p is the total number of samples, $\hat{T}_c$ are the experimentally determined torque values given by Eq. (A.1), and $\tilde{T}_{c,\bar{s}}[i]$ corresponds to the torque at the measured angular displacement $\hat{\theta}[i]$. The optimal shim $\bar{s}_1$ that solves the minimization problem in Eq. (A.7), corresponds to the inherent error in the preload of the device. Next, the required preload is estimated in terms of a shim thickness $\bar{s}$ by fitting the elastic

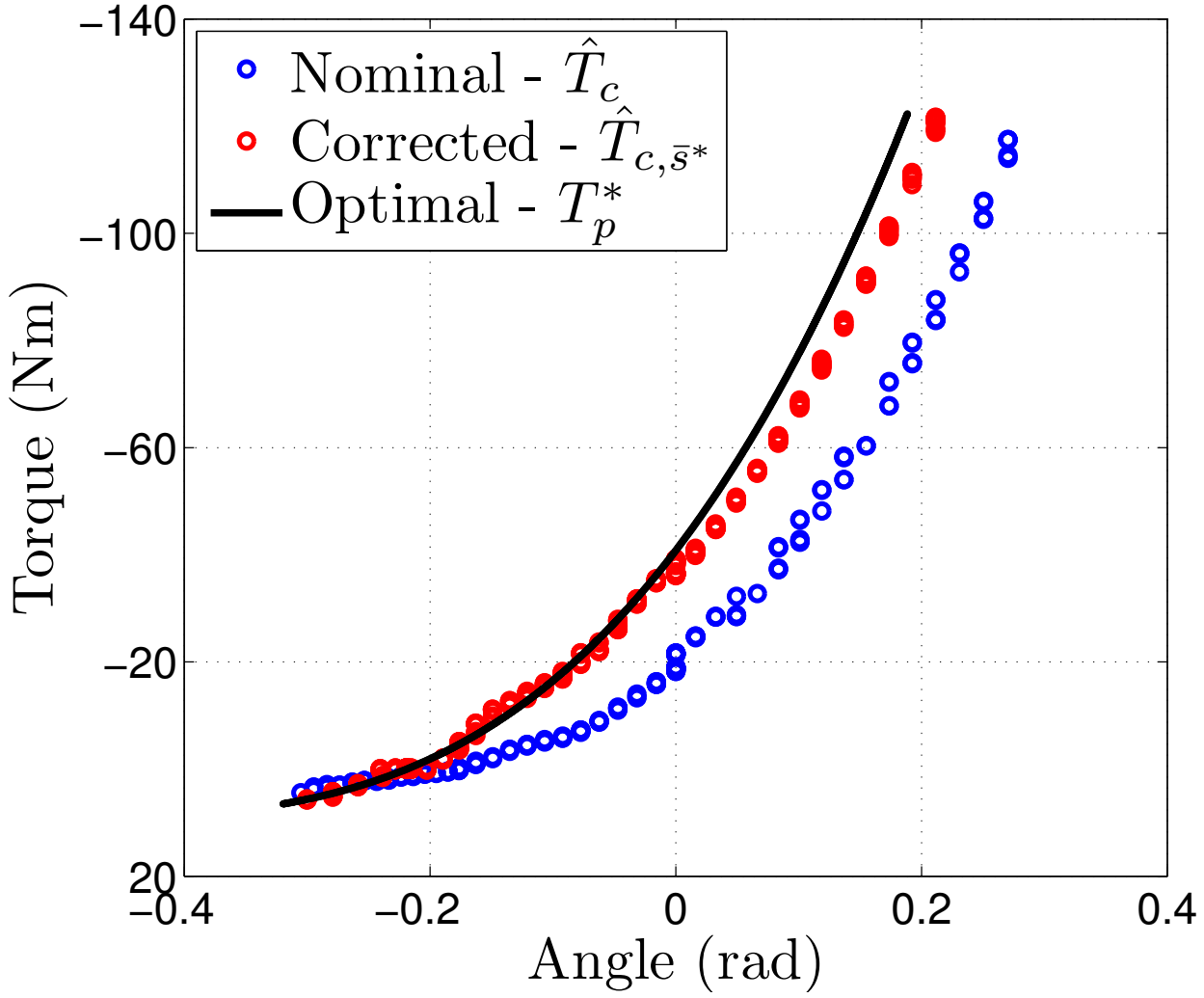


Figure A.2: Experimental validation of cam elastic response. The blue circles represent the experimental data points for the nominal case. Red circles are experimental data points with a 3 mm shim added to the device.

response with a shim in Eq. (A.6) to the optimal elastic response through the following minimization

$$\min_{\bar{s}} \sum_{i=1}^{p} \left(T_p^*[i] - \tilde{T}_{c,\bar{s}}[i] \right)^2. \tag{A.8}$$

where $T_p^*[i]$ and $\tilde{T}_{c,\bar{s}}[i]$ correspond to the optimal torque and shim-based torque, respectively, calculated at different measured angular displacement values $\hat{\theta}[i]$. The optimal solution, $\bar{s}_2$ that solves the minimization problem in Eq. (A.8), corresponds to the preload needed to correct the elastic response due to error in the follower-spring's spring constant. Thus, the shim thickness $\bar{s}^*$ needed to correct for errors in the cam's elastic response can be calculated as

$$\bar{s}^* = \bar{s}_2 - \bar{s}_1 \approx 3 \text{ mm} \tag{A.9}$$

After adding the shims, the experiment to determine the elastic response was repeated and the results are shown in Fig. A.2. The maximum percent error (Eq. (A.5)) of the ankle with the shims is 9.9%

A.2 PEA optimization source code

The following Matlab code optimizes a PEA based on an input trajectory. The output is the stiffness coefficients described in Sect. 3.3.1 in (3.6).

```
%-------------------------------------------------------------------------
% Prosethetic Ankle-Foot
%
% -- optimize PEA--
%
% Convex optimization for parallel n-th order polynomial spring
%
% v4 - 10 April 2013
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Inputs: tp   -   Parallel Trajectory
%              t   :   xd
%              t   :   x1d
%              t   :   Td
%              t   :   T1d
%         n    -   Poly order
%         a1   -   Positive Energy Cost Coeff
%         a2   -   Max Positive Power Cost Coeff
%         a3   -   Max Torque Cost Coeff
%
% Output: PEA   :   n         -   poly order
%               :   kp        -   spring constant vector
%               :   Fa        -   Active Force Requirment
%               :   F1a       -   d/dt (Active Force Requirment)
%               :   Pa        -   Acitve Power Requirment
%               :   Fp        -   Optimal Passive/Parallel Torque
%               :   Pp        -   Optimal Passive/Parallel Power
```

```
%               :   Ea_pos    -   Positive Active Energy Requirment
%               :   Ea_neg    -   Negative Active Energy Requirment
%               :   E_cost    -   Active Energy Cost
%               :   P_cost    -   Active Power Cost
%               :   F_cost    -   Active Torque Cost
%               :   fobj_val  -   fobjective val
%               :   real_cost -   real cost
%               :   Fp_prf    -   Passive/Parallel Torque Profile
%--------------------------------------------------------------------------

function PEA = opt_PEA(t, n, a1, a2, a3)
%--------------------------------------------------------------------------
% Ankle Range of Motion (for constraints)
%--------------------------------------------------------------------------
dx = (pi/1000);
max_x = max(t.xd) + dx - mod((max(t.xd) + dx), dx);
min_x = min(t.xd) - dx - mod((min(t.xd) - dx), dx);
xRange = min_x:dx:max_x;

%--------------------------------------------------------------------------
% Anon Function Defs.
%--------------------------------------------------------------------------
X = [];
X_1 = [];
X_2 = [];
X_Range = [];
for i=0:n
  X(:,end+1) = t.xd.^i;
  X_1(:,end+1) = i*t.xd.^(i-1).*t.x1d;
  X_2(:,end+1) = i*((i-1)*t.xd.^(i-2).*t.x1d.^2 + t.xd.^(i-1).*t.x2d);
  X_Range(:,end+1) = xRange.^i;
end

Tp_fun =@(kp) -X*kp;
T1p_fun =@(kp) -X_1*kp;
T2p_fun =@(kp) -X_2*kp;
Pp_fun =@(kp) Tp_fun(kp).*t.x1d;

Tm_fun =@(kp) (t.Td - Tp_fun(kp));
T1m_fun =@(kp) (t.T1d - T1p_fun(kp));
T2m_fun =@(kp) (t.T2d - T2p_fun(kp));

Pm_fun =@(kp) (Tm_fun(kp).*t.x1d);

Pm_pos_fun =@(kp) (Pm_fun(kp) + abs(Pm_fun(kp)))/2;
Pm_neg_fun =@(kp) (Pm_fun(kp) - abs(Pm_fun(kp)))/2;

Em_pos_fun =@(kp) sum( Pm_pos_fun(kp) );
Em_neg_fun =@(kp) sum( Pm_neg_fun(kp) );

Em_actual_pos_fun =@(kp) trapz(Pm_pos_fun(kp))*t.dt;
Em_actual_neg_fun =@(kp) trapz(Pm_neg_fun(kp))*t.dt;

Tp_range_fun =@(kp) -X_Range*kp;

E_cost_fun =@(kp) Em_pos_fun(kp)*t.dt;
P_cost_fun =@(kp) max(Pm_pos_fun(kp));
T_cost_fun =@(kp) norm(Tm_fun(kp), inf);

fobjective =@(kp) a1*E_cost_fun(kp) ...
                + a2*P_cost_fun(kp) ...
                + a3*T_cost_fun(kp);

real_cost_fun =@(kp) Em_actual_pos_fun(kp) ...
                     + P_cost_fun(kp) ...
                     + T_cost_fun(kp);
%--------------------------------------------------------------------------
% Optimization
%--------------------------------------------------------------------------
X_CON = [zeros(numel(xRange),1)];
for i=1:n
  X_CON(:,end+1) = xRange.^i;
end

Aineq = -diag([sign(xRange)])*X_CON;
bineq = zeros(numel(xRange), 1);

cvx_solver sedumi
cvx_solver sdpt3
```

```
cvx_quiet true
cvx_begin
  variables kp(n+1)
  minimize fobjective(kp);
  subject to
    Aineq*kp <= bineq
cvx_end

%-------------------------------------------------------------------------
% NONCVX Optimize
%-------------------------------------------------------------------------
%X_CON = [zeros(numel(xRange),1)];
%for i=1:n
  %X_CON(:,end+1) = xRange.^i;
%end

%Aineq = -diag([sign(xRange)])*X_CON;
%bineq = zeros(numel(xRange), 1);

%guess = zeros(1, n+1);
%fminObj =@(x) fobjective(x');
%[x, fval] = fmincon(fminObj, guess, Aineq, bineq, [], []);
%kp = x';
%cvx_optval = fval;

%-----------------------------
% Store Results
%-----------------------------
PEA.order = n;
PEA.kp = kp';

PEA.Tm = Tm_fun(kp);
PEA.T1m = T1m_fun(kp);
PEA.T2m = T2m_fun(kp);

PEA.Pm = Pm_fun(kp);

PEA.Tp = Tp_fun(kp);
PEA.T1p = T1p_fun(kp);
PEA.Pp = Pp_fun(kp);

PEA.Em_pos = Em_pos_fun(kp);
PEA.Em_neg = Em_neg_fun(kp);

PEA.Em_actual_pos = Em_actual_pos_fun(kp);
PEA.Em_actual_neg = Em_actual_neg_fun(kp);
PEA.Em_actual_net = Em_actual_pos_fun(kp)-Em_actual_neg_fun(kp);

PEA.E_cost = E_cost_fun(kp);
PEA.P_cost = P_cost_fun(kp);
PEA.T_cost = T_cost_fun(kp);

PEA.fobj_val = cvx_optval;
PEA.real_cost = real_cost_fun(kp);

PEA.Tp_range = Tp_range_fun(kp);

PEA.Tp_scaler_fun =@(x) -(x.^[0:1:n])*kp;
PEA.Tp_fun =@(x) arrayfun(PEA.Tp_scaler_fun, x);
```

A.3 Cam curve generation source code

The following Matlab code generates the cam curve for manufacturing, used for the cam presented in Sect. 3.4.2.

```
%-------------------------------------------------------------------------
% Prosethetic Ankle-Foot
%
% -- make cam shape --
%
% Make a cam given kp*
```

```
%
% v4 - 17 June 2013
%-------------------------------------------------------------------------
function CAM = make_cam(t,n,mass,kp,keff,Rp,ecc,Rf)

%-------------------------------------------------------------------------
% Discretize Ankle Angle
%-------------------------------------------------------------------------
dx = (pi/1000);

% Discrete Circle
xFull = (-pi):dx:(pi-dx);

% Define max/min +- 5 deg.
safeMax = max(t.xd) + deg2rad(10);
safeMin = min(t.xd) - deg2rad(10);

% Find safe data points
maxInd = find( (xFull > (safeMax - dx/2)) & (xFull < (safeMax + dx/2)) );
minInd = find( (xFull > (safeMin - dx/2)) & (xFull < (safeMin + dx/2)) );

% Find active data points
activeMaxInd = find( (xFull > (max(t.xd) - dx/2)) & ...
                     (xFull < (max(t.xd) + dx/2)));
activeMinInd = find( (xFull > (min(t.xd) - dx/2)) & ...
                     (xFull < (min(t.xd) + dx/2)));


% Descrete Active region
xActive = xFull(minInd:maxInd);

%-------------------------------------------------------------------------
% Define Polynomial Tp^* = p(x) and p_bar(x) = int p(x) dx
%-------------------------------------------------------------------------
p_bar_fun =@(x) mass*(1./[1:1:(n+1)].*x.^[1:1:(n+1)])*kp;
p_fun =@(x) mass*(x.^[0:1:n])*kp;

%-------------------------------------------------------------------------
% Find constant c
%-------------------------------------------------------------------------
c = -1.1*min( (2/keff)*arrayfun(p_bar_fun, xActive) );

%-------------------------------------------------------------------------
% Define follower deflection s = f(x) and s' = df(x)/dx
%-------------------------------------------------------------------------
s_fun =@(x) sqrt((2/keff).*p_bar_fun(x) + c);

s_active = arrayfun(s_fun, xActive);

%-------------------------------------------------------------------------
% Fit with splines
%-------------------------------------------------------------------------
fit_y =[0, 0, s_active, 0, 0];
fit_x = [xFull(1), xFull(20), xFull(minInd:maxInd), xFull(end-20), xFull(end)];
pp = spline(fit_x,fit_y);
s = ppval(pp, xFull);
s_t_fun =@(x) ppval(pp, x);

[breaks,coefs,l,k,d] = unmkpp(pp);
pp2 = mkpp(breaks,repmat(k-1:-1:1,d*l,1).*coefs(:,1:k-1),d);
s_1 = ppval(pp2,xFull);
s_1t_fun =@(x) ppval(pp2, x);

[breaks,coefs,l,k,d] = unmkpp(pp2);
pp3 = mkpp(breaks,repmat(k-1:-1:1,d*l,1).*coefs(:,1:k-1),d);
s_2 = ppval(pp3,xFull);

[breaks,coefs,l,k,d] = unmkpp(pp3);
pp4 = mkpp(breaks,repmat(k-1:-1:1,d*l,1).*coefs(:,1:k-1),d);
s_3 = ppval(pp4,xFull);

if(1)
figure;
  plot(xFull, s), hold all
  plot(xActive, s(minInd:maxInd)), hold all
  plot(xFull(maxInd), s(maxInd), 'o')
  plot(xFull(minInd), s(minInd), 'o')
  maxSafeDeg = rad2deg(xFull(maxInd))
  minSafeDeg = rad2deg(xFull(minInd))
```

```
86    maxActiveDeg = rad2deg(xFull(activeMaxInd))
87    minActiveDeg = rad2deg(xFull(activeMinInd))
88    startpoint = [xFull(1), s(1)]
89    endpoint = [xFull(end), s(end)]
90  end
91  %-------------------------------------------------------------------------
92  % Pressure Angle
93  %-------------------------------------------------------------------------
94  phi = atan( (s_1 - ecc)./(s + sqrt(Rp^2 - ecc^2)) );
95
96  phi_active_max = max(abs(phi(activeMinInd:activeMaxInd)));
97  phi_safe_max = max(abs(phi(minInd:maxInd)));
98  %-------------------------------------------------------------------------
99  % Cam Profile
100 %-------------------------------------------------------------------------
101 d = sqrt(Rp^2 - ecc^2);
102 R = sqrt( (d+s).^2 + ecc^2 );
103 gamm = atan(ecc./(d + s));
104 lambda = gamm + xFull;
105
106 xf = R.*cos(lambda);
107 yf = -R.*sin(lambda);
108
109
110 sigma = phi - xFull;
111
112 xs_out = xf + Rf.*cos(sigma);
113 xs_in = xf - Rf.*cos(sigma);
114
115 ys_out = yf + Rf.*sin(sigma);
116 ys_in = yf - Rf.*sin(sigma);
117
118 startcam = [xs_in(1), ys_in(1)]
119 endcam = [xs_in(end), ys_in(end)]
120
121 %-------------------------------------------------------------------------
122 % Return CAM stuff
123 %-------------------------------------------------------------------------
124 CAM.c = c;
125
126 CAM.max_active_def = max(s(activeMinInd:activeMaxInd));
127 CAM.min_active_def = min(s(activeMinInd:activeMaxInd));
128
129 CAM.max_safe_def = max(s(minInd:maxInd));
130 CAM.min_safe_def = min(s(minInd:maxInd));
131
132 CAM.s = s;
133 CAM.s_1 = s_1;
134 CAM.s_2 = s_2;
135 CAM.s_3 = s_3;
136
137 CAM.phi = phi;
138 CAM.max_active_phi = phi_active_max;
139 CAM.max_active_phi_deg = rad2deg(phi_active_max);
140 CAM.max_safe_phi = phi_safe_max;
141 CAM.max_safe_phi_deg = rad2deg(phi_safe_max);
142
143 CAM.follower = [xf; yf];
144 CAM.outer = [xs_out; ys_out];
145 CAM.inner = [xs_in; ys_in];
146
147 CAM.R_profile = R;
148 CAM.lambda_profile = lambda;
149
150 CAM.maxInd = maxInd;
151 CAM.minInd = minInd;
152
153 CAM.safeMinInd = maxInd;
154 CAM.safeMaxInd = minInd;
155
156 CAM.activeMaxInd = activeMaxInd;
157 CAM.activeMinInd = activeMinInd;
158
159 CAM.xFull = xFull;
160 CAM.dx = dx;
161
162 CAM.s_t_fun =@(x) s_t_fun(x);
163 CAM.s_1t_fun =@(x) s_1t_fun(x);
164
165 phi_fun =@(x) atan( (s_1t_fun(x) - ecc)./(s_t_fun(x) + sqrt(Rp^2 - ecc^2)) );
```

```
166  CAM.phi_ = phi_fun(t.xd);
167
168  %-------------------------------------------------------------------------
169  % Forces
170  %-------------------------------------------------------------------------
171  a0 = 0.40;
172  b = 0.57;
173  a = a0 + s_t_fun(t.xd);
174  d =0.012;
175
176  CAM.a = a;
177  CAM.Fs = keff*s_t_fun(t.xd);
178  CAM.N = -CAM.Fs./cos(phi_fun(t.xd));
179
180  CAM.R2 = ((a+b).*CAM.N.*sin(phi_fun(t.xd)) ...
181           - (d/2).*(CAM.Fs - CAM.N.*cos(phi_fun(t.xd))))/b;
182  CAM.R1 = CAM.R2 - CAM.N.*sin(phi_fun(t.xd));
183
184  CAM.Tc = -keff*s_t_fun(t.xd).*s_1t_fun(t.xd);
185  CAM.Tc_fun =@(x) -keff*s_t_fun(x).*s_1t_fun(x);
186
187  CAM.Fs_max_safe = max(keff*s(minInd:maxInd));
188  CAM.Fs_min_safe = min(keff*s(minInd:maxInd));
189
190  CAM.Fs_max_active = max(keff*s(activeMinInd:activeMaxInd));
191  CAM.Fs_min_active = min(keff*s(activeMinInd:activeMaxInd));
192
193  CAM.maxR1 = max(abs(CAM.R1));
194  CAM.maxR2 = max(abs(CAM.R2));
```

Appendix B

BIOMECHANICS TOOLBOX (BIOMECHTB)

B.1 Data processing Matlab code

This code processes Vicon inverse dynamics, ground reaction force, and subject data files, and stores them as Matlab structures. The processing is detailed in Sect.

B.1.1 vicon_process_trial.m

```
function rtn = vicon_process_trial(trialName,varargin)
  % This function processes vicon + embedded files

  nVarArgs = length(varargin);

  % Defaults
  processEmb = 1;

  % Motor 2
  k_t = 0.00775;
  k_s = 1223;
  Vcc = 60;
  R_T = 1800;
  maxMotorAmp = 14;
  maxMotorRpm = 80000;


  % Motor 1
%  k_t = 0.0053;
%  k_s = 1730;
%  Vcc = 30;
%  maxMotorAmp = 20;
%  maxMotorRpm = 40000;
%  R_T = 472;


  modelCutOff = 6;
  GRFCutOff = 50;
  embCutOff = 50;
  eventThreshold = 50;

  for i=1:2:nVarArgs
    switch varargin{i}
      case 'processEmb'
        processEmb = varargin{i+1};
      case 'k_t'
        % Torque Constant (Nm/A)
        k_t = varargin{i+1};
      case 'k_s'
        % Speed Constant (rpm/V)
        k_s = varargin{i+1};
      case 'Vcc'
        % Operating Voltage (V)
        Vcc = varargin{i+1};
      case 'R_T'
        % Total tranmission ratio
        R_T = varargin{i+1};
      case 'maxMotorAmp'
        % Max Motor Current (A) -- setting on ESCON 70/10 servo controller
        maxMotorAmp = varargin{i+1};
```

```
51          case 'maxMotorRpm'
52            % Max Motor Speed (Rpm) -- setting on ESCON 70/10 servo controller
53            maxMotorRpm = varargin{i+1};
54          case 'modelCutOff'
55            modelCutOff = varargin{i+1};
56          case 'GRFCutOff'
57            GRFCutOff = varargin{i+1};
58          case 'embCutOff'
59            embCutOff = varargin{i+1};
60          case 'eventThreshold'
61            eventThreshold = varargin{i+1};
62          otherwise
63            fprintf('\n%s option not found!\n',varargin{i});
64            return
65        end
66      end
67
68      % Process subject params
69      rtn.subject = vicon_process_subject(trialName);
70
71      % Process inverse dynamics model
72      model = vicon_process_model(trialName, ...
73                                  'cutOff',modelCutOff);
74
75      rtn.info.model = model.info;
76      rtn.params.model = model.params;
77      rtn.time = model.time;
78      rtn.model = model.id;
79      rtn.markers = model.markers;
80
81      % Process vGRF
82      grf = vicon_process_grf(trialName, ...
83                              'cutOff',GRFCutOff, ...
84                              'eventThreshold',eventThreshold);
85
86      rtn.info.grf = grf.info;
87      rtn.params.grf = grf.params;
88      rtn.grf.time = grf.time;
89      rtn.grf.r = grf.r;
90      rtn.grf.l = grf.l;
91      ge = grf.ge;
92
93      % Process HS
94      for foot = {'l','r'}
95
96        % Remove TO before first HS
97        ge.(foot{:}).to_time(ge.(foot{:}).to_time < ge.(foot{:}).hs_time(1)) = [];
98
99        % Remove TO after last HS
100       ge.(foot{:}).to_time(ge.(foot{:}).to_time > ge.(foot{:}).hs_time(end)) = [];
101
102       % Reformat
103       % segment.l.time = [HS1, HS2,
104       %                                 HS2, HS3,
105       %                                 HS3, HS4]
106       segment.(foot{:}).gait.time =  ...
107         [ge.(foot{:}).hs_time(1:end-1), ge.(foot{:}).hs_time(2:end)];
108
109       segment.(foot{:}).stance.time =  ...
110         [ge.(foot{:}).hs_time(1:end-1), ge.(foot{:}).to_time(1:end)];
111
112       segment.(foot{:}).swing.time =  ...
113         [ge.(foot{:}).to_time(1:end), ge.(foot{:}).hs_time(2:end)];
114
115     end
116
117     rtn.ge.grf = ge;
118     rtn.segment.grf = segment;
119
120     % Process embedded data
121     if (processEmb)
122       emb = embedded_process_data(trialName, ...
123                                   'k_t',k_t, ...
124                                   'k_s',k_s, ...
125                                   'Vcc',Vcc, ...
126                                   'R_T',R_T, ...
127                                   'maxMotorAmp',maxMotorAmp, ...
128                                   'maxMotorRpm',maxMotorRpm, ...
129                                   'cutOff',embCutOff);
130
131       rtn.params.emb = emb.params;
```

```
132       rtn.info.emb = emb.info;
133       rtn.control = emb.control;
134       rtn.ge.emb = emb.ge;
135       rtn.emb = emb.data;
136     end
137
138
139     %
140     %
141     % Time normalize -- Whole Gait Cycle
142     %
143     %
144
145
146     % Gait cycle
147     gaitCycle = linspace(0,100,1001);
148     rtn.gait.gaitCycle = gaitCycle;
149
150     % Field names
151     q_name = fieldnames(rtn.model);
152     q_grf = fieldnames(rtn.grf.l);
153
154     % Process embedded data
155     if (processEmb)
156       q_control = fieldnames(rtn.control);
157     end
158
159     % Time Normalization
160     for foot = {'l','r'}
161
162       hs_time = rtn.segment.grf.(foot{:}).gait.time;
163
164       for i=1:numel(hs_time(:,1))
165
166         % If recorded hs occurs before model data, skip
167         if (hs_time(i,1) < rtn.time(1)) | (hs_time(i,2) > rtn.time(end))
168           continue;
169         end
170
171         % Find vicon time stamp corresponding to HS
172         temp = rtn.time - hs_time(i,1);
173         temp(temp < 0) = inf;
174         [~,ii] = min(temp);
175         hs1 = ii;
176
177         temp = rtn.time - hs_time(i,2);
178         temp(temp < 0) = inf;
179         [~,ii] = min(temp);
180         hs2 = ii;
181
182         rawTime = rtn.time(hs1:hs2);
183         rawGaitCylce = (rawTime - rawTime(1))./(rawTime(end) - rawTime(1))*100;
184
185         rtn.gait.model.normalized.(foot{:}).time{i} = rawTime;
186
187         % Interpolate inverse dynamics (vicon)
188         for j=1:numel(q_name)
189           rtn.gait.model.normalized.(foot{:}).(q_name{j}).X(:,i) = ...
190             interp1(rawGaitCylce, rtn.model.(q_name{j}).X(hs1:hs2), ...
191                     gaitCycle, 'spline');
192           rtn.gait.model.normalized.(foot{:}).(q_name{j}).Y(:,i) = ...
193              interp1(rawGaitCylce, rtn.model.(q_name{j}).Y(hs1:hs2), ...
194                      gaitCycle, 'spline');
195           rtn.gait.model.normalized.(foot{:}).(q_name{j}).Z(:,i) = ...
196              interp1(rawGaitCylce, rtn.model.(q_name{j}).Z(hs1:hs2), ...
197                      gaitCycle, 'spline');
198         end
199
200         % Find grf time stamp corresponding to HS
201         temp = rtn.grf.time - hs_time(i,1);
202         temp(temp < 0) = inf;
203         [~,ii] = min(temp);
204         hs1 = ii;
205
206         temp = rtn.grf.time - hs_time(i,2);
207         temp(temp < 0) = inf;
208         [~,ii] = min(temp);
209         hs2 = ii;
210
```

```
211         rawTime = rtn.grf.time(hs1:hs2);
212         rawGaitCylce = (rawTime - rawTime(1))./(rawTime(end) - rawTime(1))*100;
213 
214         rtn.gait.grf.normalized.(foot{:}).grf.time{i} = rawTime;
215 
216         % Interpolate vGRF
217         for j=1:numel(q_grf)
218 
219           rtn.gait.grf.normalized.(foot{:}).(['L',q_grf{j}])(:,i) = ...
220             interp1(rawGaitCylce, rtn.grf.l.(q_grf{j})(hs1:hs2), ...
221                     gaitCycle, 'spline');
222           rtn.gait.grf.normalized.(foot{:}).(['R',q_grf{j}])(:,i) = ...
223             interp1(rawGaitCylce, rtn.grf.r.(q_grf{j})(hs1:hs2), ...
224                     gaitCycle, 'spline');
225         end
226 
227 
228         % Interpolate embedded system data
229         if (processEmb)
230 
231           % Find emb time stamp corresponding to HS
232           temp = rtn.control.time - hs_time(i,1);
233           temp(temp < 0) = inf;
234           [~,ii] = min(temp);
235           hs1 = ii;
236 
237           temp = rtn.control.time - hs_time(i,2);
238           temp(temp < 0) = inf;
239           [~,ii] = min(temp);
240           hs2 = ii;
241 
242           rawTime = rtn.control.time(hs1:hs2);
243           rawGaitCylce = (rawTime - rawTime(1))./(rawTime(end) - rawTime(1))*100;
244 
245           rtn.gait.emb.normalized.(foot{:}).control.time{i} = rawTime;
246 
247           for j=2:numel(q_control)
248             rtn.gait.emb.normalized.(foot{:}).(q_control{j})(:,i) = ...
249               interp1(rawGaitCylce, rtn.control.(q_control{j})(hs1:hs2), ...
250                       gaitCycle, 'linear');
251           end
252         end
253 
254       end
255 
256       % Mean, std
257       for j=1:numel(q_name)
258          jointVarMean_X =  ...
259               mean(rtn.gait.model.normalized.(foot{:}).(q_name{j}).X, 2);
260          jointVarMean_Y = ...
261               mean(rtn.gait.model.normalized.(foot{:}).(q_name{j}).Y, 2);
262          jointVarMean_Z =  ...
263               mean(rtn.gait.model.normalized.(foot{:}).(q_name{j}).Z, 2);
264 
265          jointVarStd_X =  ...
266               std(rtn.gait.model.normalized.(foot{:}).(q_name{j}).X')';
267          jointVarStd_Y = ...
268               std(rtn.gait.model.normalized.(foot{:}).(q_name{j}).Y')';
269          jointVarStd_Z = ...
270                std(rtn.gait.model.normalized.(foot{:}).(q_name{j}).Z')';
271 
272          rtn.gait.model.(foot{:}).(q_name{j}).X = ...
273                                    [jointVarMean_X + jointVarStd_X, ...
274                                     jointVarMean_X, ...
275                                     jointVarMean_X - jointVarStd_X];
276 
277          rtn.gait.model.(foot{:}).(q_name{j}).Y = ...
278                                     [jointVarMean_Y + jointVarStd_Y, ...
279                                      jointVarMean_Y, ...
280                                      jointVarMean_Y - jointVarStd_Y];
281 
282          rtn.gait.model.(foot{:}).(q_name{j}).Z = ...
283                                     [jointVarMean_Z + jointVarStd_Z, ...
284                                      jointVarMean_Z, ...
285                                      jointVarMean_Z - jointVarStd_Z];
286       end
287 
288       % Mean, std
289       for j=1:numel(q_grf)
```

```
290         varMean = ...
291         mean(rtn.gait.grf.normalized.(foot{:}).(['L',q_grf{j}]), 2);
292
293         varStd = ...
294           std(rtn.gait.grf.normalized.(foot{:}).(['L',q_grf{j}])')';
295
296         rtn.gait.grf.(foot{:}).(['L',q_grf{j}]) = [varMean + varStd, ...
297                                                    varMean, ...
298                                                    varMean - varStd];
299
300         varMean = ...
301         mean(rtn.gait.grf.normalized.(foot{:}).(['R',q_grf{j}]), 2);
302
303         varStd = ...
304           std(rtn.gait.grf.normalized.(foot{:}).(['R',q_grf{j}])')';
305
306         rtn.gait.grf.(foot{:}).(['R',q_grf{j}]) = [varMean + varStd, ...
307                                                    varMean, ...
308                                                    varMean - varStd];
309       end
310
311       % Mean, std
312       if (processEmb)
313        for j=2:numel(q_control)
314         varMean = mean(rtn.gait.emb.normalized.(foot{:}).(q_control{j}), 2);
315         varStd = std(rtn.gait.emb.normalized.(foot{:}).(q_control{j})')';
316
317         rtn.gait.emb.(foot{:}).(q_control{j}) = [varMean + varStd, ...
318                                                  varMean, ...
319                                                  varMean - varStd];
320        end
321       end
322
323     end
324
325
326     %
327     %
328     % Time normalize -- Stance Cycle
329     %
330     %
331
332
333     % Gait cycle
334     stanceCycle = linspace(0,100,1001);
335     rtn.stance.stanceCycle = stanceCycle;
336
337     % Time Normalization
338     for foot = {'l','r'}
339
340       hs_time = rtn.segment.grf.(foot{:}).stance.time;
341
342       for i=1:numel(hs_time(:,1))
343
344         % If recorded hs occurs before model data, skip
345         if hs_time(i,1) < rtn.time(1)
346           continue;
347         end
348
349         % Find vicon time stamp corresponding to HS
350         temp = rtn.time - hs_time(i,1);
351         temp(temp < 0) = inf;
352         [~,ii] = min(temp);
353         hs1 = ii;
354
355         temp = rtn.time - hs_time(i,2);
356         temp(temp < 0) = inf;
357         [~,ii] = min(temp);
358         hs2 = ii;
359
360         rawTime = rtn.time(hs1:hs2);
361         rawStanceCylce = (rawTime - rawTime(1))./(rawTime(end) - rawTime(1))*100;
362
363         rtn.gait.model.normalized.(foot{:}).time{i} = rawTime;
364
365         % Interpolate inverse dynamics (vicon)
366         for j=1:numel(q_name)
367           rtn.stance.model.normalized.(foot{:}).(q_name{j}).X(:,i) = ...
368             interp1(rawStanceCylce, rtn.model.(q_name{j}).X(hs1:hs2), ...
369                     stanceCycle, 'spline');
370           rtn.stance.model.normalized.(foot{:}).(q_name{j}).Y(:,i) = ...
```

```
371             interp1(rawStanceCylce, rtn.model.(q_name{j}).Y(hs1:hs2), ...
372                     stanceCycle, 'spline');
373           rtn.stance.model.normalized.(foot{:}).(q_name{j}).Z(:,i) = ...
374             interp1(rawStanceCylce, rtn.model.(q_name{j}).Z(hs1:hs2), ...
375                     stanceCycle, 'spline');
376         end
377
378         % Find grf time stamp corresponding to HS
379         temp = rtn.grf.time - hs_time(i,1);
380         temp(temp < 0) = inf;
381         [~,ii] = min(temp);
382         hs1 = ii;
383
384         temp = rtn.grf.time - hs_time(i,2);
385         temp(temp < 0) = inf;
386         [~,ii] = min(temp);
387         hs2 = ii;
388
389         rawTime = rtn.grf.time(hs1:hs2);
390         rawStanceCylce = (rawTime - rawTime(1))./(rawTime(end) - rawTime(1))*100;
391
392         rtn.stance.grf.normalized.(foot{:}).grf.time{i} = rawTime;
393
394         % Interpolate vGRF
395         for j=1:numel(q_grf)
396
397           rtn.stance.grf.normalized.(foot{:}).(['L',q_grf{j}])(:,i) = ...
398             interp1(rawStanceCylce, rtn.grf.l.(q_grf{j})(hs1:hs2), ...
399                     stanceCycle, 'spline');
400           rtn.stance.grf.normalized.(foot{:}).(['R',q_grf{j}])(:,i) = ...
401             interp1(rawStanceCylce, rtn.grf.r.(q_grf{j})(hs1:hs2), ...
402                     stanceCycle, 'spline');
403         end
404
405
406         % Interpolate embedded system
407         if (processEmb)
408
409           % Find emb time stamp corresponding to HS
410           temp = rtn.control.time - hs_time(i,1);
411           temp(temp < 0) = inf;
412           [~,ii] = min(temp);
413           hs1 = ii;
414
415           temp = rtn.control.time - hs_time(i,2);
416           temp(temp < 0) = inf;
417           [~,ii] = min(temp);
418           hs2 = ii;
419
420           rawTime = rtn.control.time(hs1:hs2);
421           rawStanceCylce = (rawTime - rawTime(1))./(rawTime(end) - rawTime(1))*100;
422
423           rtn.stance.emb.normalized.(foot{:}).control.time{i} = rawTime;
424
425           for j=2:numel(q_control)
426             rtn.stance.emb.normalized.(foot{:}).(q_control{j})(:,i) = ...
427               interp1(rawStanceCylce, rtn.control.(q_control{j})(hs1:hs2), ...
428                       stanceCycle, 'linear');
429           end
430         end
431
432       end
433
434       % Mean, std
435       for j=1:numel(q_name)
436          jointVarMean_X =  ...
437               mean(rtn.stance.model.normalized.(foot{:}).(q_name{j}).X, 2);
438           jointVarMean_Y = ...
439               mean(rtn.stance.model.normalized.(foot{:}).(q_name{j}).Y, 2);
440           jointVarMean_Z =  ...
441               mean(rtn.stance.model.normalized.(foot{:}).(q_name{j}).Z, 2);
442
443           jointVarStd_X =  ...
444               std(rtn.stance.model.normalized.(foot{:}).(q_name{j}).X')';
445           jointVarStd_Y = ...
446               std(rtn.stance.model.normalized.(foot{:}).(q_name{j}).Y')';
447           jointVarStd_Z = ...
448                std(rtn.stance.model.normalized.(foot{:}).(q_name{j}).Z')';
```

```
449
450         rtn.stance.model.(foot{:}).(q_name{j}).X = ...
451                                 [jointVarMean_X + jointVarStd_X, ...
452                                  jointVarMean_X, ...
453                                  jointVarMean_X - jointVarStd_X];
454
455         rtn.stance.model.(foot{:}).(q_name{j}).Y = ...
456                                 [jointVarMean_Y + jointVarStd_Y, ...
457                                  jointVarMean_Y, ...
458                                  jointVarMean_Y - jointVarStd_Y];
459
460         rtn.stance.model.(foot{:}).(q_name{j}).Z = ...
461                                 [jointVarMean_Z + jointVarStd_Z, ...
462                                  jointVarMean_Z, ...
463                                  jointVarMean_Z - jointVarStd_Z];
464       end
465
466       % Mean, std
467       for j=1:numel(q_grf)
468          varMean = ...
469          mean(rtn.stance.grf.normalized.(foot{:}).(['L',q_grf{j}]), 2);
470
471          varStd = ...
472            std(rtn.stance.grf.normalized.(foot{:}).(['L',q_grf{j}])')';
473
474          rtn.stance.grf.(foot{:}).(['L',q_grf{j}]) = [varMean + varStd, ...
475                                                       varMean, ...
476                                                       varMean - varStd];
477
478          varMean = ...
479          mean(rtn.stance.grf.normalized.(foot{:}).(['R',q_grf{j}]), 2);
480
481          varStd = ...
482            std(rtn.stance.grf.normalized.(foot{:}).(['R',q_grf{j}])')';
483
484          rtn.stance.grf.(foot{:}).(['R',q_grf{j}]) = [varMean + varStd, ...
485                                                       varMean, ...
486                                                       varMean - varStd];
487       end
488
489
490       if (processEmb)
491        for j=2:numel(q_control)
492          varMean = mean(rtn.stance.emb.normalized.(foot{:}).(q_control{j}), 2);
493          varStd = std(rtn.stance.emb.normalized.(foot{:}).(q_control{j})')';
494
495          rtn.stance.emb.(foot{:}).(q_control{j}) = [varMean + varStd, ...
496                                                     varMean, ...
497                                                     varMean - varStd];
498        end
499       end
500
501     end
502
503     figure;
504       subplot(311); hold all;
505         plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.r.RAnkleAngles.X,'r', ...
506           'lineWidth',0.1);
507         plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.l.LAnkleAngles.X,'k', ...
508           'lineWidth',0.1);
509         grid on
510       subplot(312); hold all;
511         plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.r.RAnkleMoment.X,'r', ...
512           'lineWidth',0.1);
513         plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.l.LAnkleMoment.X,'k', ...
514           'lineWidth',0.1);
515         grid on
516       subplot(313); hold all;
517         plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.r.RAnklePower.X,'r', ...
518           'lineWidth',0.1);
519         plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.l.LAnklePower.X,'k', ...
520           'lineWidth',0.1);
521         grid on
522   end
```

B.1.2 vicon_process_model.m

```
function rtn = vicon_process_model(trialName,varargin)

  % This function parses vicon model files

  nVarArgs = length(varargin);

  % Defaults
  cutOff = 6;

  for i=1:2:nVarArgs
    switch varargin{i}
      case 'cutOff'
        cutOff = varargin{i+1};
      otherwise
        fprintf('\n%s option not found!\n',varargin{i});
        return
    end
  end

  % Plots
  debug = 0;

  % Parse model file
  file = ['./',trialName,'_Model.csv'];
  if exist(file,'file') ~= 2
    fprintf('\n\tModel file not found: %s\n', file);
    rtn = []
    return
  end

  fid = fopen(file,'r');

  % Store trialName
  rtn.info.file = file;
  rtn.params.processing_cutOff = cutOff;

  % skip two words, go next line, grab sample freq
  fs = fscanf(fid, '%*s %*s\n %f', 1);

  rtn.params.fs = fs;

  % skip line
  tline = fgetl(fid);

  % parse labels and create struct fields names
  tline = fgetl(fid);
  labels = textscan(tline, '%s', 'delimiter', ',');
  [m, n] = size(labels{1});
  labelCnt = 0;
  for i=1:m
    if ~(strcmp(labels{1}{i},''))
      % pasre header on label
      [token, remain] = strtok(labels{1}{i},':');
      labelCnt = labelCnt + 1;
      labelName{labelCnt} = remain(2:end);
    end
  end

  % find number of subfields
  tline = fgetl(fid);
  sublabels = textscan(tline, '%s', 'delimiter', ',');
  [m, n] = size(sublabels{1});

  % skip line of units
  fgetl(fid);

  % put data in matrix
  D = zeros(1,m);
  formatSpec = [repmat(['%f'],1,m-1), '%f\n'];
  rowCount = 1;
  while(1)
    C = textscan(fid, formatSpec, 'delimiter', ',');
    dataRow = cell2mat(C);
    if isempty(dataRow)
      break;
    end
    D(rowCount,:) = cell2mat(C);
    rowCount = rowCount + 1;
  end
```

```
80
81      % Time = (frame number - 1) x dt
82      rtn.time = (D(:,1) - 1)*(1/fs);
83
84      % Butterworth 4th order
85      [b,a] = butter(4,2*(cutOff/fs));
86
87      % put data (X,Y,Z subfields per field)
88      % Fix units
89      % Pattern is (L/R)(joint)Angle, Moment, Power
90      for i=1:(labelCnt/3)
91
92        % Angle ------------------------------------------------------------------
93        jointAngle_X = deg2rad(D(:,3 + (9*i - 9)));
94        jointAngle_Y = deg2rad(D(:,4 + (9*i - 9)));
95        jointAngle_Z = deg2rad(D(:,5 + (9*i - 9)));
96
97        % Set NaN to zero
98        jointAngle_X(isnan(jointAngle_X)) = 0;
99        jointAngle_Y(isnan(jointAngle_Y)) = 0;
100       jointAngle_Z(isnan(jointAngle_Z)) = 0;
101
102       % Filter
103       jointAngle_X = filtfilt(b,a,jointAngle_X);
104       jointAngle_Y = filtfilt(b,a,jointAngle_Y);
105       jointAngle_Z = filtfilt(b,a,jointAngle_Z);
106
107       d_jointAngle_X = gradient(jointAngle_X)*fs;
108       d_jointAngle_Y = gradient(jointAngle_Y)*fs;
109       d_jointAngle_Z = gradient(jointAngle_Z)*fs;
110
111       d2_jointAngle_X = gradient(d_jointAngle_X)*fs;
112       d2_jointAngle_Y = gradient(d_jointAngle_Y)*fs;
113       d2_jointAngle_Z = gradient(d_jointAngle_Z)*fs;
114
115       % Moment -----------------------------------------------------------------
116       jointMoment_X = -D(:,6 + (9*i - 9))/1000;
117       jointMoment_Y = -D(:,7 + (9*i - 9))/1000;
118       jointMoment_Z = -D(:,8 + (9*i - 9))/1000;
119
120       % Set NaN to zero
121       jointMoment_X(isnan(jointMoment_X))= 0;
122       jointMoment_Y(isnan(jointMoment_Y))= 0;
123       jointMoment_Z(isnan(jointMoment_Z))= 0;
124
125       % Filter
126       jointMoment_X = filtfilt(b,a,jointMoment_X);
127       jointMoment_Y = filtfilt(b,a,jointMoment_Y);
128       jointMoment_Z = filtfilt(b,a,jointMoment_Z);
129
130       d_jointMoment_X = gradient(jointMoment_X)*fs;
131       d_jointMoment_Y = gradient(jointMoment_Y)*fs;
132       d_jointMoment_Z = gradient(jointMoment_Z)*fs;
133
134       d2_jointMoment_X = gradient(d_jointMoment_X)*fs;
135       d2_jointMoment_Y = gradient(d_jointMoment_Y)*fs;
136       d2_jointMoment_Z = gradient(d_jointMoment_Z)*fs;
137
138       % Power (calculate) ------------------------------------------------------
139       jointPower_X = d_jointAngle_X.*jointMoment_X;
140       jointPower_Y = d_jointAngle_Y.*jointMoment_Y;
141       jointPower_Z = d_jointAngle_Z.*jointMoment_Z;
142
143       d_jointPower_X = gradient(jointPower_X)*fs;
144       d_jointPower_Y = gradient(jointPower_Y)*fs;
145       d_jointPower_Z = gradient(jointPower_Z)*fs;
146
147       d2_jointPower_X = gradient(d_jointPower_X)*fs;
148       d2_jointPower_Y = gradient(d_jointPower_Y)*fs;
149       d2_jointPower_Z = gradient(d_jointPower_Z)*fs;
150
151       % Store Angle info -------------------------------------------------------
152       rtn.id.(labelName{1 + (3*i - 3)}).X = jointAngle_X;
153       rtn.id.(labelName{1 + (3*i - 3)}).Y = jointAngle_Y;
154       rtn.id.(labelName{1 + (3*i - 3)}).Z = jointAngle_Z;
155
156       % 1st time derivative
157       rtn.id.(['d_',labelName{1 + (3*i - 3)}]).X = d_jointAngle_X;
```

```
158       rtn.id.(['d_',labelName{1 + (3*i - 3)}]).Y = d_jointAngle_Y;
159       rtn.id.(['d_',labelName{1 + (3*i - 3)}]).Z = d_jointAngle_Z;
160
161       % 2nd time derivative
162       rtn.id.(['d2_',labelName{1 + (3*i - 3)}]).X = d2_jointAngle_X;
163       rtn.id.(['d2_',labelName{1 + (3*i - 3)}]).Y = d2_jointAngle_Y;
164       rtn.id.(['d2_',labelName{1 + (3*i - 3)}]).Z = d2_jointAngle_Z;
165
166       if(debug)
167         figure;
168           subplot(311); hold all;
169             title(['Saggital Plane ', labelName{1 + (3*i - 3)}],'fontsize',20)
170             plot(jointAngle_X,'k')
171           subplot(312); hold all;
172             plot(d_jointAngle_X,'k')
173           subplot(313); hold all;
174             plot(d2_jointAngle_X,'k')
175       end
176
177       % Store Moment info ------------------------------------------------------
178       rtn.id.(labelName{2 + (3*i - 3)}).X = jointMoment_X;
179       rtn.id.(labelName{2 + (3*i - 3)}).Y = jointMoment_Y;
180       rtn.id.(labelName{2 + (3*i - 3)}).Z = jointMoment_Z;
181
182       % 1st time derivative
183       rtn.id.(['d_',labelName{2 + (3*i - 3)}]).X = d_jointMoment_X;
184       rtn.id.(['d_',labelName{2 + (3*i - 3)}]).Y = d_jointMoment_Y;
185       rtn.id.(['d_',labelName{2 + (3*i - 3)}]).Z = d_jointMoment_Z;
186
187       % 2nd time derivative
188       rtn.id.(['d2_',labelName{2 + (3*i - 3)}]).X = d2_jointMoment_X;
189       rtn.id.(['d2_',labelName{2 + (3*i - 3)}]).Y = d2_jointMoment_Y;
190       rtn.id.(['d2_',labelName{2 + (3*i - 3)}]).Z = d2_jointMoment_Z;
191
192       if(debug)
193         figure;
194           subplot(311); hold all;
195             title(['Saggital Plane ', labelName{2 + (3*i - 3)}],'fontsize',20)
196             plot(jointMoment_X,'k')
197           subplot(312); hold all;
198             plot(d_jointMoment_X,'k')
199           subplot(313); hold all;
200             plot(d2_jointMoment_X,'k')
201       end
202
203       % Store Power info -------------------------------------------------------
204       rtn.id.(labelName{3 + (3*i - 3)}).X = jointPower_X;
205       rtn.id.(labelName{3 + (3*i - 3)}).Y = jointPower_Y;
206       rtn.id.(labelName{3 + (3*i - 3)}).Z = jointPower_Z;
207
208       % 1st time derivative
209       rtn.id.(['d_',labelName{3 + (3*i - 3)}]).X = d_jointPower_X;
210       rtn.id.(['d_',labelName{3 + (3*i - 3)}]).Y = d_jointPower_Y;
211       rtn.id.(['d_',labelName{3 + (3*i - 3)}]).Z = d_jointPower_Z;
212
213       % 2nd time derivative
214       rtn.id.(['d2_',labelName{3 + (3*i - 3)}]).X = d2_jointPower_X;
215       rtn.id.(['d2_',labelName{3 + (3*i - 3)}]).Y = d2_jointPower_Y;
216       rtn.id.(['d2_',labelName{3 + (3*i - 3)}]).Z = d2_jointPower_Z;
217
218       if(debug)
219         figure;
220           subplot(311); hold all;
221             title(['Saggital Plane ', labelName{3 + (3*i - 3)}],'fontsize',20)
222             plot(jointPower_X,'k')
223           subplot(312); hold all;
224             plot(d_jointPower_X,'k')
225           subplot(313); hold all;
226             plot(d2_jointPower_X,'k')
227       end
228
229     end
230
231     % Marker Trajectories
232     % skip line
233     tline = fgetl(fid);
234
```

```
235     % skip line
236     tline = fgetl(fid);
237
238     % parse labels and create struct fields names
239     tline = fgetl(fid);
240     labels = textscan(tline, '%s', 'delimiter', ',');
241     [m, n] = size(labels{1});
242     labelCnt = 0;
243     for i=1:m
244       if ~(strcmp(labels{1}{i},''))
245         % pasre header on label
246         [token, remain] = strtok(labels{1}{i},':');
247         labelCnt = labelCnt + 1;
248         labelName{labelCnt} = remain(2:end);
249       end
250     end
251
252     % find number of subfields
253     tline = fgetl(fid);
254     sublabels = textscan(tline, '%s', 'delimiter', ',');
255     [m, n] = size(sublabels{1});
256
257     % skip line of units
258     fgetl(fid);
259
260     % put data in matrix
261     D = zeros(1,m);
262     formatSpec = [repmat(['%f'],1,m-1), '%f\n'];
263     rowCount = 1;
264     while(1)
265       C = textscan(fid, formatSpec, 'delimiter', ',');
266       dataRow = cell2mat(C);
267       if isempty(dataRow)
268         break;
269       end
270       D(rowCount,:) = cell2mat(C);
271       rowCount = rowCount + 1;
272     end
273
274     % put data (X,Y,Z subfields per field)
275     for i=1:labelCnt
276       rtn.markers.(labelName{i}).X = D(:,3 + (3*i - 3));
277       rtn.markers.(labelName{i}).Y = D(:,4 + (3*i - 3));
278       rtn.markers.(labelName{i}).Z = D(:,5 + (3*i - 3));
279     end
280
281     fclose(fid);
282   end
```

B.1.3 vicon_process_grf.m

```
1   function rtn = vicon_process_grf(trialName,varargin)
2
3     nVarArgs = length(varargin);
4
5     cutOff = 60;
6     eventThreshold = 50;
7
8     for i=1:2:nVarArgs
9       switch varargin{i}
10        case 'cutOff'
11          cutOff = varargin{i+1};
12        case 'eventThreshold'
13          eventThreshold = varargin{i+1};
14        otherwise
15          fprintf('\n%s option not found!\n',varargin{i});
16          return
17      end
18    end
19
20    % Parse model file
21    file = ['./',trialName,'_GRF.csv'];
22    if (exist(file,'file') ~= 2)
23      fprintf('\n\tGRF file not found: %s\n', file);
24      rtn = []
25      return
26    end
27
```

```
28    rtn.info.file = file;
29    rtn.params.processing_cutOff = cutOff;
30    rtn.params.eventThreshold = eventThreshold;
31
32    fid = fopen(file,'r');
33
34    % skip line
35    tline = fgetl(fid);
36    fs =fscanf(fid,'%f', 1);
37
38    rtn.params.fs = fs;
39
40    fclose(fid);
41
42    % Data
43    D = csvread(file,5,0);
44    frame_ = D(:,1);
45    subframe_ = D(:,2);
46    frame = (frame_ - 1).*10 + subframe_;
47
48    % Butterworth 4th order
49    [b,a] = butter(4,2*(cutOff/fs));
50
51    % Filter/Store data
52    rtn.time = frame.*(1/fs);
53    rtn.r.Fx = filtfilt(b,a,D(:,3));
54    rtn.r.Fy = filtfilt(b,a,D(:,4));
55    rtn.r.Fz = filtfilt(b,a,D(:,5));
56    rtn.l.Fx = filtfilt(b,a,D(:,6));
57    rtn.l.Fy = filtfilt(b,a,D(:,7));
58    rtn.l.Fz = filtfilt(b,a,D(:,8));
59
60    % Gait events
61    threshold = (rtn.r.Fz < eventThreshold);
62    dthreshold = threshold(2:end) - threshold(1:end-1);
63
64    to = find(dthreshold == 1);
65    hs = find(dthreshold == -1);
66
67    if(1)
68      figure; hold all;
69        plot(rtn.time,rtn.r.Fz);
70        plot(rtn.time(hs),rtn.r.Fz(hs),'ok')
71        plot(rtn.time(to),rtn.r.Fz(to),'or')
72    end
73
74    rtn.ge.r.hs_time(:,1) = rtn.time(hs);
75    rtn.ge.r.to_time(:,1) = rtn.time(to);
76
77    threshold = (rtn.l.Fz < eventThreshold);
78    dthreshold = threshold(2:end) - threshold(1:end-1);
79
80    to = find(dthreshold == 1);
81    hs = find(dthreshold == -1);
82
83    if(1)
84      figure; hold all;
85        plot(rtn.time,rtn.l.Fz);
86        plot(rtn.time(hs),rtn.l.Fz(hs),'ok')
87        plot(rtn.time(to),rtn.l.Fz(to),'or')
88    end
89
90    rtn.ge.l.hs_time(:,1) = rtn.time(hs);
91    rtn.ge.l.to_time(:,1) = rtn.time(to);
92
93  end
```

B.1.4 vicon_process_subject.m

```
1   function rtn = vicon_process_grf(trialName,varargin)
2
3     nVarArgs = length(varargin);
4
5     cutOff = 60;
6     eventThreshold = 50;
7
8     for i=1:2:nVarArgs
```

```
9       switch varargin{i}
10        case 'cutOff'
11          cutOff = varargin{i+1};
12        case 'eventThreshold'
13          eventThreshold = varargin{i+1};
14        otherwise
15          fprintf('\n%s option not found!\n',varargin{i});
16          return
17      end
18    end
19
20    % Parse model file
21    file = ['./',trialName,'_GRF.csv'];
22    if (exist(file,'file') ~= 2)
23      fprintf('\n\tGRF file not found: %s\n', file);
24      rtn = []
25      return
26    end
27
28    rtn.info.file = file;
29    rtn.params.processing_cutOff = cutOff;
30    rtn.params.eventThreshold = eventThreshold;
31
32    fid = fopen(file,'r');
33
34    % skip line
35    tline = fgetl(fid);
36    fs =fscanf(fid,'%f', 1);
37
38    rtn.params.fs = fs;
39
40    fclose(fid);
41
42    % Data
43    D = csvread(file,5,0);
44    frame_ = D(:,1);
45    subframe_ = D(:,2);
46    frame = (frame_ - 1).*10 + subframe_;
47
48    % Butterworth 4th order
49    [b,a] = butter(4,2*(cutOff/fs));
50
51    % Filter/Store data
52    rtn.time = frame.*(1/fs);
53    rtn.r.Fx = filtfilt(b,a,D(:,3));
54    rtn.r.Fy = filtfilt(b,a,D(:,4));
55    rtn.r.Fz = filtfilt(b,a,D(:,5));
56    rtn.l.Fx = filtfilt(b,a,D(:,6));
57    rtn.l.Fy = filtfilt(b,a,D(:,7));
58    rtn.l.Fz = filtfilt(b,a,D(:,8));
59
60    % Gait events
61    threshold = (rtn.r.Fz < eventThreshold);
62    dthreshold = threshold(2:end) - threshold(1:end-1);
63
64    to = find(dthreshold == 1);
65    hs = find(dthreshold == -1);
66
67    if(1)
68      figure; hold all;
69        plot(rtn.time,rtn.r.Fz);
70        plot(rtn.time(hs),rtn.r.Fz(hs),'ok')
71        plot(rtn.time(to),rtn.r.Fz(to),'or')
72    end
73
74    rtn.ge.r.hs_time(:,1) = rtn.time(hs);
75    rtn.ge.r.to_time(:,1) = rtn.time(to);
76
77    threshold = (rtn.l.Fz < eventThreshold);
78    dthreshold = threshold(2:end) - threshold(1:end-1);
79
80    to = find(dthreshold == 1);
81    hs = find(dthreshold == -1);
82
83    if(1)
84      figure; hold all;
85        plot(rtn.time,rtn.l.Fz);
86        plot(rtn.time(hs),rtn.l.Fz(hs),'ok')
87        plot(rtn.time(to),rtn.l.Fz(to),'or')
88    end
89
```

```
90    rtn.ge.l.hs_time(:,1) = rtn.time(hs);
91    rtn.ge.l.to_time(:,1) = rtn.time(to);
92
93  end
```

B.2 AB-ILC implementation Matlab code

This is a Matlab implementation of the AB-ILC algorithm detailed in Sect.

```
1   function S_k = ab_ilc(yd_k, y_k, S_km1, varargin)
2   % ab_ilc : Adaptive backstepping iterative learning control
3   %
4   %
5   %   Algorithm (applied to each harmonic):
6   %
7   %     Update Conditions:
8   %
9   %     c1 = |E_k| < |E_bar_k|
10  %
11  %     c2 = |E_k| > |E_bar_k| + epsilon
12  %
13  %     c3 = |Y_k - Y_0 | > alpha |E_0|
14  %
15  %     Cache:
16  %              / 1/gamma * rho_km1    if c2 or c3
17  %     rho_k =
18  %              \ rho_km1              else
19  %
20  %
21  %               /   E_k          if c1 and ~c3
22  %     E_bar_k =
23  %               \  E_bar_km1    else
24  %
25  %
26  %               /  U_k          if c1 and ~c3
27  %     U_bar_k =
28  %               \  U_bar_km1    else
29  %
30  %
31  %   U_kp1 = U_bar_k + rho_k * G_m^{-1} * E_bar_k
32  %
33  %   Note: rho_k always is in a range [0,1], use gain to set the learning gain.
34  %   epsilon is padding for error comparison. Use std of output signal druing
35  %   k=0 iteration.
36  %
37  %   Use:
38  %
39  %   First iteration:
40  %   S_0 = adaptiveILC(yd_k, y_k, 0, 'maxharmonic', maxharmonic, ...
41  %                                   'scalarmodel',Gm)
42  %
43  %   All others:
44  %   S_k = adaptiveILC(yd_k, y_k, S_km1)
45
46  %
47  % Example:
48  %   S_0 = adaptiveILC(yd_0, y_0, 0, 'init', maxharmonic);
49  %   S_1 = adaptiveILC(yd_1, y_1, S_0);
50  %   S_2 = adaptiveILC(yd_2, y_2, S_1);
51  %   .
52  %   .
53  %   .
54  %   S_k = adaptiveILC(yd_k, y_k, gain_k, S_km1);
55
56
57    % Defaults
58    init = 0;
59    alpha = 2;
60    gamma = 2;
61
62    % Init
63    if ~isstruct(S_km1)
64      init = 1;
65      nVarArgs = length(varargin);
```

```
66      for i=1:2:nVarArgs
67        switch varargin{i}
68          case 'maxharmonic'
69            maxharmonic = varargin{i+1};
70          case 'epsilon'
71            epsilon = varargin{i+1};
72          case 'gain'
73            gain = varargin{i+1};
74          case 'alpha'
75            alpha = varargin{i+1};
76          case 'gamma'
77            gamma = varargin{i+1};
78          otherwise
79            fprintf('\n%s option not found!\n',varargin{i});
80            return
81        end
82      end
83    end
84
85    % Number of points
86    N = numel(yd_k);
87
88    % This is really f/fs -> fs is the sampling freq.
89    % Typically use normalized data (e.g., %gait) hence fs = NNpoints/T_gait
90    f = (0:(N/2))/N;
91    f = f(:);
92
93    % FFTs
94    Yd_k = fft(yd_k);
95    Yd_k = Yd_k(1:(N/2+1));
96
97    Y_k = fft(y_k);
98    Y_k = Y_k(1:(N/2+1));
99
100   % Error
101   E_k = Yd_k - Y_k;
102
103   e_k_inf = norm(yd_k - y_k,inf);
104   e_k_2 = norm(yd_k - y_k,2);
105   E_k_inf = norm(Yd_k - Y_k,inf);
106   E_k_2 = norm(Yd_k - Y_k,2);
107
108   % Initialize S_0 if flag is set
109   if (init)
110
111     % Init rho
112     rho_k = ones(numel(f),1);
113     rho_k((maxharmonic+2):end) = 0;
114
115     U_kp1 = rho_k .* gain .* E_k;
116     u_kp1 = ifft([U_kp1; conj(flipud(U_kp1(2:end-1)))]);
117
118     % Struct
119     S_0.opt.gain = gain;
120     S_0.opt.gamma = gamma;
121     S_0.opt.epsilon = epsilon;
122     S_0.opt.alpha = alpha;
123     S_0.opt.maxharmonic = maxharmonic;
124
125     S_0.f = f;
126     S_0.Yd_k = Yd_k;
127     S_0.yd_k = yd_k;
128     S_0.Y_k = Y_k;
129     S_0.y_k = y_k;
130     S_0.rho_k = rho_k;
131
132     S_0.E_bar_k = E_k;
133     S_0.U_bar_k = zeros(numel(f),1);
134
135     S_0.E_0 = E_k;
136     S_0.Yd_0 = Yd_k;
137
138     S_0.E_k = E_k;
139     S_0.U_kp1 = U_kp1;
140     S_0.u_kp1 = u_kp1;
141
142     S_0.e_k_inf = e_k_inf;
143     S_0.e_k_2 = e_k_2;
144     S_0.E_k_inf = E_k_inf;
145     S_0.E_k_2 = E_k_2;
146
```

```
147       S_k = S_0;
148       return
149     end
150
151     % Unwrap input struct.
152     Yd_km1 = S_km1.Yd_k;
153     rho_km1 = S_km1.rho_k;
154     U_k = S_km1.U_kp1;
155     E_km1 = S_km1.E_k;
156
157     E_bar_km1 = S_km1.E_bar_k;
158     U_bar_km1 = S_km1.U_bar_k;
159
160     % Conditions
161     c1 = abs(E_k) < abs(E_bar_km1);
162     c2 = abs(E_k) > (abs(E_bar_km1) + S_km1.opt.epsilon);
163     c3 = abs(S_km1.Yd_0 - Yd_k) > S_km1.opt.alpha.*abs(S_km1.E_0);
164
165
166     % Updates
167     rho_k = (rho_km1/S_km1.opt.gamma).*(c2 | c3) ...
168             + (rho_km1).*(~(c2 | c3));
169
170     E_bar_k = (E_k).*(c1 & ~c3) ...
171               + (E_bar_km1).*(~(c1 & ~c3));
172
173     U_bar_k = (U_k).*(c1 & ~c3) ...
174               + (U_bar_km1).*(~(c1 & ~c3));
175
176     U_kp1 = U_bar_k + rho_k .* S_km1.opt.gain .* E_bar_k;
177     u_kp1 = ifft([U_kp1; conj(flipud(U_kp1(2:(end-1))))]);
178
179     % Copy previous structure
180     S_k = S_km1;
181
182     S_k.c1 = c1;
183     S_k.c2 = c2;
184     S_k.c3 = c3;
185
186     if(0)
187     mh = S_km1.opt.maxharmonic;
188     figure; hold all;
189       stem(f(1:mh+1),S_k.c1(1:mh+1))
190       stem(f(1:mh+1),S_k.c2(1:mh+1))
191       stem(f(1:mh+1),S_k.c3(1:mh+1))
192     end
193
194     % Update
195     S_k.Yd_k = Yd_k;
196     S_k.yd_k = yd_k;
197     S_k.Y_k = Y_k;
198     S_k.y_k = y_k;
199
200     S_k.E_bar_k = E_bar_k;
201     S_k.U_bar_k = U_bar_k;
202     S_k.rho_k = rho_k;
203
204     S_k.E_k = E_k;
205     S_k.U_kp1 = U_kp1;
206     S_k.u_kp1 = u_kp1;
207
208     S_k.e_k_inf = e_k_inf;
209     S_k.e_k_2 = e_k_2;
210     S_k.E_k_inf = E_k_inf;
211     S_k.E_k_2 = E_k_2;
212   end
```

B.3 ILC user interface implementation Matlab code

This is a text based user interface, for use during experiments, for the AB-ILC algorithm. It provides the experimentalist with details of the state of the learning algorithm.

```
1   function rtn = ilc_poweredAnkle_tui(varargin)
2   %%
```

```
% ilc_poweredAnkle_tui  Text based user interface for iterative learning
%                       control.
%
%   ilc_poweredAnkle_tui() default configuration.
%
%   ilc_poweredAnkle_tui(Name,Value, ...) Name, Value pairs for adjustable
%   settings.
%

  nVarArgs = length(varargin);


  % Defauts Settings
  % algo setting
  gain = 0.5;
  maxharmonic = 10;
  smooth_vector = [90,5];


  % Motor 2
  k_t = 0.00775;
  k_s = 1223;
  Vcc = 60;
  R_T = 1800;
  maxMotorAmp = 14;
  maxMotorRpm = 80000;
  file_suffix = '';
%  % Motor 1 (old settings from S5)
%  k_t = 0.0053;
%  k_s = 1730;
%  Vcc = 30;
%  maxMotorAmp = 20;
%  maxMotorRpm = 40000;
%  R_T = 472;
% fc_norm = (2*20)/1000;



  gaitCycle = 0:0.1:(100 - 0.1);
  N = numel(gaitCycle);

  % 10 \omega0
  fc_norm = (2*10)/N;


  for i=1:2:nVarArgs
    switch varargin{i}
      case 'gain'
        gain = varargin{i+1};
      case 'maxharmonic'
        maxharmonic = varargin{i+1}
      case 'fc_norm'
        fc_norm = varargin{i+1}
      case 'smooth_vector'
        smooth_vector = varargin{i+1};
      case 'k_t'
        % torque constant (Nm/A)
        k_tau = varargin{i+1};
      case 'k_s'
        % speed constant (rpm/V)
        k_s = varargin{i+1};
      case 'Vcc'
        % operating voltage (V)
        Vcc = varargin{i+1};
      case 'R_T'
        % total tranmission ratio
        R_T = varargin{i+1};
      case 'maxMotorAmp'
        % max motor current (A) -- setting on ESCON 70/10 servo controller
        maxMotorAmp = varargin{i+1};
      case 'maxMotorRpm'
        % max motor speed (Rpm) -- setting on ESCON 70/10 servo controller
        maxMotorRpm = varargin{i+1};
      otherwise
        fprintf('\n%s option not found!\n',varargin{i})
        rtn = [];
        return
    end
  end
```

```
83
84      %-------------------------------------------------------------------------
85      % Plots
86      % -------------------------------------------------------------------------
87      fig = figure;
88
89        % |E_k| v freq.
90        subplot(3,3,4); hold all;
91          h_E_bar_v_f_p_gamma = stem(0:maxharmonic, 0.*(0:maxharmonic),'--r');
92          h_E_bar_v_f = stem(0:maxharmonic, 0.*(0:maxharmonic),'k');
93          h_E_v_f = stem((0:maxharmonic)+0.2, 0.*(0:maxharmonic),'m');
94          ylabel('$| E |$','fontsize',17,'interpreter','latex');
95          xlabel('Harmonic','fontsize',17,'interpreter','latex');
96          grid on; box on;
97          xlim([-0.3,maxharmonic+0.5])
98          l = legend([h_E_bar_v_f,h_E_bar_v_f_p_gamma,h_E_v_f], ...
99                     {'$E^*_k$','$E^*_k + \gamma$', '$E_k$'});
100         set(l,'interpreter','latex', ...
101               'Position', [0.0446 0.5507 0.0571  0.0726]);
102
103
104       % |U_k| v freq.
105       subplot(3,3,7), hold all;
106         h_U_bar_v_f = stem(0:maxharmonic, 0.*(0:maxharmonic),'k');
107         h_U_v_f = stem((0:maxharmonic)+0.2, 0.*(0:maxharmonic),'m');
108         h_U_kp1_v_f = stem((0:maxharmonic)-0.2, 0.*(0:maxharmonic),'b');
109         ylabel('$| U |$','fontsize',17,'interpreter','latex');
110         xlabel('Harmonic','fontsize',17,'interpreter','latex');
111         grid on; box on;
112         xlim([-0.3,maxharmonic+0.5])
113         l = legend([h_U_bar_v_f,h_U_v_f,h_U_kp1_v_f], ...
114                    {'$U^*_k$','$U_k$','$U_{k+1}$'});
115         set(l,'interpreter','latex', ...
116               'Position',[0.0581    0.2779    0.0400    0.0500]);
117
118
119
120       % rho v freq
121       subplot(3,3,1); hold all;
122         h_rho_v_f = stem(0:maxharmonic, 0.*(0:maxharmonic),'k');
123         ylabel('$\rho$','fontsize',17,'interpreter','latex');
124         xlabel('Harmonic','fontsize',17,'interpreter','latex');
125         grid on; box on;
126         xlim([-0.3,maxharmonic+0.5])
127
128       % y v t
129       subplot(3,3,2:3); hold all;
130         h_title = title(['Iteration = ', num2str(0)], ...
131                         'interpreter','latex','fontsize',17);
132         h_y_v_t = plot(gaitCycle,0.*gaitCycle,'k');
133         h_yd_v_t = plot(gaitCycle,0.*gaitCycle,'r');
134         h_u_k_v_t = plot(gaitCycle,0.*gaitCycle,'g');
135         h_u_kp1_v_t = plot(gaitCycle,0.*gaitCycle,'c');
136         h_tau_ak_v_t = plot(gaitCycle,0.*gaitCycle,'m');
137         h_stance_1_v_t = plot([0 0],[0,0],'--k');
138         h_pre_1_v_t = plot([0 0],[0,0],'--k');
139         ylabel('(Nm)','fontsize',17,'interpreter','latex');
140         grid on; box on
141         l = legend([h_y_v_t,h_yd_v_t,h_u_k_v_t,h_u_kp1_v_t,h_tau_ak_v_t], ...
142             {'$\tau_{p,k}$','$\tau_{b,k}$', ...
143              '$R_T k_\tau \cdot u_k$', ...
144              '$R_T k_\tau \cdot u_{k+1}$', ...
145              '$\tau_{a,k} + \tau_{bias}$'});
146         set(l,'interpreter','latex', ...
147               'Position', [0.9149    0.8275    0.0648    0.0953]);
148
149
150       % u v t
151       subplot(3,3,5:6); hold all;
152         title('$U_{k+1} = U^*_k + \rho E^*_k$','interpreter','latex')
153         h_u_v_t = plot(gaitCycle,0.*gaitCycle,'g');
154         h_u_filt_v_t = plot(gaitCycle,0.*gaitCycle,'m');
155         h_stance_2_v_t = plot([0 0],[0,0],'--k');
156         h_pre_2_v_t = plot([0 0],[0,0],'--k');
157         xlabel('\% gait','fontsize',17,'interpreter','latex');
158         ylabel('(A)','fontsize',17,'interpreter','latex');
159         grid on; box on;
```

```
160
161         l = legend([h_u_v_t,h_u_filt_v_t], ...
162             {'$\hat{u}_{k+1}$','$u_{k+1}$'});
163         set(l,'interpreter','latex', ...
164               'Position',[0.9140    0.5692    0.0468    0.0500]);
165
166
167       % e v t | f
168       subplot(3,3,8:9); hold all
169          h_er_inf_t = plot(0,0,'k');
170          h_er_2_t = plot(0,0,'r');
171          h_er_inf_f = plot(0,0,'--b');
172          h_er_2_f = plot(0,0,'--g');
173          h_er_inf_t_min = plot(0,0,'ok');
174          h_er_2_t_min = plot(0,0,'or');
175          h_er_inf_f_min = plot(0,0,'ob');
176          h_er_2_f_min = plot(0,0,'og');
177          xlabel('Iteration','fontsize',17,'interpreter','latex');
178          ylabel('Error','fontsize',17,'interpreter','latex')
179          grid on; box on;
180          l = legend('$\|e\|_\infty$','$\|e\|_2$','$\|E\|_\infty$','$\|E\|_2$');
181          set(l,'interpreter','latex','box','off');
182
183       fig.Position = [3 61 1426 738];
184       fig.PaperPosition = [-5.3542,1.3542,19.2083,8.2917];
185
186     % Filter for uff smoothing
187     [b,a] = butter(1,fc_norm);
188
189     settings.maxharmonic = maxharmonic;
190     settings.gain = gain;
191     settings.fc_norm = fc_norm;
192
193     rtn.settings = settings;
194
195     fprintf('\n----------------------\n')
196     fprintf('Iterative Learning UI\n')
197     fprintf('----------------------\n')
198     fprintf('\nSettings:\n')
199
200     fprintf('\tMaximum harmonic = %i\n',maxharmonic)
201     fprintf('\tLearning Gain = %.3f\n',gain)
202     fprintf('\tNormalized Cutoff Freq = %.3f\n',fc_norm)
203
204     % Error Vectors
205     Einf_t = [];
206     E2_t = [];
207     Einf_f = [];
208     E2_f = [];
209
210     k=1;
211     while(1)
212       fprintf('\n.................\n')
213       fprintf('Iteration %i\n',k - 1);
214       fprintf('.................\n')
215
216       % Get k trial
217       while(1)
218         fprintf('\n\t');
219         trial = input('Enter trial name: ','s');
220         fprintf(['\n\t',trial]);
221         usr_input = input(' - Is this correct? [y/n] ','s');
222         if strcmp('y',usr_input)
223           break;
224         end
225       end
226
227       % Data Analysis
228       fprintf('\n\tParsing data...\n');
229       rtn.T{k} = vicon_process_trial(trial, ...
230                                      'k_t',k_t, ...
231                                      'k_s',k_s, ...
232                                      'Vcc',Vcc, ...
233                                      'R_T',R_T, ...
234                                      'maxMotorAmp',maxMotorAmp, ...
235                                      'maxMotorRpm',maxMotorRpm);
236
237       % Get info on k=0 iteration
```

```
238       if (k==1)
239         prostheticSide = rtn.T{k}.subject.amp_side;
240         mass = rtn.T{k}.subject.mass_kg;
241         rtn.settings.prostheticSide = prostheticSide;
242         rtn.settings.mass = mass;
243         mass = 85;
244
245         fprintf('\tProsthetic Side = %s\n',prostheticSide);
246         fprintf('\tMass = %i\n',mass)
247       end
248
249       time = rtn.T{k}.time;
250       grf.time = rtn.T{k}.grf.time;
251       emb.time = rtn.T{k}.emb.time;
252
253       % Desired output and output
254       if strcmp(prostheticSide,'left')
255
256         % Means
257         yd_k = rtn.T{k}.gait.model.r.RAnkleMoment.X(:,2);
258         y_k = rtn.T{k}.gait.model.l.LAnkleMoment.X(:,2);
259
260         % all normalized trajectories
261         yd_k_all = rtn.T{k}.gait.model.normalized.r.RAnkleMoment.X;
262         y_k_all = rtn.T{k}.gait.model.normalized.l.LAnkleMoment.X;
263
264         % Timing
265         Tp = rtn.T{k}.segment.grf.l.gait.time(:,2) ...
266             - rtn.T{k}.segment.grf.l.gait.time(:,1);
267
268         stance_percent = mean((rtn.T{k}.segment.grf.l.stance.time(:,2) ...
269                         - rtn.T{k}.segment.grf.l.stance.time(:,1))./Tp).*100;
270
271         % HS events
272         grf.hs = rtn.T{k}.ge.grf.l.hs_time;
273         emb.hs = rtn.T{k}.ge.emb.l.hs_time;
274
275       else
276
277         % mean torques
278         yd_k = rtn.T{k}.gait.model.l.LAnkleMoment.X(:,2);
279         y_k = rtn.T{k}.gait.model.r.RAnkleMoment.X(:,2);
280
281         % all normalized trajectories
282         yd_k_all = rtn.T{k}.gait.model.normalized.l.LAnkleMoment.X;
283         y_k_all = rtn.T{k}.gait.model.normalized.r.RAnkleMoment.X;
284
285
286         % Timing
287         Tp = rtn.T{k}.segment.grf.r.gait.time(:,2) ...
288             - rtn.T{k}.segment.grf.r.gait.time(:,1);
289
290         stance_percent = mean((rtn.T{k}.segment.grf.r.stance.time(:,2) ...
291                         - rtn.T{k}.segment.grf.r.stance.time(:,1))./Tp).*100;
292
293         % HS events
294         grf.hs = rtn.T{k}.ge.grf.r.hs_time;
295         emb.hs = rtn.T{k}.ge.emb.r.hs_time;
296
297       end
298
299       % Remove last point (0% == 100%) we want N even for FFT
300       yd_k = yd_k(1:(end-1));
301       y_k = y_k(1:(end-1));
302
303       while(1)
304
305         % If first iteration, initialize learning
306         if (k==1)
307           fprintf('\n\tInitializing learning structures...\n');
308
309           % First iteration, find y_0 std in freq. domain
310           L = numel(y_k_all(1:end-1,1));
311           f = (0:(L/2));
312           Y_k_all = fft(y_k_all(1:end-1,:));
313           Y_k_all = Y_k_all(1:(L/2)+1,:);
314           Y_k_abs_mean = mean(abs(Y_k_all)')';
315           Y_k_abs_std = std(abs(Y_k_all)')';
316
```

```
317           epsilon = 3.*Y_k_abs_std;
318
319           figure;
320             title('Standard deviation of output','fontsize',20);
321             plot(f,epsilon,'ok');
322             xlabel('Harmonic','fontsize',20);
323             ylabel('STD','fontsize',20)
324             xlim([0,maxharmonic])
325
326             pause
327
328           rtn.S{k} = ab_ilc(yd_k,y_k,0, ...
329                             'maxharmonic', maxharmonic, ...
330                             'gain',gain, ...
331                             'alpha',2, ...
332                             'gamma',2, ...
333                             'epsilon',epsilon);
334
335         else
336           fprintf('\n\tLearning...\n');
337           rtn.S{k} = ab_ilc(yd_k,y_k,rtn.S{k-1});
338
339
340
341         end
342
343         if(0)
344           figure; hold all;
345             plot(time,y);
346             for i=1:numel(grf.hs)
347               plot([grf.hs(i) grf.hs(i)], [min(y) max(y)],'-r')
348             end
349             for i=1:numel(emb.hs)
350               plot([emb.hs(i) emb.hs(i)], [min(y) max(y)],'--r')
351             end
352         end
353
354         % Find HS delay
355         if (numel(grf.hs) ~= numel(emb.hs))
356           fprintf('\t\t\tHS mismatch, cannot calculate delay');
357           est_delay = nan;
358         else
359           est_delay = mean(emb.hs - grf.hs);
360         end
361
362         smooth_vector(1) = (stance_percent + 20);
363         Tp_mean = mean(Tp);
364
365         fprintf('\n\t\tTiming Info\n')
366         fprintf('\t\t-------------\n')
367         fprintf('\t\tEst. HS delay = %.3f (s), %.3f%%\n', ...
368           est_delay, (est_delay/Tp_mean).*100)
369         fprintf('\t\tTp = %.3f (s)\n',Tp_mean)
370         fprintf('\t\tInital Smoothing = %.2f%%\n',smooth_vector(2))
371         fprintf('\t\tStance = %.2f%%\n',smooth_vector(1))
372
373         % Store Timing Info
374         info.Tp_mean = Tp_mean;
375         info.stance_percent = stance_percent;
376         info.smooth_vector = smooth_vector;
377
378         % Store the errors
379         Einf_t(k) = rtn.S{k}.e_k_inf;
380         E2_t(k) = rtn.S{k}.e_k_2;
381         Einf_f(k) = rtn.S{k}.E_k_inf;
382         E2_f(k) = rtn.S{k}.E_k_2;
383
384         [~,iinf_t] = min(Einf_t(1:k)./Einf_t(1));
385         [~,i2_t] = min(E2_t(1:k)./E2_t(1));
386         [~,iinf_f] = min(Einf_f(1:k)./Einf_f(1));
387         [~,i2_f] = min(E2_f(1:k)./E2_f(1));
388
389         preFiltU_kp1 = rtn.S{k}.U_kp1;
390
391         % Filter uff  [Nm/kg]
392         u_kp1_filt = rtn.S{k}.u_kp1;
393         u_kp1_filt(1:(round(smooth_vector(2)*10)+1)) = 0;
394         u_kp1_filt((round(smooth_vector(1)*10)+1):end) = 0;
395         u_kp1_filt = filtfilt(b,a,u_kp1_filt);
396
```

```
397         % Store in frequency domain
398         temp = fft(u_kp1_filt);
399         rtn.S{k}.U_kp1 = temp(1:(1000/2+1));
400
401         if(0)
402         figure; hold all;
403           stem(abs(rtn.S{k}.U_kp1), '-k')
404           stem(abs(preFiltU_kp1), '--r')
405         end
406
407         % Convert to [A]
408         u_kp1_A = mass*rtn.S{k}.u_kp1*(1/(R_T.*k_t));
409         u_kp1_A_filt = mass*u_kp1_filt*(1/(R_T.*k_t));
410
411         % Convert to [Nm]
412         u_kp1_Nm_filt = u_kp1_A_filt ./ (1/(R_T.*k_t));
413         u_kp1_Nm = u_kp1_A ./ (1/(R_T.*k_t));
414
415         rtn.S{k}.u_kp1_Nm_filt = u_kp1_Nm_filt;
416         rtn.S{k}.u_kp1_Nm = u_kp1_Nm;
417
418         rtn.S{k}.u_kp1_A_filt = u_kp1_A_filt;
419         rtn.S{k}.u_kp1_A = u_kp1_A;
420
421         % update plots
422         set(h_title,'string',['Iteration = ', num2str(k-1)]);
423         set(h_u_v_t,'YData',u_kp1_A);
424         set(h_u_filt_v_t,'YData',u_kp1_A_filt);
425         set(h_stance_2_v_t,'XData',[smooth_vector(1),smooth_vector(1)], ...
426                            'YData',[min(u_kp1_A_filt),max(u_kp1_A_filt)]);
427         set(h_pre_2_v_t,'XData',[smooth_vector(2),smooth_vector(2)], ...
428                            'YData',[min(u_kp1_A_filt),max(u_kp1_A_filt)]);
429
430         set(h_tau_ak_v_t,'YData',rtn.T{k}.gait.emb.l.tau_a(1:end-1,2))
431
432         if k > 1
433           set(h_u_k_v_t, 'YData',rtn.S{k-1}.u_kp1_Nm_filt);
434           set(h_U_v_f,'YData',abs(rtn.S{k-1}.U_kp1(1:(maxharmonic+1))));
435         end
436
437         set(h_U_kp1_v_f,'YData',abs(rtn.S{k}.U_kp1(1:(maxharmonic+1))));
438
439         set(h_U_bar_v_f,'YData',abs(rtn.S{k}.U_bar_k(1:(maxharmonic+1))));
440         set(h_E_bar_v_f,'YData',abs(rtn.S{k}.E_bar_k(1:(maxharmonic+1))));
441         set(h_E_bar_v_f_p_gamma,'YData', ...
442           abs(rtn.S{k}.E_bar_k(1:(maxharmonic+1))) ...
443             + rtn.S{k}.opt.epsilon(1:(maxharmonic+1)))
444         set(h_y_v_t,'YData', rtn.S{k}.y_k.*mass);
445         set(h_yd_v_t,'YData',rtn.S{k}.yd_k.*mass);
446
447         set(h_u_kp1_v_t,'YData', rtn.S{k}.u_kp1_Nm_filt);
448         set(h_stance_1_v_t,'XData',[smooth_vector(1),smooth_vector(1)], ...
449                            'YData',[min(rtn.S{k}.yd_k.*mass),max(rtn.S{k}.yd_k.*mass)]);
450         set(h_pre_1_v_t,'XData',[smooth_vector(2),smooth_vector(2)], ...
451                            'YData',[min(rtn.S{k}.yd_k.*mass),max(rtn.S{k}.yd_k.*mass)]);
452
453         set(h_er_inf_t_min,'XData',iinf_t-1,'YData',Einf_t(iinf_t)./Einf_t(1));
454         set(h_er_2_t_min,'XData',i2_t-1,'YData',E2_t(i2_t)./E2_t(1));
455         set(h_er_inf_f_min,'XData',iinf_f-1,'YData',Einf_f(iinf_f)./Einf_f(1));
456         set(h_er_2_f_min,'XData',i2_f-1,'YData',E2_f(i2_f)./E2_f(1));
457
458         set(h_er_inf_t,'XData',(1:k)-1,'YData',Einf_t(1:k)./Einf_t(1));
459         set(h_er_2_t,'XData',(1:k)-1,'YData',E2_t(1:k)./E2_t(1));
460         set(h_er_inf_f,'XData',(1:k)-1,'YData',Einf_f(1:k)./Einf_f(1));
461         set(h_er_2_f,'XData',(1:k)-1,'YData',E2_f(1:k)./E2_f(1));
462
463         set(h_rho_v_f,'YData',abs(rtn.S{k}.rho_k(1:(maxharmonic+1)).*gain));
464         set(h_E_v_f,'YData',abs(rtn.S{k}.E_k(1:(maxharmonic+1))));
465
466         % Promt for signal sufficient
467         fprintf('\n\tSignal learned!\n');
468         fprintf('\t\tIs this signal sufficient?')
469         usr_input = input('[y|n] ', 's');
470         if strcmp(usr_input,'n')
471           while(1)
472             fprintf('\t\tDo you want to change a parameter?\n')
473             fprintf('\t\t 1 - gain\n')
474             fprintf('\t\t 2 - earlysmooth\n')
```

```
475             fprintf('\t\t 3 - latesmooth\n');
476             fprintf('\t\t 4 - done\n')
477             while(1)
478               fprintf('\t\t\t');
479               usr_input = input('Enter response: ');
480               if isscalar(usr_input)
481                 break;
482               end
483             end
484
485             switch usr_input
486               case 1
487                 fprintf('\t\t\t');
488                 usr_input = input('Enter new gain value: ');
489                 gain = usr_input;
490               case 2
491                 fprintf('\t\t\t');
492                 usr_input = input('Enter new earlysmooth: ');
493                 smooth_vector(2) = usr_input;
494               case 3
495                 fprintf('\t\t\t');
496                 usr_input = input('Enter new latesmooth: ');
497                 smooth_vector(1) = usr_input;
498               case 4
499                 fprintf('\tLearning Gain = %.3f\n',gain)
500                 fprintf('\tSmoothing = %i%% - %i%%\n', smooth_vector)
501                 break;
502               otherwise
503                 warning('Not an option');
504             end
505           end
506         else
507           break;
508         end
509       end
510
511       rtn.S{k}.info = info;
512
513       % End of Iteration Promt
514       fprintf('\n\t');
515       fprintf('Write feedforward signal to file?');
516       usr_input = input('[y|n]','s');
517       fprintf('\n');
518
519       if strcmp(usr_input,'y')
520         writeFlag = 1;
521         file = ['./','uff_',file_suffix,num2str(k)];
522         if exist(file,'file') == 2
523           fprintf('\n\t\tFile already exist');
524           usr_input = input(' overwrite? [y/n]','s');
525           if strcmp('y',usr_input)
526             writeFlag = 1;
527           else
528             writeFlag = 0;
529             fprintf('\n\tSignal not written to file.\n');
530           end
531         end
532         if writeFlag
533           fid = fopen(file,'w');
534           fprintf(fid,'%f\n',rtn.S{k}.u_kp1_A_filt);
535           fclose(fid);
536         end
537       end
538
539       fprintf('\n\t');
540       usr_input = input('Continue with next k? [y/n]','s');
541       if strcmp('n',usr_input)
542         fprintf('\t\t')
543         usr_input = input('Redo k? [y/n] (n will exit ui)','s');
544         if strcmp('y',usr_input)
545           continue;
546         end
547         break;
548       end
549
550       % Increment
551       k = k + 1;
552     end
553   end
```

Appendix C

OPENWEARABLE C IMPLEMENTATION

This appendix contains some of the C implementations of openWearble – an open-source software/hardware project initiated during my thesis work. Included below are the entry point source code for all three processors: main.c (CPU), pru0_main.c (PRU1), and pru1_main.c (PRU2). Additionally, the text based user interface (tui.c), the heel-strike detection implementation (gaitPhase.c), and the structure declarations (mem_types.h) are included. Lastly, a sample subject file is also included. All files can be accessed at:

C.1 CPU entry point: main.c

```
/* Copyright 2017 Jonathan Realmuto

 Licensed under the Apache License, Version 2.0 (the "License");
 you may not use this file except in compliance with the License.
 You may obtain a copy of the License at

     http://www.apache.org/licenses/LICENSE-2.0

 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.
============================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <signal.h>
#include <unistd.h>
#include <time.h>
#include <string.h>

#include <prussdrv.h>
#include <pruss_intc_mapping.h>

#include "mem_types.h"
#include "pru_wrappers.h"
#include "gpio.h"
#include "common.h"
#include "tui.h"

// Global -----------------------------------------------------------------
int debug;
volatile int doneFlag = 0;

// Protoypes --------------------------------------------------------------
void sigintHandler(int sig);
int loadPruSoftware(void);
void startPruLoop(void);


```

```
44  // ---------------------------------------------------------------------------
45  int main(int argc, char **argv)
46  {
47    // Command Line Inputs
48    if(argc != 1){
49      if(strcmp(argv[1], "-v") == 0)
50        debug = 1;
51    }
52    else{
53      debug = 0;
54    }
55
56    // Welcome
57    if(!(debug)){
58      printf("\n----------------------\n");
59      printf("Welcome to openWearable v0.1\n");
60      printf("----------------------\n");
61    }
62    else{
63      printf("\n----------------------\n");
64      printf(" DEBUG MODE \n");
65      printf("----------------------\n");
66      signal(SIGINT, sigintHandler);
67    }
68
69    // Initialize the library, PRUs and interrupts.
70    if(pru_init() != 0)
71      return -1;
72
73    // Initialize debug buffer.
74    initDebugBuffer();
75
76    // Load parameters from file to memory.
77    if(loadParameters("config/PA_AO2") != 0){
78      printf("\nParameter file not found!\n");
79    }
80    printParameters(stdout);
81
82    // Load iir filter coefficients from file to memory.
83    if(loadIirFilterCoeff("config/lowpass_1_6Hz")){
84        printf("\nFilter coefficient file not found!\n");
85    }
86    printFirCoeff(stdout);
87
88    // Load lookup table from file to memory.
89    if(loadLookUpTable("config/sine_13amps_1Hz") != 0){
90      printf("\nLookup table file not found!\n");
91    }
92  //  printFFLookUpTable(stdout);
93
94    // Load binaries to PRUs IRAM.
95    if(loadPruSoftware() != 0)
96      return -1;
97
98    clearFlowBitField();
99
100   // Enable PRUs
101   enablePru(1);
102
103   // Start control loop
104   printf("\n\nPress enter to start\n\n");
105   getchar();
106
107   startPruLoop();
108
109   // Debug Loop
110   if(debug){
111     circBuffInit();
112     while(!(doneFlag)){
113       logData();
114     }
115     enablePru(0);
116     printDebugBuffer();
117
118     // Cleanup
119     pru_cleanup();
120     printf("pru0 and pru1 cleaned up.\n");
121   }
```

```
122
123     // TUI
124     else {
125       if(init_tui() != 0){
126         printf("TUI init fail.");
127         return -1;
128       }
129
130       // UI loop
131       if(start_tui() == 1){
132           enablePru(0);
133           printDebugBuffer();
134
135           // Cleanup
136           pru_cleanup();
137           printf("pru0 and pru1 cleaned up.\n");
138           raise(SIGINT);
139       }
140     }
141     return 0;
142   }
143
144   /* Methods ------------------------------------------------------------- */
145   void sigintHandler(int sig)
146   {
147     signal(sig, SIG_IGN);
148     doneFlag = 1;
149   }
150
151   int loadPruSoftware(void)
152   {
153     int rtn = 0;
154
155     // Run PRU0 software
156     if( (rtn = pru_run(PRU0, "./bin/pru0_main_text.bin")) != 0){
157       printf("pru_run() failed (PRU0)");
158       return -1;
159     }
160
161     // Run PRU1 software
162     if( (rtn = pru_run(PRU1, "./bin/pru1_main_text.bin")) != 0){
163       printf("pru_run() failed (PRU1)");
164       return -1;
165     }
166     return rtn;
167   }
168
169   void startPruLoop(void)
170   {
171     armToPru1Interrupt();
172     armToPru0Interrupt();
173   }
```

C.2 User interface: tui.c

```
1   /* Copyright 2017 Jonathan Realmuto
2
3    Licensed under the Apache License, Version 2.0 (the "License");
4    you may not use this file except in compliance with the License.
5    You may obtain a copy of the License at
6
7        http://www.apache.org/licenses/LICENSE-2.0
8
9    Unless required by applicable law or agreed to in writing, software
10   distributed under the License is distributed on an "AS IS" BASIS,
11   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12   See the License for the specific language governing permissions and
13   limitations under the License.
14  ============================================================================*/
15
16  #include <stdlib.h>
17  #include <stdio.h>
18  #include <stdint.h>
19  #include <unistd.h>
20  #include <math.h>
21  #include <termios.h>
22  #include <signal.h>
23  #include <fcntl.h>
```

```
24  #include <string.h>
25
26  #include "gpio.h"
27  #include "common.h"
28  #include "pru_wrappers.h"
29  #include "tui.h"
30
31  #define DEBUG_PIN "P8_15"
32
33  tui_t *ptui;
34
35  uint32_t gpio_debug;
36
37  /* ---------------------------------------------------------------------------
38   * Function init_tui()
39   * ------------------------------------------------------------------------- */
40  int init_tui(void)
41  {
42    struct sigaction action_ui;
43
44    ptui = malloc(sizeof(tui_t));
45    ptui->io_ready = 0;
46
47    /* Set up action for SIGIO */
48    action_ui.sa_handler = io_cb;
49    sigemptyset(&action_ui.sa_mask);
50    action_ui.sa_flags = 0;
51    if(sigaction(SIGIO, &action_ui, NULL) == -1)
52      printf("Error sigaction\n");
53
54    /* Set up file descriptor */
55    if(fcntl(0, F_SETOWN, getpid()) == -1){
56      printf("F_SETOWN error.\n");
57      return -1;
58    }
59    if(fcntl(0, F_SETFL, fcntl(0, F_GETFL) | O_ASYNC | O_NONBLOCK) == -1){
60      printf("Error setting stdin fd flags.\n");
61      return -1;
62    }
63    printf("TUI initialized.\n");
64
65    /* Debug pin */
66    /* set enable pins */
67    if(get_gpio_number(DEBUG_PIN, &gpio_debug) != 0){
68      printf("Error getting gpio number.\n");
69      return -1;
70    }
71    gpio_export(gpio_debug);
72    gpio_set_direction(gpio_debug, OUTPUT);
73
74    return 0;
75  }
76
77  /* ---------------------------------------------------------------------------
78   * Function io_cb()
79   * ------------------------------------------------------------------------- */
80  void io_cb(int sig)
81  {
82    ptui->io_ready = 1;
83  }
84
85  /* ---------------------------------------------------------------------------
86   * Function ui_menu_cb()
87   * ------------------------------------------------------------------------- */
88  void tui_menu(void)
89  {
90    char side;
91
92    if (getProsSide() == 1)
93      side = 'L';
94    else
95      side = 'R';
96
97    printf("\n\n-------------------------------------------------------------------------\n"
98           "hs_delay = %i, u_bias = %3.2f, ProsSide = %c, FF enabled = %d, FFgain = %3.2f\n"
99           "Menu: a - Enter new hs_delay\n"
100          "      s - Enter new u_bias\n"
101          "      d - Change prosthetic limb side\n"
102          "      f - Collect trial\n"
103          "      g - Save parameters\n"
104          "      h - Load parameters\n"
```

```
105             "      j - Load Feedforward lookup table\n"
106             "      k - Toggle feedforward control\n"
107             "      l - Enter new FFgain\n"
108             "      p - Test Feedforward\n"
109             "      q - Step Response\n"
110             "      e - exit\n"
111             "-------------------------------------------------------------------------\n",
112             geths_delay(), getu_bias(), side, getFFenable(), getFFgain());
113     fflush(stdout);
114   }
115
116   /* ------------------------------------------------------------------------
117    * Function tui()
118    * ------------------------------------------------------------------------ */
119   int start_tui(void)
120   {
121     char inChar;
122     float inFloat;
123     char inString[256];
124
125     char logFile[256] = "datalog/";
126     char configFile[256] = "config/";
127
128     tui_menu();
129     fflush(stdout);
130     while(1){
131
132       // Clear inputs
133       inChar = ' ';
134       inFloat = 0.0;
135       inString[0] = '\0';
136
137       // Wait for user input.
138       if(ptui->io_ready){
139         scanf(" %c", &inChar);
140
141         switch(inChar){
142
143           // ---- Exit -------------------------------------------------------
144           case 'e' :
145             return 1;
146
147           // ---- Set hs_delay -----------------------------------------------
148           case 'a' :
149             printf("\t\tEnter new hs_delay: ");
150             fflush(stdout);
151             ptui->io_ready = 0;
152
153             // Wait for user input.
154             while(1)
155               if(ptui->io_ready)
156                 break;
157
158             scanf(" %f", &inFloat);
159             seths_delay((uint32_t)inFloat);
160             tui_menu();
161             ptui->io_ready = 0;
162             break;
163
164           // ---- Set u_bias -------------------------------------------------
165           case 's' :
166             printf("\t\tEnter new u_bias: ");
167             fflush(stdout);
168             ptui->io_ready = 0;
169
170             // Wait for user input.
171             while(1)
172               if(ptui->io_ready)
173                 break;
174
175             scanf(" %f", &inFloat);
176             setu_bias(inFloat);
177             tui_menu();
178             ptui->io_ready = 0;
179             break;
180
181           // ---- Set ProSide-------------------------------------------------
182           case 'd' :
183             printf("\t\tChange ProsSide [y/n]? ");
184             fflush(stdout);
185             ptui->io_ready = 0;
```

```
186
187             // Wait for user input.
188             while(1)
189               if(ptui->io_ready)
190                 break;
191
192             scanf(" %c", &inChar);
193             if (inChar == 'y')
194               setProsSide( !getProsSide());
195             tui_menu();
196             ptui->io_ready = 0;
197             break;
198
199           // ---- Collect trial ------------------------------------------------
200           case 'f' :
201             printf("\t\tEnter trial name: ");
202             fflush(stdout);
203             ptui->io_ready = 0;
204
205             // Wait for input.
206             while(1)
207               if(ptui->io_ready)
208                 break;
209
210             scanf(" %s", inString);
211             strcat(logFile, inString);
212             printf("\t\tSaving data to %s\n",logFile);
213
214             // Init file and circbuff
215             logFileInit(logFile);
216             circBuffInit();
217
218             // Wait for enter to start saving data
219             printf("\t\tPress enter to start collection...\n");
220             fflush(stdout);
221             ptui->io_ready = 0;
222             while(1)
223               if(ptui->io_ready)
224                 break;
225             scanf(" %c", &inChar);
226
227             // Wait for enter to stop collection
228             printf("\t\tPress enter to stop collection...\n");
229             fflush(stdout);
230             ptui->io_ready = 0;
231
232             // Data collection loop
233             while(1){
234
235               logData();
236
237               // Check for input
238               if(ptui->io_ready)
239                 break;
240             }
241             scanf(" %c", &inChar);
242
243             closeLogFile();
244             logFile[0] = '\0';
245             strcat(logFile, "datalog/");
246             tui_menu();
247             fflush(stdout);
248             ptui->io_ready = 0;
249             break;
250
251           // ---- Save Parameters ----------------------------------------------
252           case 'g' :
253             printf("\t\tEnter parameter file name: ");
254             fflush(stdout);
255             ptui->io_ready = 0;
256
257             // Wait for user input
258             while(1)
259               if(ptui->io_ready)
260                 break;
261
262             scanf(" %s", inString);
263
264             /* Echo Trial Name */
265             strcat(configFile, inString);
```

```
266             printf("\t\tSaving parameters to %s\n",configFile);
267             fflush(stdout);
268             saveParameters(configFile);
269             configFile[0] = '\0';
270             strcat(configFile, "config/");
271             tui_menu();
272             ptui->io_ready = 0;
273             break;
274
275           // ---- Load Parameters -------------------------------------------------
276           case 'h' :
277             printf("\t\tEnter parameter file name: ");
278             fflush(stdout);
279             ptui->io_ready = 0;
280
281             // Wait for user input
282             while(1)
283               if(ptui->io_ready)
284                 break;
285
286             scanf(" %s", inString);
287
288             /* Echo Trial Name */
289             strcat(configFile, inString);
290             printf("\t\tLoading parameters from %s\n",configFile);
291             fflush(stdout);
292             loadParameters(configFile);
293             configFile[0] = '\0';
294             strcat(configFile, "config/");
295             tui_menu();
296             ptui->io_ready = 0;
297             break;
298
299           // ---- Load Lookup -----------------------------------------------------
300           case 'j' :
301             printf("\t\tEnter lookup table file name: ");
302             fflush(stdout);
303             ptui->io_ready = 0;
304
305             // Wait for user input
306             while(1)
307               if(ptui->io_ready)
308                 break;
309
310             scanf(" %s", inString);
311
312             /* Echo Trial Name */
313             strcat(configFile, inString);
314             printf("\t\tLoading lookup table from %s\n",configFile);
315             fflush(stdout);
316             loadLookUpTable(configFile);
317             configFile[0] = '\0';
318             strcat(configFile, "config/");
319             tui_menu();
320             ptui->io_ready = 0;
321             break;
322
323
324           // ---- Toggle feedforward ----------------------------------------------
325           case 'k' :
326             setFFenable( getFFenable() ^ 1);
327             printf("\t\tFeedforward toggled.\n");
328             fflush(stdout);
329             ptui->io_ready = 0;
330             tui_menu();
331             break;
332
333           // ---- FFgain ----------------------------------------------------------
334           case 'l' :
335
336             printf("\t\tEnter new ffgain: ");
337             fflush(stdout);
338             ptui->io_ready = 0;
339
340             // Wait for user input.
341             while(1)
342               if(ptui->io_ready)
343                 break;
344
```

```
345             scanf(" %f", &inFloat);
346             setFFgain(inFloat);
347             tui_menu();
348             ptui->io_ready = 0;
349             break;
350
351           // ---- Test FF ------------------------------------------------
352           case 'p' :
353             printf("\t\tEnter trial name: ");
354             fflush(stdout);
355             ptui->io_ready = 0;
356
357             // Wait for input.
358             while(1)
359               if(ptui->io_ready)
360                 break;
361
362             scanf(" %s", inString);
363             strcat(logFile, inString);
364             printf("\t\tSaving data to %s\n",logFile);
365
366             // Init
367             logFileInit(logFile);
368             circBuffInit();
369
370             // Wait for enter to start saving data
371             printf("\t\tPress enter to start collection...\n");
372             fflush(stdout);
373             ptui->io_ready = 0;
374             while(1)
375               if(ptui->io_ready)
376                 break;
377             scanf(" %c", &inChar);
378
379
380             startFFtest();
381
382             // Wait for enter to stop collection
383             printf("\t\tPress enter to stop collection...\n");
384             fflush(stdout);
385             ptui->io_ready = 0;
386
387             // Data collection
388             while(1){
389
390               logData();
391
392               // Check for input
393               if(ptui->io_ready)
394                 break;
395             }
396             scanf(" %c", &inChar);
397             stopFFtest();
398
399             closeLogFile();
400             logFile[0] = '\0';
401             strcat(logFile, "datalog/");
402             tui_menu();
403             fflush(stdout);
404             ptui->io_ready = 0;
405             break;
406
407           // ---- Step Response ------------------------------------------
408           case 'q' :
409             printf("\t\tEnter trial name: ");
410             fflush(stdout);
411             ptui->io_ready = 0;
412
413             // Wait for input.
414             while(1)
415               if(ptui->io_ready)
416                 break;
417
418             scanf(" %s", inString);
419             strcat(logFile, inString);
420             printf("\t\tSaving data to %s\n",logFile);
421
422             // Init
423             logFileInit(logFile);
424             circBuffInit();
```

```
425
426             printf("\t\tEnter demand current: ");
427             fflush(stdout);
428             ptui->io_ready = 0;
429
430             // Wait for user input.
431             while(1)
432               if(ptui->io_ready)
433                 break;
434
435             scanf(" %f", &inFloat);
436             setStepCurrent(inFloat);
437
438             // Wait for enter to start saving data
439             printf("\t\tPress enter to start collection...\n");
440             fflush(stdout);
441             ptui->io_ready = 0;
442             while(1)
443               if(ptui->io_ready)
444                 break;
445             scanf(" %c", &inChar);
446
447             startStepResponse();
448
449             // Wait for enter to stop collection
450             printf("\t\tPress enter to stop collection...\n");
451             fflush(stdout);
452             ptui->io_ready = 0;
453
454             // Data collection loop
455             while(1){
456               logData();
457
458               // Check for input
459               if(ptui->io_ready)
460                 break;
461             }
462             scanf(" %c", &inChar);
463             printf("\t\tReset step resp. vars.\n");
464             resetStepRespVars();
465
466             closeLogFile();
467             logFile[0] = '\0';
468             strcat(logFile, "datalog/");
469             tui_menu();
470             fflush(stdout);
471             ptui->io_ready = 0;
472             break;
473
474         }
475       }
476     }
477   }
478
479   void logData(void)
480   {
481    //gpio_set_value(gpio_debug, HIGH);
482    circBuffUpdate();
483    writeState();
484    //gpio_set_value(gpio_debug, LOW);
485   }
486
487   int tui_cleanup(void)
488   {
489     if(fcntl(0, F_SETOWN, NULL) == -1){
490       printf("F_SETOWN error.\n");
491       return -1;
492     }
493     if(fcntl(0, F_SETFL, fcntl(0, F_GETFL) & ~O_ASYNC & ~O_NONBLOCK) == -1){
494       printf("Error setting stdin fd flags.\n");
495       return -1;
496     }
497     free(ptui);
498     ptui = NULL;
499
500     printf("TUI cleaned up.\n");
501     return 0;
502   }
```

C.3 PRU0 entry point: pru0_main.c

```
/* Copyright 2017 Jonathan Realmuto

 Licensed under the Apache License, Version 2.0 (the "License");
 you may not use this file except in compliance with the License.
 You may obtain a copy of the License at

     http://www.apache.org/licenses/LICENSE-2.0

 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.
============================================================================*/

#include <stdlib.h>
#include <stdint.h>
#include <pru_cfg.h>
#include <pru_ctrl.h>
#include <pru_iep.h>
#include <pru_intc.h>

#include "rsc_table.h"
#include "mem_types.h"
#include "hw_types.h"

#include "adcdriver.h"
#include "gaitPhase.h"
#include "viconsync.h"
#include "filter.h"


// Prototypes ------------------------------------------------------------------

// ---------------------------------------
// Edit these to customize
// ---------------------------------------
void initialize(void);
void updateState(uint32_t cnt, uint32_t si);
void updateControl(uint32_t cnt, uint32_t si);
void cleanUp(void);

// ---------------------------------------
// Helpers do not modify
// ---------------------------------------
void initMemory(void);
void updateCounters(uint32_t *cnt, uint32_t *si);
void debugPinHigh(void);
void debugPinLow(void);
void iepTimerInit(uint32_t count);
void iepInterruptInit(void);
void startTimer(void);
void clearTimerFlag(void);
void clearIepInterrupt(void);
void pru0ToArmInterrupt(void);

// Globals ---------------------------------------------------------------------
volatile register uint32_t __R30;
volatile register uint32_t __R31;

// Constant Table
volatile far pruIntc CT_INTC
  __attribute__((cregister("PRU_INTC", far), peripheral));

volatile pruCfg CT_CFG
  __attribute__((cregister("PRU_CFG", near), peripheral));

volatile far pruIep CT_IEP
  __attribute__((cregister("PRU_IEP", near), peripheral));

// Shared memory pointer
shared_mem_t *s;

// Param pointer
param_mem_t *p;

// LookUp tables
lookUp_mem_t *l;
```

```
79
80  // Debug Buffer
81  volatile uint32_t *debugBuffer;
82
83  // Main ----------------------------------------------------------------------
84  int main(void)
85  {
86    uint32_t cnt = 0;
87    uint32_t stateIndx = 0;
88
89    initialize();
90
91    // Wait for host (ARM) interrupt
92    while( (__R31 & HOSTO_MASK) == 0){}
93
94    clearIepInterrupt();
95    startTimer();
96
97    // Control Loop
98    while(1){
99
100     // Poll for IEP timer interrupt
101     while((CT_INTC.SECR0 & (1 << 7)) == 0){}
102
103     clearTimerFlag();
104     debugPinHigh();
105
106     s->cntrl_bit.shdw_enable = s->cntrl_bit.enable;
107
108     if(s->cntrl_bit.resetGaitPhase){
109       s->cntrl_bit.resetGaitPhase = 0;
110       gaitPhaseInit();
111     }
112
113     updateState(cnt, stateIndx);
114
115     s->cntrl_bit.pru0_done = 1;
116
117     // Poll for pru1 to be done
118     while(!(s->cntrl_bit.pru1_done));
119
120     // Reset pru1 done bit
121     s->cntrl_bit.pru1_done = 0;
122
123     updateCounters(&cnt, &stateIndx);
124
125     if(!(s->cntrl_bit.shdw_enable))
126         break;
127
128     clearIepInterrupt();
129     debugPinLow();
130   }
131
132   debugPinLow();
133   cleanUp();
134   __halt();
135   return 0;
136 }
137
138 // -----------------------------------------------------------------------------
139 //
140 // initialize(), cleanup(), updateState(), updateControl()
141 // can be modified to customize behavior
142 //
143 // -----------------------------------------------------------------------------
144 void initialize(void)
145 {
146   /*** Init Pru ***/
147
148   // Clear SYSCFG[STANDBY_INIT] to enable OCP master port
149   CT_CFG.SYSCFG_bit.STANDBY_INIT = 0;
150
151   // PRU CTRL: CTR_EN = 0x1 - enable cycle counter
152   HWREG(PRU_CTRL_BASE) |= (1 << 3);
153
154   // Pin Mux
155   CT_CFG.GPCFG0 = 0;
156
157   /*** Memory ***/
158   initMemory();
159
```

```
160     // Init timer
161     iepInterruptInit();
162     iepTimerInit(p->frq_clock_ticks);
163     clearIepInterrupt();
164
165     // Add pru dependent peripheral init methods here
166     adcInit();
167     gaitPhaseInit();
168
169     // Init filter buffers
170     fix16_iirInit(p->filtBuffer[0].x, p->filtBuffer[0].y, MAX_IIR_ORDER+1);
171     fix16_iirInit(p->filtBuffer[1].x, p->filtBuffer[1].y, MAX_IIR_ORDER+1);
172     fix16_iirInit(p->filtBuffer[2].x, p->filtBuffer[2].y, MAX_IIR_ORDER+1);
173     fix16_iirInit(p->filtBuffer[3].x, p->filtBuffer[3].y, MAX_IIR_ORDER+1);
174     fix16_iirInit(p->filtBuffer[4].x, p->filtBuffer[4].y, MAX_IIR_ORDER+1);
175     fix16_iirInit(p->filtBuffer[5].x, p->filtBuffer[5].y, MAX_IIR_ORDER+1);
176   }
177
178   void cleanUp(void)
179   {
180     // Add pru dependent peripheral cleanup methods here
181     adcCleanUp();
182
183     // Clear all interrupts
184     clearIepInterrupt();
185     CT_INTC.SECR0 = 0xFFFFFFFF;
186     CT_INTC.SECR1 = 0xFFFFFFFF;
187   }
188
189   void updateState(uint32_t cnt, uint32_t si)
190   {
191     int16_t adc[8];
192     fix16_t s1, s2, s3, s4, s5, s6;
193
194     s->state[si].timeStamp = cnt;
195     s->state[si].sync = viconSync();
196
197     // Insole samples
198     adcSample_2(adc);
199     adcSample_3(adc);
200
201     // Filter heel sensors for gaitphase
202     s1 = fix16_iir(p->filt.N, p->filt.b, p->filt.a,
203                    p->filtBuffer[0].x, p->filtBuffer[0].y,
204                    adc[4]);
205
206     s2 = fix16_iir(p->filt.N, p->filt.b, p->filt.a,
207                    p->filtBuffer[1].x, p->filtBuffer[1].y,
208                    (int16_t) fix16_to_int(s1));
209
210     s3 = fix16_iir(p->filt.N, p->filt.b, p->filt.a,
211                    p->filtBuffer[2].x, p->filtBuffer[2].y,
212                    (int16_t) fix16_to_int(s2));
213
214
215     s4 = fix16_iir(p->filt.N, p->filt.b, p->filt.a,
216                    p->filtBuffer[3].x, p->filtBuffer[3].y,
217                    adc[7]);
218
219     s5 = fix16_iir(p->filt.N, p->filt.b, p->filt.a,
220                    p->filtBuffer[4].x, p->filtBuffer[4].y,
221                    (int16_t) fix16_to_int(s4));
222
223     s6 = fix16_iir(p->filt.N, p->filt.b, p->filt.a,
224                    p->filtBuffer[5].x, p->filtBuffer[5].y,
225                    (int16_t) fix16_to_int(s5));
226
227
228     // Pack Stuct
229     s->state[si].adc[2] = adc[2];
230     s->state[si].adc[3] = adc[3];
231     s->state[si].adc[4] = (int16_t)fix16_to_int(s2);
232     s->state[si].adc[5] = adc[5];
233     s->state[si].adc[6] = adc[6];
234     s->state[si].adc[7] = (int16_t)fix16_to_int(s5);
235     s->state[si].d_heelForce[0] = (int16_t)fix16_to_int(fix16_ssub(s2, s3));
236     s->state[si].d_heelForce[1] = (int16_t)fix16_to_int(fix16_ssub(s5, s6));
237
```

```
238     // Gait phase
239     leftGaitPhaseDetect(cnt, s->state[si].adc[4], s->state[si].d_heelForce[0]);
240
241     s->state[si].l_meanGaitPeriod = p->l_prevPeriod;
242     s->state[si].l_gaitPhase = p->l_prevGaitPhase;
243     s->state[si].l_hsStamp = p->l_prevHsStamp;
244
245     rightGaitPhaseDetect(cnt, s->state[si].adc[7], s->state[si].d_heelForce[1]);
246
247     s->state[si].r_meanGaitPeriod = p->r_prevPeriod;
248     s->state[si].r_gaitPhase = p->r_prevGaitPhase;
249     s->state[si].r_hsStamp = p->r_prevHsStamp;
250
251     // Motor analog samples
252     adcSample_1(adc);
253     s->state[si].adc[0] = adc[0];
254     s->state[si].adc[1] = adc[1];
255   }
256
257   void updateControl(uint32_t cnt, uint32_t si)
258   {
259
260   }
261
262   // -----------------------------------------------------------------------------
263   // Helper Functions
264   // -----------------------------------------------------------------------------
265   void initMemory(void)
266   {
267     void *ptr = NULL;
268
269     // Memory map for shared memory
270     ptr = (void *)PRU_L_SHARED_DRAM;
271     s = (shared_mem_t *) ptr;
272
273     // Memory map for parameters (pru0 DRAM)
274     ptr = (void *) PRU_DRAM;
275     p = (param_mem_t *) ptr;
276
277     // Memory map for feedforward lookup table (pru1 DRAM)
278     ptr = (void *) PRU_OTHER_DRAM;
279     l = (lookUp_mem_t *) ptr;
280
281     // Point global debug buffer
282     debugBuffer = &(p->debugBuffer[0]);
283   }
284
285   void updateCounters(uint32_t *cnt, uint32_t *si)
286   {
287     // Set buffer location for circ buffer
288     s->stateIndex = (*si);
289
290     (*cnt)++;
291     (*si)++;
292     (*si) %= SIZE_STATE_BUFF;
293   }
294
295   void debugPinHigh(void)
296   {
297     __R30 |= (1 << PRU0_DEBUG_PIN);
298   }
299
300   void debugPinLow(void)
301   {
302     __R30 &= ~(1 << PRU0_DEBUG_PIN);
303   }
304
305   void iepTimerInit(uint32_t count)
306   {
307     // Enable ocp_clk
308     CT_CFG.IEPCLK_bit.OCP_EN = 1;
309
310     // Disable counter
311     CT_IEP.TMR_GLB_CFG_bit.CNT_EN = 0;
312
313     // Reset Count register
314     CT_IEP.TMR_CNT = 0x0;
315
316     // Clear overflow status register
317     CT_IEP.TMR_GLB_STS_bit.CNT_OVF = 0x1;
```

```
318
319     // Set compare value
320     CT_IEP.TMR_CMP0 = (count-1);
321
322     // Clear compare status
323     CT_IEP.TMR_CMP_STS_bit.CMP_HIT = 0xFF;
324
325     // Disable compensation
326     CT_IEP.TMR_COMPEN_bit.COMPEN_CNT = 0x0;
327
328     // Enable CMP0 and reset on event
329     CT_IEP.TMR_CMP_CFG_bit.CMP0_RST_CNT_EN = 1;
330     CT_IEP.TMR_CMP_CFG_bit.CMP_EN = 1;
331   }
332
333   void iepInterruptInit(void)
334   {
335     /*** System event 7 Interrupt -> Host 1 ***/
336
337     // Disable Global enable interrupts
338     CT_INTC.GER = 0;
339
340     // Clear any pending PRU-generated events
341     __R31 = 0x00000000;
342
343     // Map event 7 to channel 2
344     CT_INTC.CMR1_bit.CH_MAP_7 = 1;
345
346     // Map Channel 2 to host 2
347     CT_INTC.HMR0_bit.HINT_MAP_1 = 1;
348
349     // Clear the status of all interrupts
350     CT_INTC.SECR0 = 0xFFFFFFFF;
351     CT_INTC.SECR1 = 0xFFFFFFFF;
352
353     // Enable event 7
354     CT_INTC.EISR = 7;
355
356     // Enable Host interrupt 1
357     CT_INTC.HIEISR |= (1 << 0);
358
359     // Gloablly enable interrupts
360     CT_INTC.GER = 1;
361   }
362
363   void startTimer(void)
364   {
365     // Start Timer
366     CT_IEP.TMR_GLB_CFG = 0x11;
367   }
368
369   void clearTimerFlag(void)
370   {
371     // Clear compare status
372     CT_IEP.TMR_CMP_STS_bit.CMP_HIT = 0xFF;
373   }
374
375   void pru0ToArmInterrupt(void)
376   {
377     __R31 = PRU0_ARM_INT;
378   }
379
380   void clearIepInterrupt(void)
381   {
382       // Clear interrupt status
383       CT_INTC.SECR0 = (1<<7);
384       __R31 = 0x00000000;
385   }
```

C.4 PRU1 entry point: pru1_main.c

```
1   /* Copyright 2017 Jonathan Realmuto
2
3    Licensed under the Apache License, Version 2.0 (the "License");
4    you may not use this file except in compliance with the License.
5    You may obtain a copy of the License at
6
7        http://www.apache.org/licenses/LICENSE-2.0
8
```

```
 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.
=============================================================================*/

#include <stdlib.h>
#include <stdint.h>
#include <pru_cfg.h>
#include <pru_ctrl.h>
#include <pru_iep.h>
#include <pru_intc.h>

#include "rsc_table.h"
#include "mem_types.h"
#include "hw_types.h"

#include "spidriver.h"
#include "encoder.h"
#include "maxonmotor.h"
#include "fix16.h"

#include "mpu9150imu.h"
#include "pwmdriver.h"
#include "gaitPhase.h"

#define FIX16_1000  0x3E80000

// Prototypes ------------------------------------------------------------------

// ---------------------------------------
// Edit these to customize
// ---------------------------------------
void initialize(void);
void updateState(uint32_t cnt, uint32_t si);
void updateControl(uint32_t cnt, uint32_t si);
void cleanUp(void);

// ---------------------------------------
// Helpers
// ---------------------------------------
void calibratePWMcmp2current(uint32_t cnt, uint32_t si);
void initMemory(void);
void updateCounters(uint32_t *cnt, uint32_t *si);
void debugPinHigh(void);
void debugPinLow(void);
void interruptInit(void);
void clearInterrupt(void);


// Globals ---------------------------------------------------------------------
volatile register uint32_t __R30;
volatile register uint32_t __R31;

volatile far pruIntc CT_INTC
  __attribute__((cregister("PRU_INTC", far), peripheral));

volatile pruCfg CT_CFG
  __attribute__((cregister("PRU_CFG", near), peripheral));

volatile far pruIep CT_IEP
  __attribute__((cregister("PRU_IEP", near), peripheral));

// State pointer
shared_mem_t *s;

// Param pointer
param_mem_t *p;

// Feedforward lookup table pointer
lookUp_mem_t *l;

// Debug Buffer
volatile uint32_t *debugBuffer;

// Main ------------------------------------------------------------------------
int main(void)
{
  uint32_t cnt = 0;
  uint32_t stateIndx = 0;
```

```
90
91      initialize();
92
93      // Uncomment, make, run, recomment to tare.
94      //encoderSetZeroAngle();
95
96      // Wait for host interrupt
97      while( (__R31 & HOST1_MASK) == 0){}
98
99      clearInterrupt();
100
101     // Control Loop
102     while(1){
103
104       // Poll of IEP timer interrupt
105       while((CT_INTC.SECR0 & (1 << 7)) == 0);
106
107       debugPinHigh();
108
109       updateState(cnt, stateIndx);
110
111       debugPinLow();
112
113       // Poll for pru0 done bit
114       while(!(s->cntrl_bit.pru0_done));
115
116       debugPinHigh();
117
118       // Reset pru0 done bit
119       s->cntrl_bit.pru0_done = 0;
120
121       // Update Control
122       // use to calibrate pwm instead of updateControl
123       //calibratePWMcmp2current(cnt, stateIndx);
124       updateControl(cnt, stateIndx);
125
126       s->cntrl_bit.pru1_done = 1;
127
128       updateCounters(&cnt, &stateIndx);
129
130       // Check enable bit
131       if(!(s->cntrl_bit.shdw_enable))
132           break;
133
134       // Wait till interrupt is cleared
135       while(CT_INTC.SECR0 & (1 << 7));
136
137       debugPinLow();
138     }
139     debugPinLow();
140     cleanUp();
141     __halt();
142     return 0;
143   }
144
145   // -------------------------------------------------------------------------
146   //
147   // initialize(), cleanup(), updateState(), updateControl()
148   // can be modified to customize behavior
149   //
150   // -------------------------------------------------------------------------
151   void initialize(void)
152   {
153     /*** Init Pru ***/
154
155     // Clear SYSCFG[STANDBY_INIT] to enable OCP master port
156     CT_CFG.SYSCFG_bit.STANDBY_INIT = 0;
157
158     // Pin Mux
159     CT_CFG.GPCFG0 = 0;
160
161     // Zero detect flags
162     p->encoderDetect = 0;
163   //  p->imuDetect = 0;
164
165     // Memory and Interrupt
166     initMemory();
167     interruptInit();
168
169     // Modules
170     spiInit();
```

```
171
172     // Add pru dependent peripheral init methods here
173     if (encoderInit() == 0)
174       p->encoderDetect = 1;
175     else
176       p->encoderDetect = 0;
177
178   //  if (imuInit() == 0)
179   //    p->imuDetect = 1;
180   //  else
181   //    p->imuDetect = 0;
182
183     motorInit();
184
185     // Zero some stuff
186     p->stepRespCnt = 0;
187     p->stepRespFlag = 0;
188     p->stepCurrent = 0;
189     p->FFgain = 0;
190   }
191
192
193   void updateControl(uint32_t cnt, uint32_t si)
194   {
195     fix16_t u_fb = 0;
196     fix16_t u_ff = 0;
197
198     uint32_t t_cnts, t1;
199
200     // Step Response
201     if (s->cntrl_bit.stepResp == 1){
202
203       if (!(p->stepRespFlag == 1)){
204         p->stepRespCnt = cnt + 1000; // Delay 1 sec before step starts
205         p->stepRespFlag = 1;
206         u_ff = 0;
207         u_fb = 0;
208       }
209
210       else if ( (p->stepRespFlag) & (cnt >= p->stepRespCnt) ){
211         u_ff = p->stepCurrent;
212         u_fb = 0;
213       }
214       else {
215         u_ff = 0;
216         u_fb = 0;
217       }
218     }
219
220     // Feedforward Test
221     else if (s->cntrl_bit.testFF == 1){
222
223       // Bias
224       u_fb = p->u_bias;
225
226       // If first call, store cnt (so waveform starts at 0)
227       if (s->cntrl_bit.testFF_flag == 0) {
228         s->cntrl_bit.testFF_flag = 1;
229         p->FFtestT0 = cnt;
230       }
231
232       // Calculate lut index t = (cnt % Tp) * (1/Tp * 1000)
233       t_cnts = ((cnt - p->FFtestT0) % 1000) * 1; // 1Hz
234       // t_cnts = ((cnt - p->FFtestT0) % 500) * 2; // 2 Hz
235       //t_cnts = ((cnt - p->FFtestT0) % 250) * 4; // 4 Hz
236
237       s->state[si].l_percentGait = t_cnts;
238
239       // Scale ff
240       u_ff = fix16_smul(p->FFgain,
241                   fix16_sdiv(fix16_from_int(l->u_ff[t_cnts]), FIX16_1000));
242     }
243
244     // Impedance and feedforward
245     else{
246
247       // Impedance feedback
248       //u_fb = fix16_smul(p->Kp, fix16_ssub(p->anklePos0, s->state[si].anklePos));
249
250       // Bias
251       u_fb = p->u_bias;
```

```
252
253
254       // Feedforward
255       if ((s->cntrl_bit.doFeedForward) && (p->gaitDetectReady)){
256
257
258         if (p->isProsLeft) {
259
260           // Time since hs
261           t1 = (cnt - (s->state[si].l_hsStamp - p->hs_delay)) * 1000;
262
263           t_cnts = t1 / (s->state[si].l_meanGaitPeriod -
264                          s->state[si].l_meanGaitPeriod / 1000);
265
266           // Saturate t_cnts;
267           if (t_cnts >= NUM_FF_LT)
268             t_cnts = NUM_FF_LT-1;
269
270           // Store percent gait
271           s->state[si].l_percentGait = t_cnts;
272         }
273         else {
274           // Time since hs
275           t1 = (cnt - (s->state[si].r_hsStamp - p->hs_delay)) * 1000;
276
277           t_cnts = t1 / (s->state[si].r_meanGaitPeriod -
278                          s->state[si].r_meanGaitPeriod / 1000);
279
280           // Saturate t_cnts;
281           if (t_cnts >= NUM_FF_LT)
282             t_cnts = NUM_FF_LT-1;
283
284           // Store percent gait
285           s->state[si].r_percentGait = t_cnts;
286         }
287
288         // Scale ff
289         u_ff = fix16_smul(p->FFgain,
290                     fix16_sdiv(fix16_from_int(l->u_ff[t_cnts]), FIX16_1000));
291       }
292     }
293
294     // Send command/store command
295     motorSetCurrent(fix16_sadd(u_ff,u_fb), &s->state[si].motorCmpValue);
296     s->state[si].u_ff = u_ff;
297     s->state[si].u_fb = u_fb;
298   }
299
300   void updateState(uint32_t cnt, uint32_t si)
301   {
302     // encoder
303     if (p->encoderDetect)
304       encoderSample(&(s->state[si].anklePos));
305
306     // imu
307   //  if (p->imuDetect)
308   //    imuSample(s->state[si].imu);
309   }
310
311   void cleanUp(void)
312   {
313     // Add pru dependent peripheral cleanup methods here
314     encoderCleanUp();
315     motorCleanUp();
316   //  imuCleanUp();
317
318     // Cleanup modules
319     spiCleanUp();
320
321     // Clear all interrupts
322     clearInterrupt();
323     CT_INTC.SECR0 = 0xFFFFFFFF;
324     CT_INTC.SECR1 = 0xFFFFFFFF;
325   }
326
327   // -------------------------------------------------------------------------
328   // Helper Functions
329   // -------------------------------------------------------------------------
330   void updateCounters(uint32_t* cnt, uint32_t* si)
331   {
```

```
332     (*cnt)++;
333     (*si)++;
334     (*si) %= SIZE_STATE_BUFF;
335   }
336
337   void debugPinHigh(void)
338   {
339     __R30 |= (1 << PRU1_DEBUG_PIN);
340   }
341
342   void debugPinLow(void)
343   {
344     __R30 &= ~(1 << PRU1_DEBUG_PIN);
345   }
346
347   void initMemory(void)
348   {
349     void *ptr = NULL;
350
351     // Memory map for shared memory
352     ptr = (void *)PRU_L_SHARED_DRAM;
353     s = (shared_mem_t *) ptr;
354
355     // Memory map for parameters (pru0 DRAM)
356     ptr = (void *) PRU_OTHER_DRAM;
357     p = (param_mem_t *) ptr;
358
359     // Memory map for feedforward lookup table (pru1 DRAM)
360     ptr = (void *) PRU_DRAM;
361     l = (lookUp_mem_t *) ptr;
362
363     // Point global debug buffer
364     debugBuffer = &(p->debugBuffer[0]);
365   }
366
367   void interruptInit(void)
368   {
369     // Disable Global enable interrupts
370     CT_INTC.GER = 0;
371
372     // Clear any pending PRU-generated events
373     __R31 = 0x00000000;
374
375     // Clear the status of all interrupts
376     CT_INTC.SECR0 = 0xFFFFFFFF;
377     CT_INTC.SECR1 = 0xFFFFFFFF;
378
379     // Gloablly enable interrupts
380     CT_INTC.GER = 1;
381   }
382
383   void clearInterrupt(void)
384   {
385     CT_INTC.SECR0 = 0xFFFFFFFF;
386     CT_INTC.SECR1 = 0xFFFFFFFF;
387      __R31 = 0x00000000;
388   }
389
390
391   void calibratePWMcmp2current(uint32_t cnt, uint32_t si)
392   {
393     if (cnt > 10000){
394       uint16_t val = 10000;
395       s->state[si].motorCmpValue = val;
396       pwmSetCmpValue(val);
397     }
398     else
399       s->state[si].motorCmpValue = 5000;
400   }
```

C.5 Heel-strike detection: gaitPhase.c

```
1   /* Copyright 2017 Jonathan Realmuto
2
3    Licensed under the Apache License, Version 2.0 (the "License");
4    you may not use this file except in compliance with the License.
5    You may obtain a copy of the License at
```

```
6
7       http://www.apache.org/licenses/LICENSE-2.0
8
9    Unless required by applicable law or agreed to in writing, software
10   distributed under the License is distributed on an "AS IS" BASIS,
11   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12   See the License for the specific language governing permissions and
13   limitations under the License.
14  ============================================================================*/
15
16  #include <stdint.h>
17
18  #include "gaitPhase.h"
19
20
21
22  void gaitPhaseInit(void)
23  {
24    for (int i=1; i>3; i++){
25      p->l_period[i] = 0;
26      p->r_period[i] = 0;
27    }
28
29    p->numOfSteps = 0;
30    p->gaitDetectReady = 0;
31
32    p->l_prevGaitPhase= 0;
33    p->r_prevGaitPhase = 0;
34    p->l_prevHsStamp = 0;
35    p->r_prevHsStamp = 0;
36    p->l_prevPeriod = 0;
37    p->r_prevPeriod = 0;
38  }
39
40  void leftGaitPhaseDetect(uint32_t cnt,
41                           volatile int16_t heelForce,
42                           volatile int16_t d_heelForce)
43  {
44    uint32_t gp = 0;
45    uint32_t hsStamp = 0;
46    uint32_t meanPeriod = 0;
47
48    if (p->l_prevGaitPhase == 0){
49      if ((d_heelForce > p->l_d_forceThrs) && (heelForce < p->l_forceThrs)){
50        gp = 1;
51        hsStamp = cnt;
52
53        // Find mean period
54        p->l_period[2] = p->l_period[1];
55        p->l_period[1] = p->l_period[0];
56        p->l_period[0] = fix16_from_int(hsStamp - p->l_prevHsStamp);
57
58        meanPeriod = (uint32_t) fix16_to_int(
59          fix16_sdiv(fix16_sadd(fix16_sadd(p->l_period[0],
60                         p->l_period[1]), p->l_period[2]), fix16_from_int(3)));
61
62        if (p->isProsLeft) {
63        // Increment step
64          p->numOfSteps++;
65          if (p->numOfSteps > 5)
66            p->numOfSteps = 5;
67
68          if (p->numOfSteps >= 5)
69            p->gaitDetectReady = 1;
70        }
71
72      }
73      else{
74        gp = 0;
75        hsStamp = p->l_prevHsStamp;
76        meanPeriod = p->l_prevPeriod;
77      }
78    }
79    else if (p->l_prevGaitPhase == 1){
80      if ((d_heelForce < (-p->l_d_forceThrs)) && (heelForce > p->l_forceThrs)){
81        gp = 0;
82        hsStamp = p->l_prevHsStamp;
83        meanPeriod = p->l_prevPeriod;
84
85      }
86      else{
```

```
87          gp = 1;
88          hsStamp = p->l_prevHsStamp;
89          meanPeriod = p->l_prevPeriod;
90        }
91      }
92
93      // Store Results
94      p->l_prevPeriod = meanPeriod;
95      p->l_prevGaitPhase = gp;
96      p->l_prevHsStamp = hsStamp;
97    }
98
99    void rightGaitPhaseDetect(uint32_t cnt,
100                              volatile int16_t heelForce,
101                              volatile int16_t d_heelForce)
102   {
103     uint32_t gp = 0;
104     uint32_t hsStamp = 0;
105     uint32_t meanPeriod = 0;
106
107     if (p->r_prevGaitPhase == 0){
108       if ((d_heelForce > p->r_d_forceThrs) && (heelForce < p->r_forceThrs)){
109         gp = 1;
110         hsStamp = cnt;
111
112         // Find mean period
113         p->r_period[2] = p->r_period[1];
114         p->r_period[1] = p->r_period[0];
115         p->r_period[0] = fix16_from_int(hsStamp - p->r_prevHsStamp);
116
117         meanPeriod = (uint32_t) fix16_to_int(
118           fix16_sdiv(fix16_sadd(fix16_sadd(p->r_period[0],
119                         p->r_period[1]), p->r_period[2]), fix16_from_int(3)));
120
121
122         if (!p->isProsLeft){
123           // Increment step
124           p->numOfSteps++;
125           if (p->numOfSteps > 5)
126             p->numOfSteps = 5;
127
128           if (p->numOfSteps >= 5)
129             p->gaitDetectReady = 1;
130         }
131
132       }
133       else{
134         gp = 0;
135         hsStamp = p->r_prevHsStamp;
136         meanPeriod = p->r_prevPeriod;
137       }
138     }
139     else if (p->r_prevGaitPhase == 1){
140       if ((d_heelForce < (-p->r_d_forceThrs)) && (heelForce > p->r_forceThrs)){
141         gp = 0;
142         hsStamp = p->r_prevHsStamp;
143         meanPeriod = p->r_prevPeriod;
144
145       }
146       else{
147         gp = 1;
148         hsStamp = p->r_prevHsStamp;
149         meanPeriod = p->r_prevPeriod;
150       }
151     }
152
153     // Store Results
154     p->r_prevPeriod = meanPeriod;
155     p->r_prevGaitPhase = gp;
156     p->r_prevHsStamp = hsStamp;
157   }
```

C.6 Memory structures: mem_types.h

```
1   /* Copyright 2017 Jonathan Realmuto
2
3    Licensed under the Apache License, Version 2.0 (the "License");
4    you may not use this file except in compliance with the License.
```

```
 5    You may obtain a copy of the License at
 6
 7        http://www.apache.org/licenses/LICENSE-2.0
 8
 9    Unless required by applicable law or agreed to in writing, software
10    distributed under the License is distributed on an "AS IS" BASIS,
11    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12    See the License for the specific language governing permissions and
13    limitations under the License.
14   ============================================================================*/
15
16   #ifndef _MEM_TYPES_
17   #define _MEM_TYPES_
18
19   #include "fix16.h"
20
21   // Constants
22   #define PRU_CLK         (200000000)
23   #define PWM_CLK         (100000000)
24   #define SPI_CLK         (48000000)
25   #define I2C1_CLK        (48000000)
26
27   #define HOST0_MASK    (0x40000000)
28   #define HOST1_MASK    (0x80000000)
29
30   #define PRU0_ARM_INT  (19 + 16)
31   #define PRU1_ARM_INT  (20 + 16)
32
33   #define SIZE_STATE_BUFF   149
34
35   #define NUM_ADC     8
36   #define NUM_IMU     6
37
38   #define NUM_FF_LT         1000
39
40   // Global addresses to hardware modules
41   #define PRU_CTRL_BASE 0x00022000
42   #define ADC_BASE      0x44E0D000
43   #define SPI1_BASE     0x481A0000
44   #define PWM_SS2_BASE  0x48304000
45   #define EPWM2_BASE    0x48304200
46   #define I2C1_BASE     0x4802A000
47
48   // Debug pins
49   #define PRU0_DEBUG_PIN  5
50   #define PRU1_DEBUG_PIN  8
51
52   #define MAX_IIR_ORDER   3
53
54   #define PRU0 0
55   #define PRU1 1
56
57
58   // ADC mux
59   // 0 - motor current
60   // 1 - motor velocity
61   // 2 - Force sensor 1.1
62   // 3 - Force sensor 1.2
63   // 4 - Force sensor 1.3
64   // 5 - Force sensor 2.1
65   // 6 - Force sensor 2.2
66   // 7 - Force sensor 2.3
67
68
69   // Structures ------------------------------------------------------------------
70
71   // Anklebot state
72   typedef struct{
73     volatile uint32_t timeStamp;
74     volatile uint32_t sync;
75     volatile uint32_t r_hsStamp;
76     volatile uint32_t l_hsStamp;
77
78     volatile uint16_t r_meanGaitPeriod;
79     volatile uint16_t l_meanGaitPeriod;
80     volatile uint16_t r_percentGait;
81     volatile uint16_t l_percentGait;
82     volatile uint16_t l_gaitPhase;
83     volatile uint16_t r_gaitPhase;
84
85     volatile uint32_t motorCmpValue;
86     volatile fix16_t anklePos;
```

```
87    volatile fix16_t ankleVel;
88    volatile fix16_t u_fb;
89    volatile fix16_t u_ff;
90
91    volatile int16_t adc[NUM_ADC];
92    volatile int16_t imu[NUM_IMU];
93    volatile int16_t d_heelForce[2];
94  } state_t;
95
96
97  // Shared Memory -> mapped to SRAM
98  typedef struct{
99    state_t state[SIZE_STATE_BUFF];
100   uint32_t stateIndex;
101
102   // Flow control
103   union{
104     volatile uint16_t cntrl;
105
106     volatile struct{
107       unsigned enable : 1;          // bit 0 (set by ARM and shadowed)
108       unsigned pru0_done : 1;       // bit 1 (set by pru0, read/reset by pru1)
109       unsigned pru1_done : 1;       // bit 2 (set by pru1, read/reset by pru0)
110       unsigned shdw_enable : 1;     // bit 3 (shawdow reg. for enable)
111       unsigned encoderTare : 1;     // bit 4 (set by arm, read/reset by pru1)
112       unsigned doFeedForward : 1;   // bit 5 (set by arm, read by pru1)
113       unsigned gaitPhaseReady: 1;   // bit 6 (set by pru0, read by pru1)
114       unsigned resetGaitPhase : 1;  // bit 7
115       unsigned testFF : 1;          // bit 8 (set by arm, reset by arm)
116       unsigned testFF_flag : 1;     // bit 9 (set by pru1)
117       unsigned stepResp : 1;        // bit 10 (set by arm, reset by pru1)
118       unsigned rsvd : 5;            // bits 11-15 reserved
119    } cntrl_bit;
120   };
121 } shared_mem_t;
122
123 // Filter IIR buffers
124 typedef struct{
125   volatile fix16_t x[MAX_IIR_ORDER+1];
126   volatile fix16_t y[MAX_IIR_ORDER+1];
127 } iir_buff_t;
128
129 // IIR Coefficients
130 typedef struct{
131   uint32_t N;
132   fix16_t b[MAX_IIR_ORDER+1];
133   fix16_t a[MAX_IIR_ORDER+1];
134 } iir_coeff_t;
135
136 // Parameter Struct -> mapped to pr0 DRAM
137 typedef struct{
138   volatile uint32_t frq_hz;
139   volatile uint32_t frq_clock_ticks;
140   volatile uint32_t mass;
141
142   volatile uint32_t hs_delay;
143   volatile fix16_t u_bias;
144   volatile uint32_t isProsLeft;
145
146   volatile int32_t l_forceThrs;
147   volatile int32_t r_forceThrs;
148   volatile int32_t l_d_forceThrs;
149   volatile int32_t r_d_forceThrs;
150
151   volatile uint32_t l_prevGaitPhase;
152   volatile uint32_t r_prevGaitPhase;
153   volatile uint32_t l_prevHsStamp;
154   volatile uint32_t r_prevHsStamp;
155   volatile uint32_t l_prevPeriod;
156   volatile uint32_t r_prevPeriod;
157
158   volatile fix16_t l_period[3];
159   volatile fix16_t r_period[3];
160   volatile uint32_t numOfSteps;
161   volatile uint32_t gaitDetectReady;
162
163   volatile uint32_t stepRespCnt;
164   volatile uint32_t stepRespFlag;
165   volatile fix16_t stepCurrent;
166
```

```
167     volatile fix16_t FFgain;
168     volatile fix16_t FFtestT0;
169
170     volatile uint32_t encoderDetect;
171     volatile uint32_t imuDetect;
172
173     iir_coeff_t filt;
174     iir_buff_t filtBuffer[6];
175
176     volatile uint32_t debugBuffer[10];
177   } param_mem_t;
178
179
180   // Feedforward lookup table -> mapped to pru1 DRAM
181   typedef struct{
182     int16_t u_ff[NUM_FF_LT];
183   } lookUp_mem_t;
184
185   #endif
```

C.7 Sample subject file: PA_A03

```
1   1000  // Freq.
2   0     // Freq. Ticks
3   76   // Subject Mass
4   30   // hs_delay
5   1   // u_bias
6   1       // isProsLeft
7   100   // l_forceThrs
8   20     // l_d_forceThrs
9   100    // r_forceThrs
10  20     // r_d_forceThrs
```

Appendix D

EMBEDDED SYSTEM HARDWARE

Figure D.1 shows the signal routing and PWM filter circuits. This circuit attached to the Beaglebone black via the pin headers (e.g., as as a *cape*). The insole sensor circuit is shown in Fig. D.2. The on board processors (e.g., Beaglebone black) with custom circuits is shown in Fig. D.3. The wearable enclosure, worn on the back of the user and secured with Velcro, is shown in Fig. D.4.

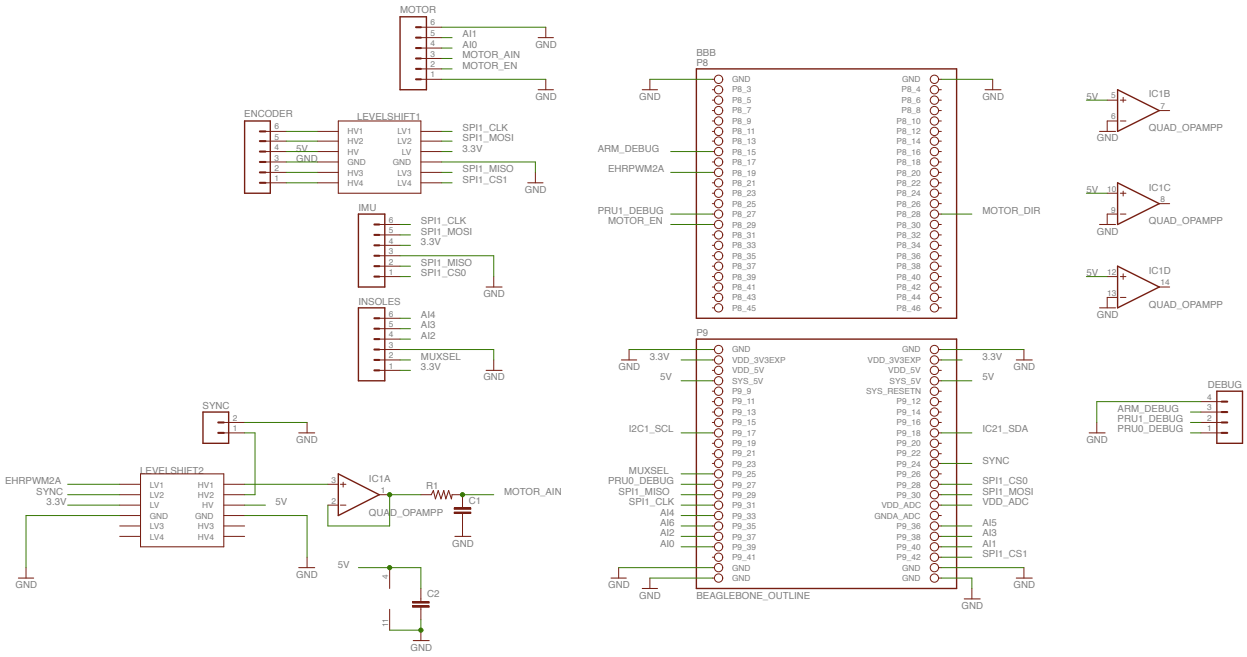


Figure D.1: Custom circuit used to route signals, including PWM filter.

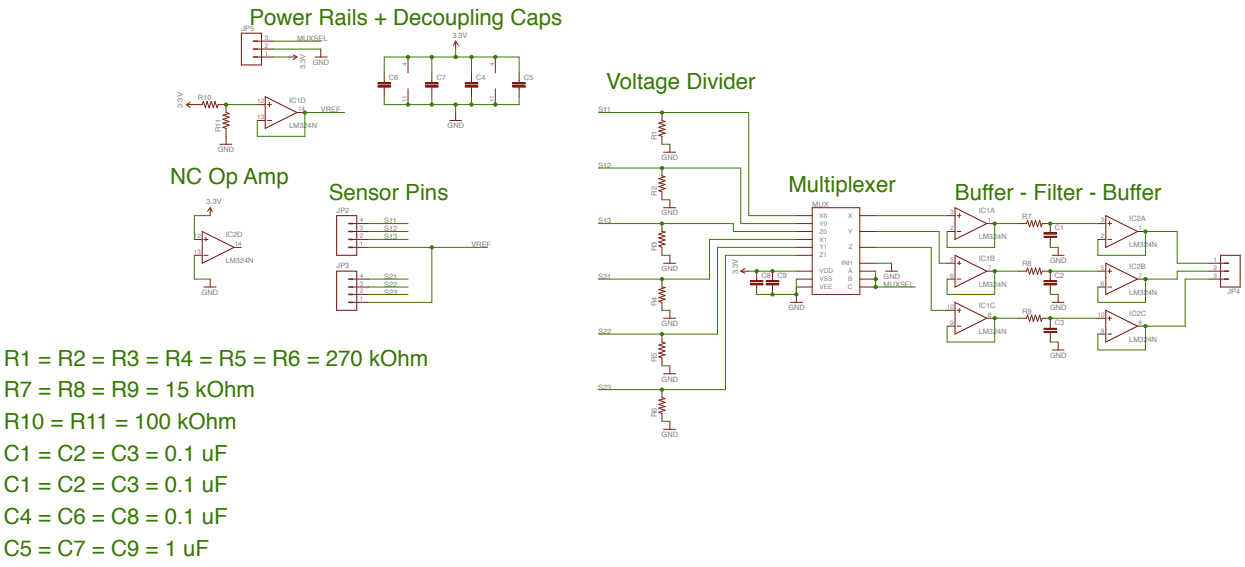


Figure D.2: Insole sensor circuit composed of excitation signal, multiplexer and voltage dividers. Note the original design is flawed since the analog filters appear after the multiplexer. The transients in the capacitors render the filters unusable, thus the signals where tapped after the multiplexer. Future designs could include analog filters before the multiplexer.

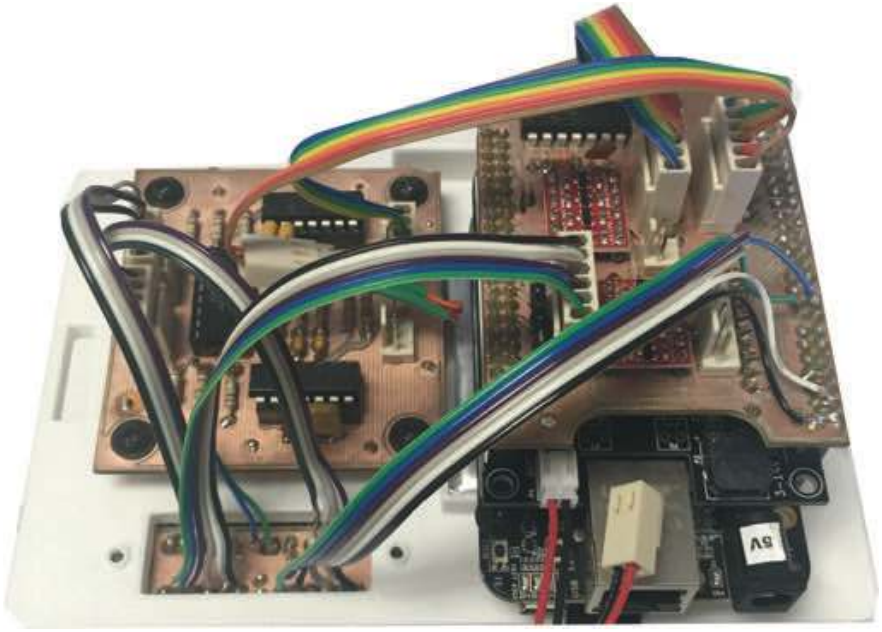


Figure D.3: Image of the embedded system with custom circuits secured to 3D printed enclosure (top not shown).

Figure D.4: Enclosure with elastic Velcro for attaching to the user. Maxon motor drive can attached to the outside of the enclosure.

Appendix E

INSTITUTIONAL REVIEW BOARD CONSENT FORM

Department of Veterans Affairs	VA PUGET SOUND HEALTH CARE SYSTEM (663) RESEARCH CONSENT FORM
SUBJECT NAME	
TITLE OF STUDY	**Ambulation Research Protocol Development**
PRINCIPAL INVESTIGATOR	**Glenn K. Klute, PhD**

LAY TITLE: Motion Analysis Lab Project Development

Researchers:

Glenn Klute, PhD Principal Investigator (206) 277-6724

For study related questions call: 206-764-2962

You are being invited to participate in a research study. The purpose of this consent form is to give you the information you will need to help you decide whether to be in the study. Please read this form carefully. You may ask questions about the purpose of the research, what we would ask you to do, the possible risks and benefits, your rights as a volunteer, and anything else about the research or this form that is not clear. When we have answered all your questions, you can decide whether you want to be in the study. You are free to discuss this with friends or family. This process is called "informed consent." We will give you a copy of this form once it is signed for your records.

1. Purpose of research study and how long it will last: You are being asked to participate in a research study because we wish to understand how people walk, both those with and without disability. The project has many components, and you will be asked to participate in only some of the methods we are developing to measure walking behavior. The information we gather will be used to help us design more complex experiments that will help people with walking difficulties walk better. This study is sponsored by the Department of Veterans Affairs. The researchers have no financial interest or other possible conflict of interest with regard to the study. We will enroll up to 400 subjects in this study, up to 380 of which will be from the Veterans Affairs Puget Sound Health Care System ("VA Puget Sound").

2. Description of the study including procedures to be used: The procedures in which you will be asked to participate are marked below with an "X." For each individual procedure, you may spend between 30 and 90 minutes being tested. If you agree to wear the step-count device, you may wear the device for up to a month. Depending on the number of tests you participate in, the total time will vary. You will not spend more than 3 hours in the lab at any one time.

All procedures will be performed at the VA Puget Sound.

☐ **Novel Prosthetic or Orthotic Device.** You will be asked to wear one or several different prosthetic and/or orthotic device(s). You will be asked to wear each device until you feel

confident and stable enough to perform the procedures checked below. If you are not able to obtain confidence and stability on any of the devices, we will stop the session.

☐ **Novel Shoe.** You will be asked to wear one or several different shoe(s). You will be asked to wear each shoe until you feel confident and stable enough to perform the procedures checked below. If you are unable to feel confident and stable on any of the shoes, we will stop the session.

☐ **Novel Instrumented Sock(s).** You will be asked to wear one or more instrumented sock(s). The sock(s) are instrumented with pressure sensors that will measure the pressure created on the bottom of your foot as you walk. The sock(s) will also record the number of steps you take when you walk. The pressures and steps will be recorded on a cell phone. If you are unable to feel comfortable wearing the sock(s), we will stop the session.

☐ **Ground Reaction Force Measurement Procedure.** For 3–10 times, you will need to:

Check each one that applies:

_____ stand & balance
_____ walk
_____ run
_____ jump

_____ barefoot
_____ in shoes

across force measurement platforms that are mounted flush with the floor. The entire procedure will take up to 30 minutes to complete.

☐ **Platform Plantar Pressure Measurement Procedure.** For 3–4 times, you will need to:

Check each one that applies:

_____ walk
_____ run
_____ jump

barefoot across a pressure measurement platform that is mounted flush with a walkway. The entire procedure will take up to 30 minutes to complete.

☐ **In-Shoe Plantar Pressure Measurement Procedure.** You will need to:

Check each one that applies:

_____ walk
_____ run
_____ jump

while wearing shoes fitted with insoles which record foot pressure Care will be taken to ensure your foot is comfortable with the insoles of the shoe. A small box worn on a belt around your waist will record the pressure as you walk. The entire procedure will take up to 30 minutes to complete.

☐ **Motion Measurement Using Optical Motion Detection System Procedure.** First, you will be asked to change into the provided tight-fitting shorts and T-shirt. Next, we will take a standard set of body measurements (such as your height, weight, leg length, foot length) using calipers, a measuring tape, and a standard scale. We will then attach small reflective markers to your body using double-sided tape and ask you to:

Check each one that applies:

_____ stand & balance
_____ walk
_____ run
_____ step over low obstacles
_____ jump

} several times as the motion of each marker is recorded by infrared cameras. Video cameras will also record the movement. The video will serve as a visual record to aid in the analysis and interpretation of the data. The entire procedure will take up to 60 minutes to complete.

A total of *(write #)* _____ measurement boxes will be attached to your body over the following segments using Velcro or compressive athletic wraps (Co-Flex™).

Body segments (one box per segment):

_____

_____

_____

_____

_____

Cables connect the instruments to a portable transmitter that you will wear around your waist using a belt. Electrical signals corresponding to segment motion will be continuously transmitted via radio waves to a computer-connected receiver. This procedure will be in conjunction with other procedures listed above and may take up to 1 hour to complete.

☐ **Locomotion and Cognitive Function Procedure.** While walking, you will answer simple questions or perform simple mental tasks. For example, you may be asked to count backward from 100 by 7s or to list all the animals you can think of. Audio recordings will be taken to record the accuracy of your answers. This procedure may be combined with other procedures to determine the effects on mental tasks upon your movement.

☐ **Metabolic Energy Measurement Procedure.** The air you breathe will be monitored for oxygen and carbon dioxide while you walk or run on a treadmill with a mouthpiece or over-ground wearing a mask and small vest. After a period of resting quietly, you will begin to exercise for 3 to 10 minutes. You can stop and rest if you get tired. The exercise and resting may be repeated up to five times. The entire procedure will take up to 90 minutes to complete.

☐ **Telemetered Surface Electromyography Procedure.** First, your skin will be shaved, abraded, and wiped clean with a sterile alcohol swab. A total of *(write #)* _____ disposable electrodes will be attached to your skin over the following muscles using medical double-sided tape.

Muscles and number of electrodes per muscle:

_____

_____

_____

_____

_____

_____

_____

Cables connect the electrodes to a portable transmitter that you will wear around your waist using a belt. Electrical signals corresponding to muscle activity will be continuously transmitted via radio waves to a computer-connected receiver. This procedure will be in conjunction with other procedures listed above and may take up to 1½ hours to complete.

☐ **Residual Limb Volume and Shape Scanning.** We will use a handheld scanner to measure the shape of your residual limb. First, we may ask to don body fitting shorts that we will provide. We will ask you to doff your prosthesis and rest while sitting for up to ten minutes to allow your residual limb to adjust to its usual size without a prosthesis. We will then ask you to stand (with supports such as crutches, handrails, assistance of study staff, etc.). Once you are in a good position, we will ask you to hold your residual limb still while we scan it with the handheld instrument. We may scan your limb more than one time; however, we will tell you before we begin the scan so you may relax between each scan. The entire procedure will take up to 30 minutes.

☐ **Split-belt Treadmill Procedure.** You will be asked to walk on a treadmill with two belts, one for each leg. You may be asked to:

- Walk with longer, shorter, wider, or narrower steps than you would normally.
- Walk in rhythm to a metronome.
- Walk on the treadmill as the belts move at different speeds.

Force measurement platforms under the treadmill belts will record the forces from your walking patterns. If a new kind of propulsive prosthetic or orthotic device is being tested, you may be asked to wear a safety harness attached to the ceiling in case you lose your balance. This procedure may be used in conjunction with other procedures. The entire procedure will take up to 90 minutes.

☐ **Load Carriage Procedure.** You will be asked to walk over ground or on our treadmill while wearing a pack with up to 16 kg (35 lbs) of additional weight. You may be asked to:

- Walk with the pack on your back like a traditional backpack.
- Walk with the pack on your front to simulate the weight gain of pregnancy.

The entire procedure will take up to 90 minutes.

☐ **Heart Rate Monitor.** We will ask you to wear a heart rate monitor. The heart rate monitor consists of a chest strap that you will wear beneath your clothes and a wrist monitor that you will wear on your wrist. The heart rate monitor will provide information about your activity levels while you are participating in one or more of the study procedures described above.

☐ **Video and/or Photographic Recording Procedure.** A video recording and/or photograph will be made to document your movements. This will help us to make sure the sensors were in the right place and your movement was correct. If this record is used in a scientific presentation or publication, your identity will be obscured.

☐ **Step Monitoring Procedure.** A small plastic box will be strapped to your ankle to count the number of steps you take. This will be worn whenever you are awake for _____ days. Please return it to the lab or mail the device back to us in the envelope provided. While wearing this device, please do not get it wet. Be sure to take the monitor off before bathing or showering.

☐ **Cross-slope Procedure.** You will be asked to walk across a raised walkway consisting of a flat beginning section, a middle section with a cross-sloping ramp, and a flat ending section. A cross-slope means the right side of the ramp will be higher than the left side, or vice versa. The cross-sloping ramp will be slightly beneath the level of the walkway and have an angle roughly 2–3 times that of a typical sidewalk ramp.

During the testing, you will only step on the ramp once per trial. Between trials, the angle and the position of the ramp may be changed in order to test both your left and right foot. In addition, the ramp will at times be covered with a latex covering to keep it hidden from view. Force measurement devices in the walkway and ramp will record the forces from your walking patterns.

This procedure may be used with other procedures listed on the consent form. The testing will take up to 90 minutes.

☐ **Brief Clinical Exam of Feet and Ankles.** A VA clinician will conduct a brief examination of your feet and ankles. This will include observation and measurement with a tape measure/ruler and an angular measurement device called a “goniometer” while you stand barefoot and/or while you are seated or lying down. The clinician will also evaluate your range of motion, perform a manual assessment of your foot/ankle musculature, and observe the relative position of different parts of your foot and ankle to one another.

☐ **Standing Balance Procedure.** You will stand as still as possible with your eyes open and your arms at your sides for 40 seconds. Force measurement platforms embedded in the floor will measure the forces while you stand. We may ask you to do this up to eight times in your natural stance. To do this, we will ask you to walk to a position on the floor where you will stand and you can position your feet in whatever way makes you feel comfortable. We may also ask you to do this up to eight times with your feet in a set position. To do this, we will ask you to walk to a position on the floor where you will stand and place your feet in a specific position (for example, shoulder width apart or feet together).

☐ **Stairway Procedure.** You will be asked to walk up or down on short sets of stairs or a motorized stairway while we record how you walk. We may ask you to do this in short bouts of a minute or so, or up to 10 minutes at a time. If a new kind of propulsive prosthetic or orthotic device is being tested, you may be asked to wear a safety harness attached to the ceiling in case you lose your balance. This procedure may be used in conjunction with other procedures. The entire procedure may take up to 90 minutes.

This study, in itself, does not include treatment or medical advice. However, you are free to ask questions during the procedures and analysis results will be made available to you if requested.

3. Description of any procedures that may result in discomfort or inconvenience: During each procedure, you may feel inconvenienced in having to visit the lab.

Motion Measurement Using Optical Motion Detection System procedure. You may experience slight skin irritation from the removal of the tape used to attach the markers to your skin. You may also feel embarrassed wearing the supplied tight-fitting clothing.

Motion Measurement Using Accelerometry System procedure. You may feel uncomfortable or embarrassed wearing the devices in public.

Locomotion and Cognitive Function procedure. Since your attention may be on the task, there is an increased chance you could trip or fall during the procedure.

Metabolic Energy Measurement procedure. You may feel uncomfortable wearing a mask that feels warm against your face.

Telemetered Surface Electromyography procedure. You may have small patches of shaved skin which may cause some embarrassment. Electrodes within a prosthetic socket may cause a small cut. If a small cut does occur, the cut will be washed with antiseptic solution and bandaged. You may experience slight skin irritation from the shaving, abrading, or alcohol wipes used to clean your skin and/or from the removal of the tape.

Residual Limb Scanning procedure. You may feel unsteady while standing still.

Step Monitoring procedure. You may be inconvenienced in having to return the monitoring device.

Load Carriage procedure. You may feel unsteady walking with a weighted pack. Handrails, spotters, and/or a safety harness will be used during the testing to ensure your safety.

Cross-slope procedure. You may feel unsteady walking on an unbalanced walkway. Handrails, spotters, and/or a safety harness will be used during the testing to ensure your safety.

Standing Balance procedure. You may feel unsteady while standing still.

Stairway procedure. You may feel unsteady walking on a motorized stairway or escalator. Handrails and/or a safety harness will be used during testing to ensure your safety.

4. Potential risks of the study: Many of these procedures slightly increase your risk of falling or becoming slightly injured. Every effort will be made to monitor your gait to reduce these risks, including the use of a safety harness during specific procedures (such as testing a novel propulsive prosthetic foot). You should notify the researchers immediately if you feel at risk during any portion of the procedure.

Although we will make every effort to keep your information secret, no system for protecting information can be completely safe. It is still possible that someone could find out you were in this study and could find out information about you. Please refer to Section 7 for information as to how the researchers will keep your information confidential to the best of their abilities.

The particular treatments or procedures in this study may involve risks that are currently unforeseeable. We will contact you as soon as possible if new findings occur during this research that may pose a risk to you.

5. Potential benefits of study: There will be no direct benefit to you by being a part of this study. However, people with walking difficulties may benefit as a result of the information that we collect from this research study.

6. Other treatment available: This is not a treatment study. Your alternative to participating in this study is to not participate.

7. Use of research results / Confidentiality: The information obtained about you will be held confidential. However, for purposes of this study, the following list of people or groups may know that you are in this study. They will have access to your records, which may include your medical records:

- Research team members
- Federal agencies including, but not limited to, the Food and Drug Administration (FDA), the Office for Human Research Protection (OHRP), the VA Office of Research Oversight (ORO), the VA Office of the Inspector General (OIG), and the Government Accountability Office (GAO)
- The VA committees that oversee research, including the Institutional Review Board that oversees the safety and ethics of VA studies
- The VA Puget Sound Fiscal Department will be provided with your full name, address, phone number, and social security number in order to authorize payment for your participation in this study
- The UW committees that oversee research, including the UW Institutional Review Board (but they will not have access to your medical records)

The purpose of this access is to review the study and make sure that it meets all legal, compliance, and administrative requirements. If a review of this study takes place, your records may be examined. The reviewers will protect your privacy.

Your information including any audio, video, camera, all test results, and answers to the questions asked will be strictly confidential. All of the information you provide will be confidential. However, if we learn that you intend to harm yourself or others, we must report that to appropriate authorities. Otherwise, only the investigators will have access to the data collected.

If you agree to participate in our study, we will assign you a study code number. We will not include your name or other identifying information on any of the information that we collect from you. Only your study code number will be used.
All coded data will be stored on secured computers or in file cabinets in locked offices. This coded data will be kept indefinitely and may be used for comparison in future research studies.

If your voice is recorded, the audio files will be labeled with your unique study code number. The audio files will be stored on a secured computer accessible only to approved research staff. The digital recorders used to audio record interviews will be kept with the research staff or stored securely at all times.

Photos that are taken will have no identifiers and will be labeled with your study code number. The video will contain your image which will look like a stick figure. Your image will also be coded. The images from the camera and video will be stored on a password-protected computer.

Once this study is completed, we will not use the code linking you to your data for any additional research. The code linking you to your data will be stored on paper in a locked filing cabinet in a locked office until the VA receives authorization to destroy it in accordance with federal records regulations. It may be several years before the code linking you to your data is actually destroyed.

We have a database, called a repository, where we will store data from this study. The data will not include any information that could identify you, such as your name or social security number. We will use the data in the repository to answer new research questions in the future. We will ask you to sign a separate consent form to include your study data in the repository.

There may be publications about this study in the future. If so, your identity will be held confidential. No personal information will be given in a publication without your approval in writing.

Your study information will be used only for research purposes and will not be sold. Information gained from this research may be used commercially for the development of new ways to diagnose or treat diseases. However, neither you nor your family will gain financially from discoveries made using the information that you provide.

8. Special circumstances: The VA requires some Veterans to pay co-payments for medical care and services. You will still have to pay these co-payments as long as they are not related to this research study.

You will be paid $20 per hour for participating in these experiments. Payments of $50 or below will be paid in cash through the VA agent cashier. Payments over $50 will be paid in a one-time lump

payment by check from the Department of the Treasury (6–8 weeks) following your study participation. You may receive an Internal Revenue Service (IRS) Form 1099. If so, your social security number will be used for this purpose.

9. Withdrawal from the study: You do not have to take part in this study. If you are in this study, you can withdraw at any time. You will not be penalized for your decision to not participate or withdraw nor will you lose your VA or other benefits if you decide to do so. The study physician has the right to terminate your participation in this study if he or she feels that it is not in your best interest. This termination will not require your consent.
If you decide to withdraw, or if you are terminated from the study, a person from the study team will then need to meet with you to discuss the necessary steps that you may need to take to end your participation in the study.

10. Questions or concerns related to the study: The study researchers (listed below) *must* be contacted immediately if:

- You think you may have been harmed or injured as a direct result of this research; and/or
- You have any questions regarding your medical care issues.

During business hours (8:00 a.m. – 4:30 p.m.) Call Dr. Glenn Klute at (206) 277-6724.

If you are experiencing a medical emergency call 911 or go to the emergency room.

You may contact the Institutional Review Board (IRB) – VA Office at (206) 277-1715 if you:

- Would like to speak with a neutral party who is not involved with this study;
- Have questions, concerns, or complaints about the research;
- Would like to verify the validity of the study; or
- Have questions about your rights as a research subject.

An IRB is an independent body made up of medical, scientific, and non-scientific members, whose job it is to ensure the protection of the rights, safety, and well-being of human subjects involved in research.

11. Research-related injury: Medical treatment will be provided, if necessary, by the VA if you are injured by being in this study. You will not be charged for this treatment. Veterans who are injured because of being in this study may receive payment under Title 38, United States Code, Section 1151. Veterans or non-Veterans who are injured may receive payment under the Federal Tort Claims Act.

You do not waive any legal rights by signing this consent form.

12. Research subject's rights: I have read or have had read to me all of the above. The study has been explained to me, including a description of what the study is about and how and why it is being done. All of my questions have been answered. I have been told of the risks and/or discomforts, possible benefits of the study, and other choices of treatment available to me. My rights as a

Department of Veterans Affairs

CONSENT FOR PRODUCTION AND USE OF VERBAL OR WRITTEN STATEMENTS, PHOTOGRAPHS, DIGITAL IMAGES, AND/OR VIDEO OR AUDIO RECORDINGS BY VA

Name of individual whose statement, likeness, or voice is requested

NOTE: The execution of this form does not authorize production or use of materials except as specified below. The specified material may be produced and used by VA for authorized purposes identified below, such as education of VA personnel, research activities, or promotional efforts. It may also be disclosed outside VA as permitted by law and as noted below. If the material is part of a VA system of records, it may be disclosed outside VA as stated in the "Routine Uses" in the "VA Privacy Act Systems of Records" published in the Federal Register.

The purpose of this form is to document your consent to the Department of Veterans Affairs' (VA) request to obtain, produce, and/or use a verbal or written statement or a photograph, digital image, and/or video or audio recording containing your likeness or voice. By signing this form, you are authorizing the production or use only as specified below.

You are NOT REQUIRED TO CONSENT TO VA's REQUEST to obtain, produce, and/or use your statement, likeness, or voice. Your decision to consent or refuse will not affect your access to any present or future VA benefits for which you are eligible.

You may rescind your consent at any time prior to or during production of a photograph, digital image, or video or audio recording, or before or during your provision of a verbal or written statement. You may rescind your consent after production is complete if the burden on VA of complying with that request is not unreasonable considering the financial and administrative costs, the ease of compliance, and the number of parties involved.

The photograph, digital image, and/or video or audio recording will be produced while I am (describe the activity or situation) (***To Be Completed by the Department of Veteran Affairs, if applicable)***

Check at least one of the following (to be completed by VA)

☐ I hereby voluntarily and without compensation authorize _____
Name of Facility

to produce a photograph, digital image, and/or video or audio recording of me (or of the above named individual if the individual is legally unable to give consent).

☐ I hereby voluntarily and without compensation authorize _____
Name of Facility

to obtain or use a verbal or written statement from me (or the of the above named individual if the individual is legally unable to give consent).

Appendix F

EXPERIMENT 1 HISTOGRAMS

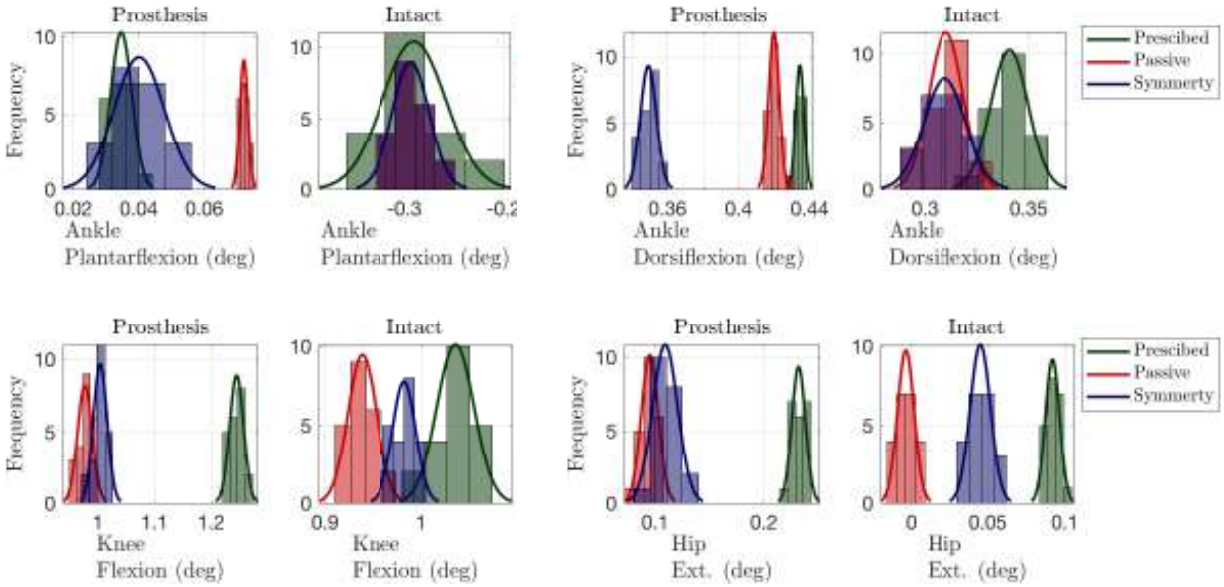


Figure F.1: Experiment 1 histograms for peak kinematic outcomes.

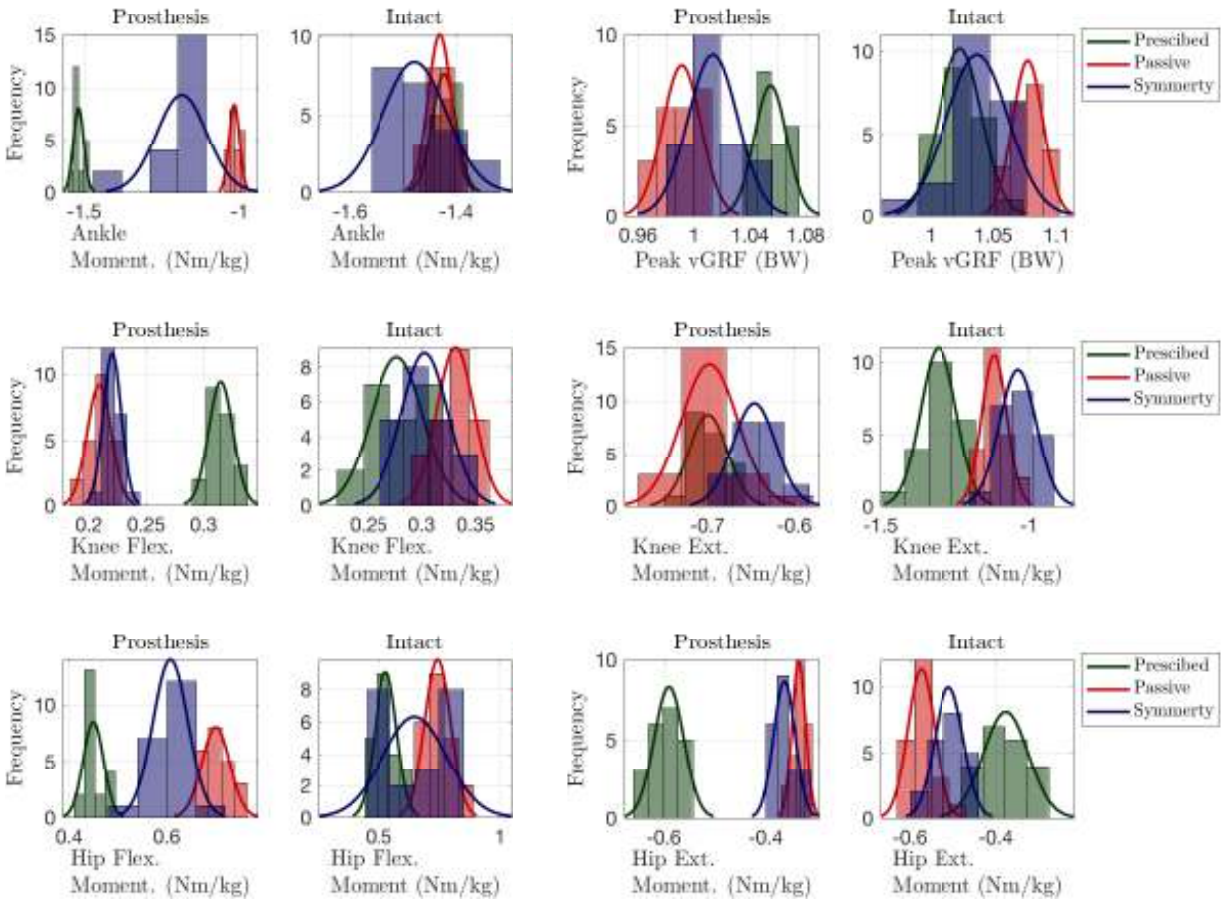


Figure F.2: Experiment 1 histograms for peak kinetic outcomes.

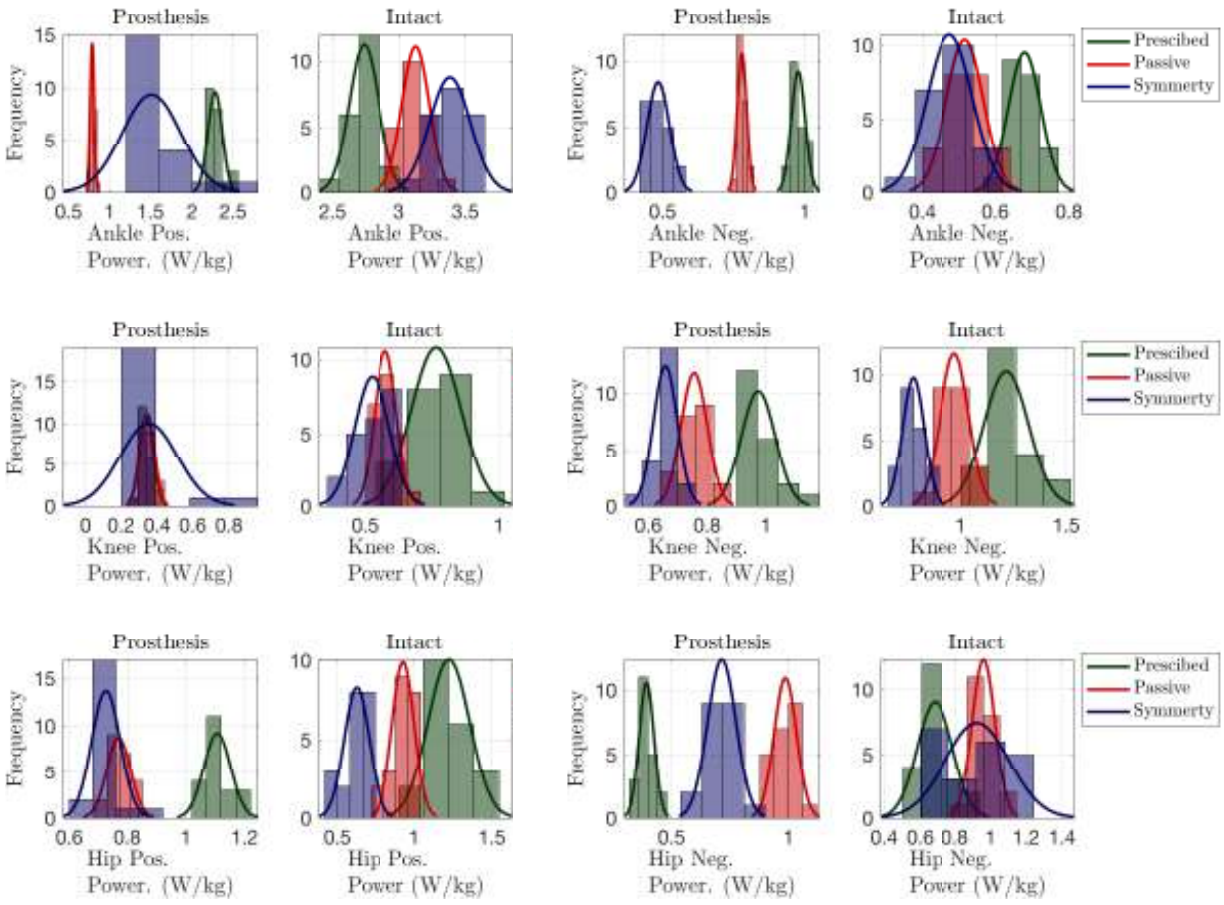


Figure F.3: Experiment 1 histograms for peak power outcomes.

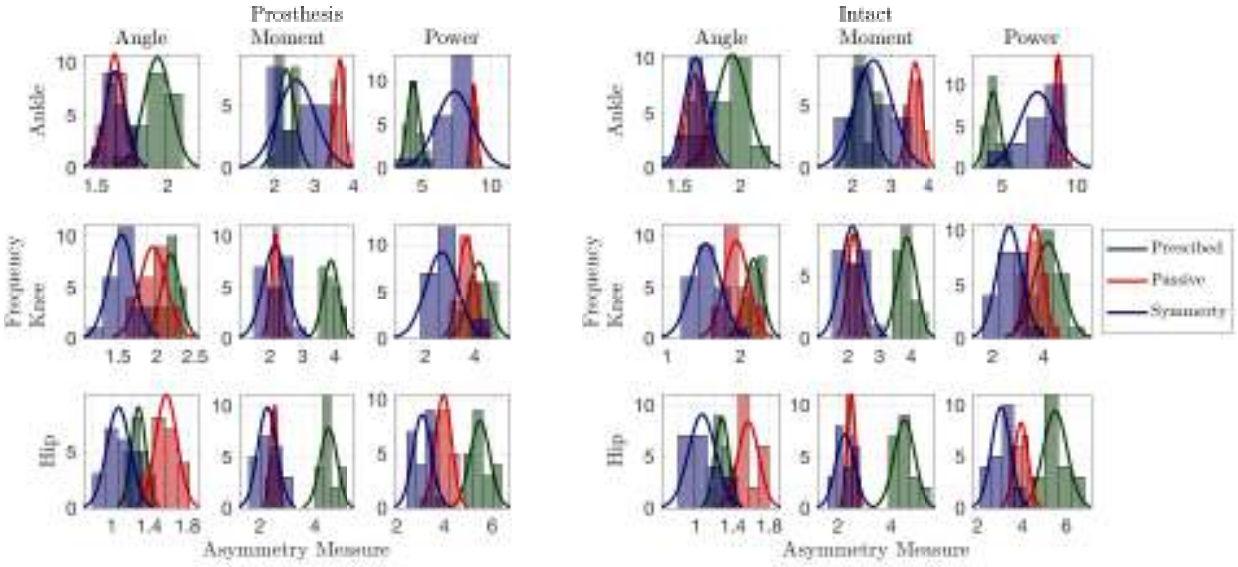


Figure F.4: Experiment 1 histograms for AM outcomes.

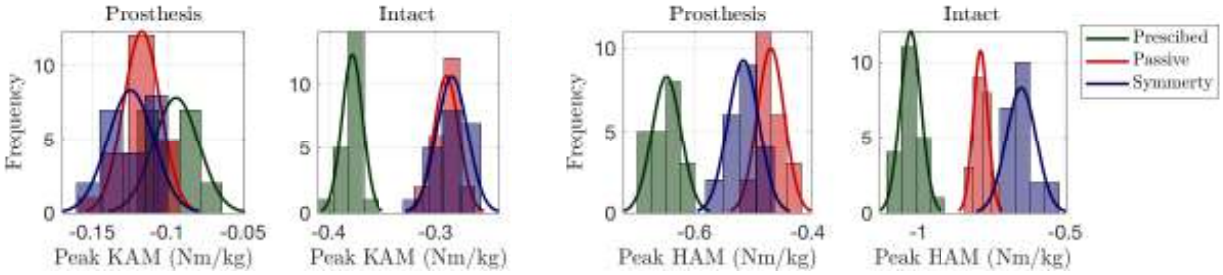


Figure F.5: Experiment 1 histograms for KAM and HAM outcomes.

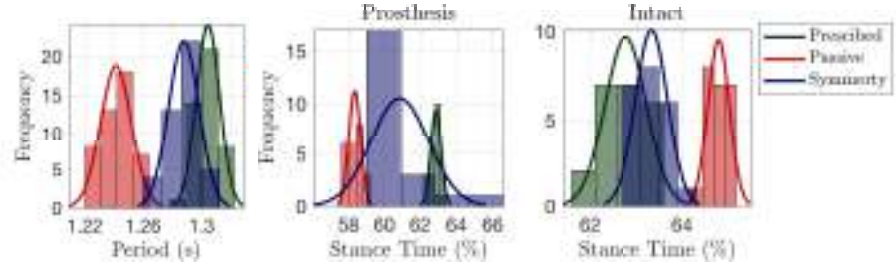


Figure F.6: Experiment 1 histograms for temporal outcomes.

Appendix G

EXPERIMENT 2 RESULTS

G.1 Summary

Results for experimental 2 are provided below. The motor current saturated at learning iteration $k = 4$, therefore outcome measures were computed with iteration $k = 3$, which show similar results as presented in Ch. 6. Note that in this case a heel strike delay, Δ_{HS}, was not used, since time-normalization was accomplished with embedded data, e.g., insoles. In addition, the allowable growth factor α was added to the algorithm at a later date, thus α was not used.

Table G.1: Algorithm and motor parameters for experiment 2.

Gearbox transmission	R_G	5.8
Effective transmission	R_T	471
Torque constant (Nm/A)	k_τ	0.0053
Speed constant (rpm/V)	k_s	1730
Initial gain	ρ	0.5
Pre-smooth time (%)	t_1	10
Post-smooth time (%)	t_2	90
Max harmonic (rad/s)	ω_{max}	$10 \cdot \omega_0$
Filter cutoff	ω_c	$20 \cdot \omega_0$
HS delay (ms)	Δ_{HS}	NA
Decay rate	γ	2
Allowable reference growth factor	α	NA

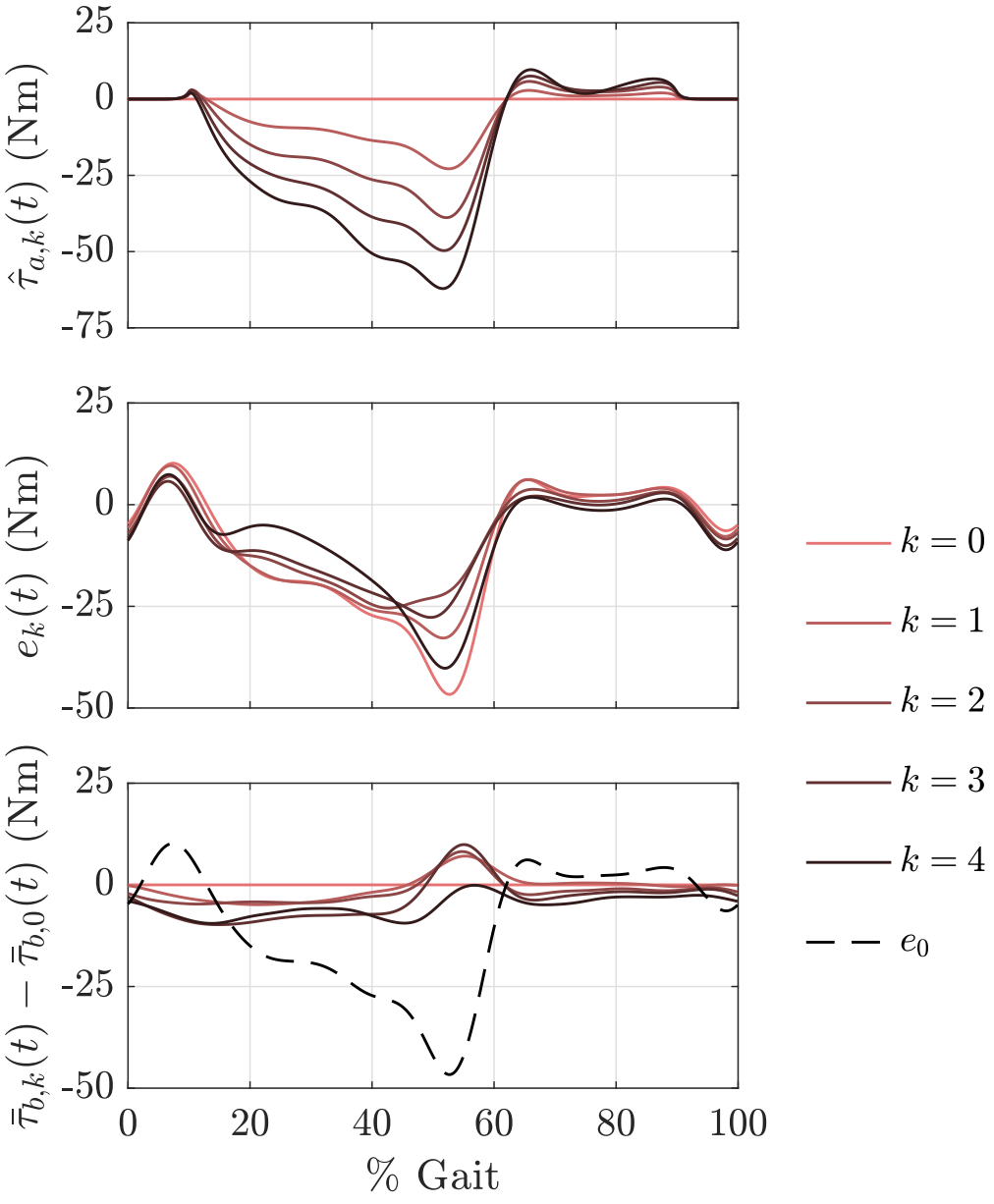


Figure G.1: Experiment 2 time domain evolution of active torque $\hat{\tau}_{a,k}$, error e_k, and the change in reference signal, e.g., $\bar{\tau}_{b,k} - \bar{\tau}_{b,0}$.

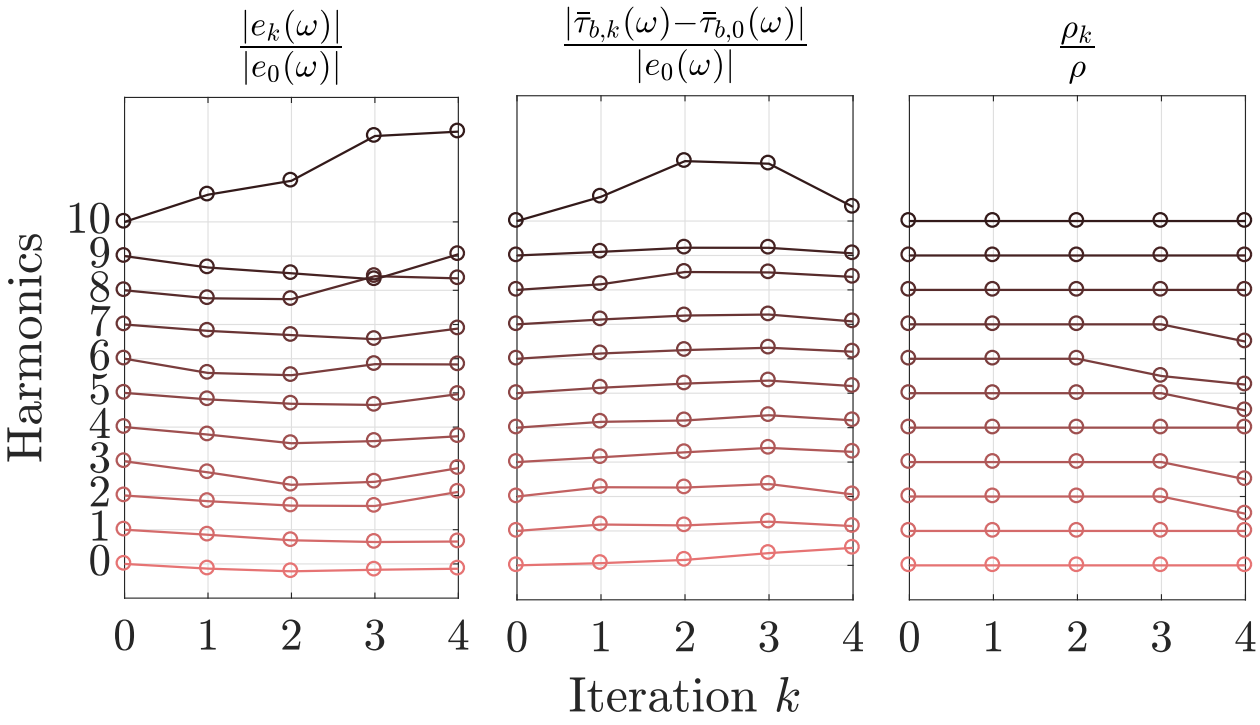


Figure G.2: Experiment 2 frequency domain evolution of the magnitude of error, normalized by the initial error, $|e_k|/|e_0|$, the magnitude of change in reference signal, normalized by the initial error $|\bar{\tau}_{b,k} - \bar{\tau}_{b,0}|/|e_0|$, and adaptation in ρ_k, normalized by the initial ρ_0.

Table G.2: Experiment 2 joint kinematic outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Plantarflexion (deg)	1.99(0.19)^{a}	−16.66(1.85)	−3.14(0.17)^{ab}	−25.17(1.21)^{b}	−6.54(1.50)^{abc}	−27.06(1.01)^{bc}
Peak Ankle Dorsiflexion (deg)	24.90(0.13)^{a}	19.52(0.51)	23.82(0.21)^{ab}	17.49(0.62)^{b}	19.99(0.30)^{abc}	16.89(0.66)^{bc}
Peak Knee Flexion (deg)	71.34(0.70)^{a}	59.30(1.10)	56.40(0.69)^{ab}	44.10(0.98)^{b}	58.10(0.93)^{abc}	43.38(0.53)^{bc}
Peak Hip Extension (deg)	13.29(0.36)^{a}	5.28(0.26)	4.20(0.45)^{ab}	−3.42(0.37)^{b}	4.94(0.58)^{abc}	−3.19(0.33)^{bc}

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

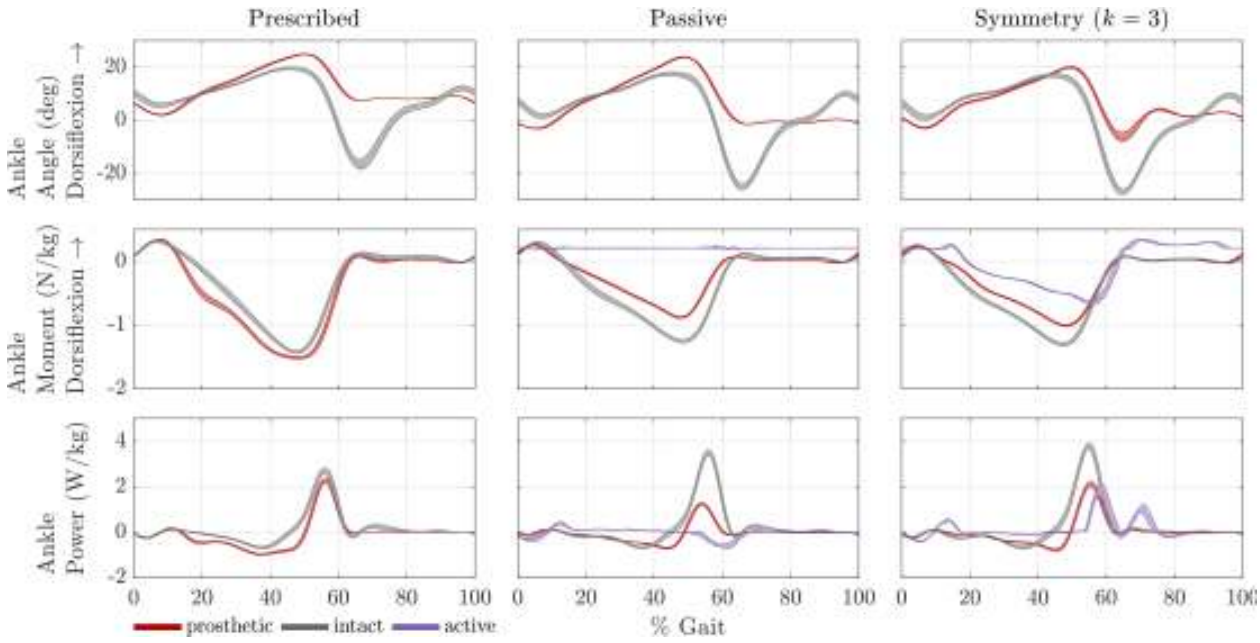


Figure G.3: Experiment 2 sagittal ankle mechanics for the three experimental conditions.

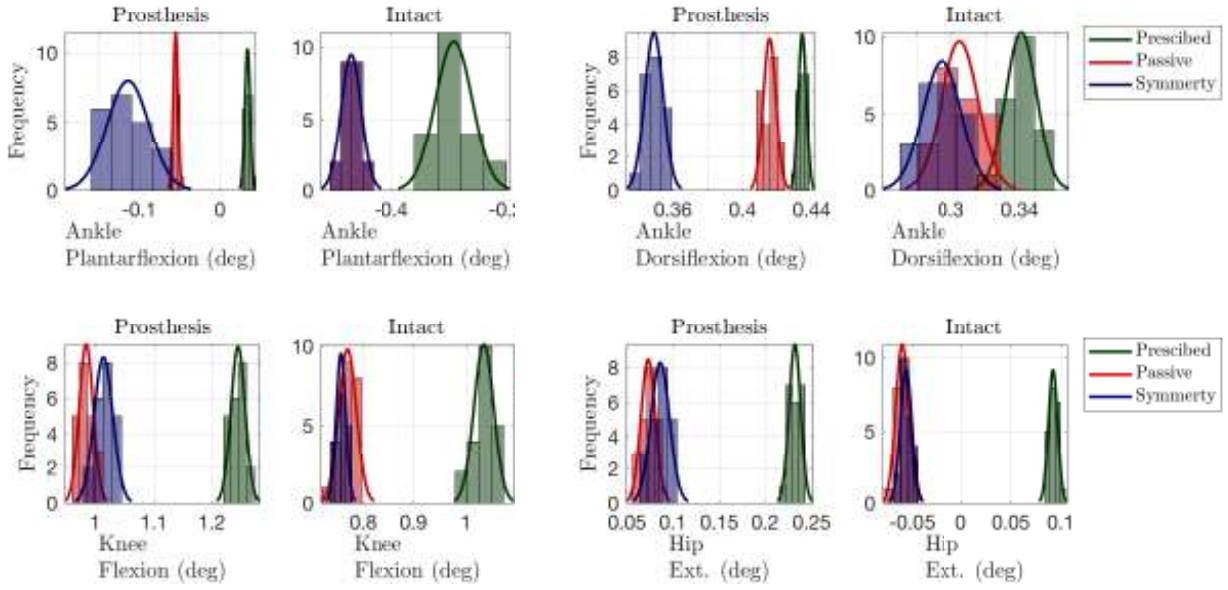


Figure G.4: Experiment 2 histograms for peak kinematic outcomes.

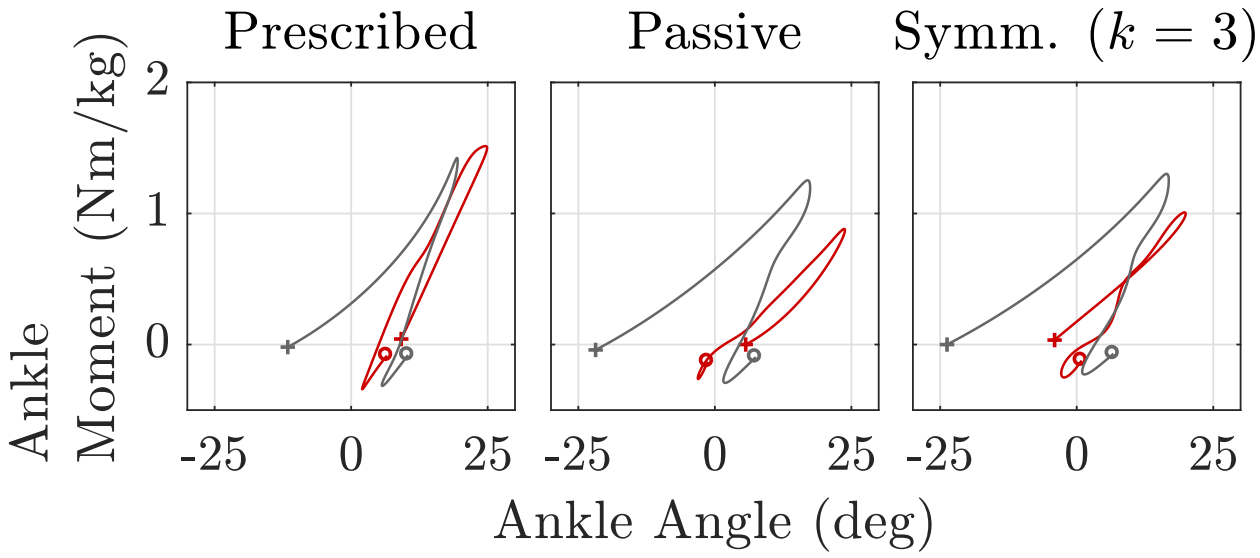


Figure G.5: Experiment 2 ankle work loops for the three experimental conditions.

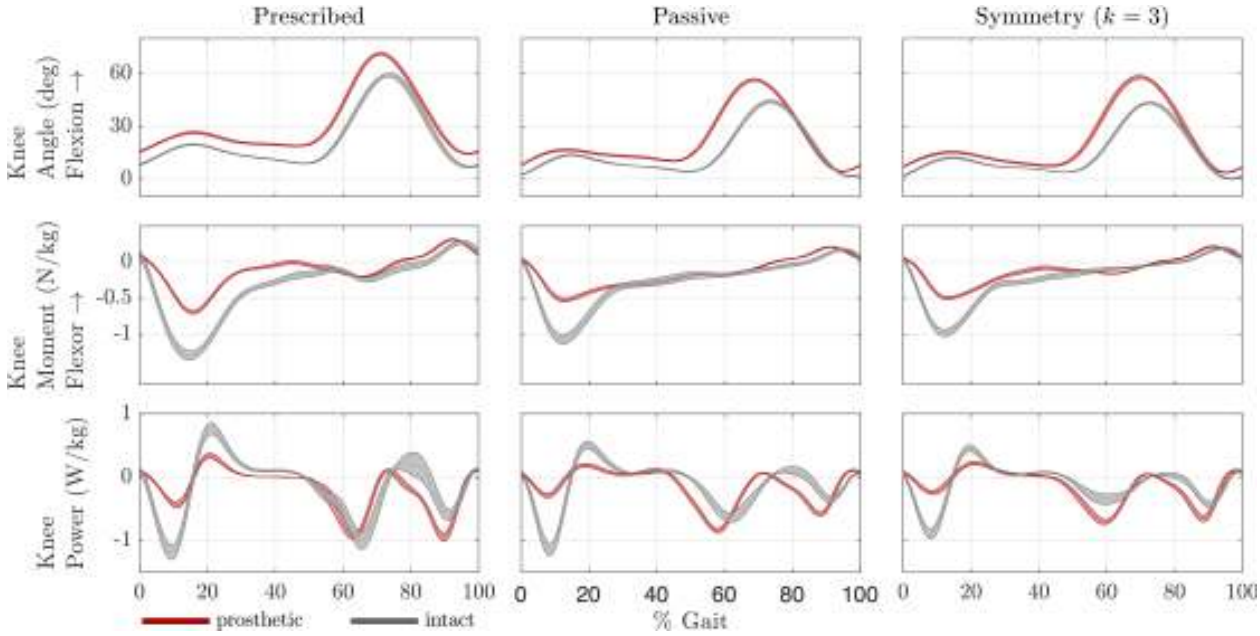


Figure G.6: Experiment 2 sagittal plane knee mechanics for the three experimental conditions.

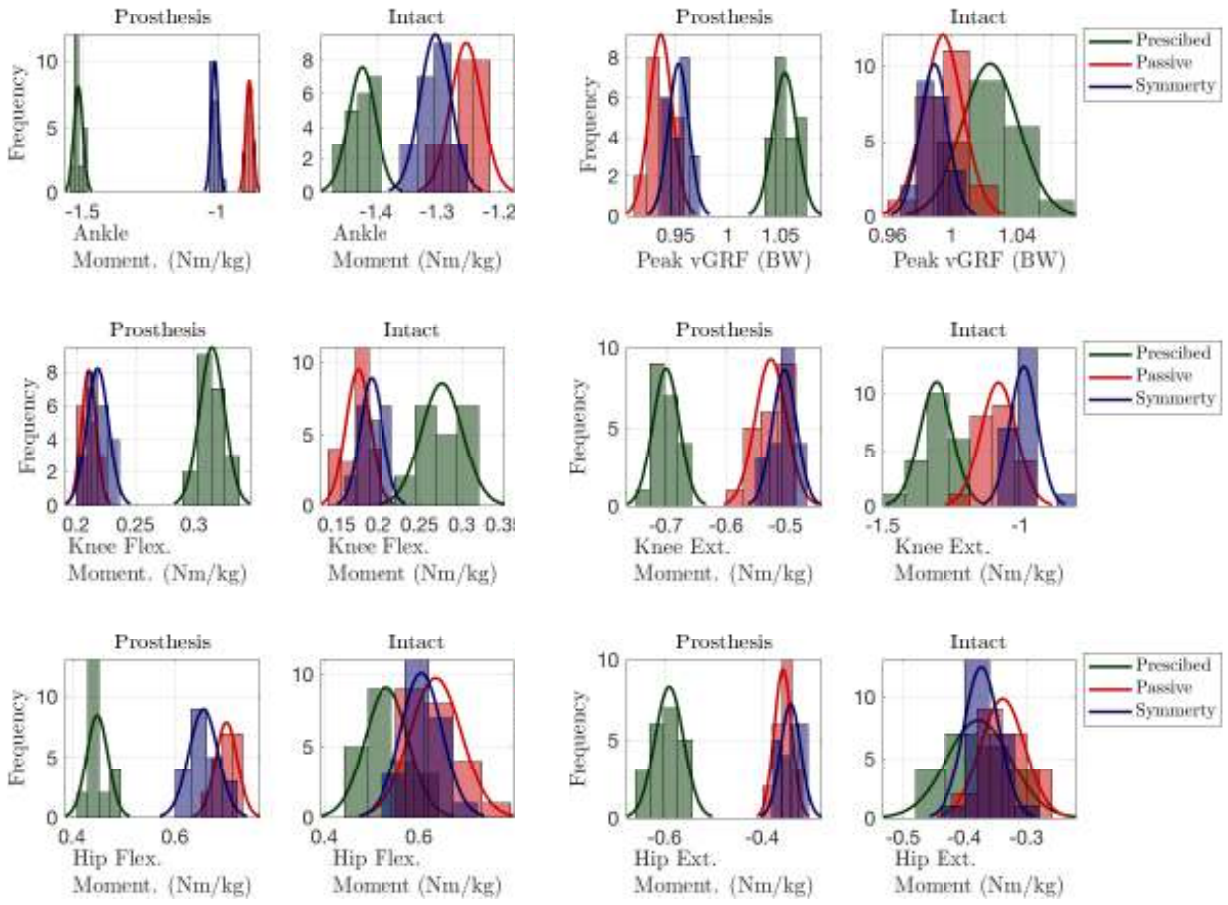


Figure G.7: Experiment 2 histograms for peak kinetic outcomes.

Table G.3: Experiment 2 joint kinetics and vGRF outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Moment (Nm/kg)	$-1.52(0.02)^{a}$	$-1.42(0.02)$	$-0.88(0.01)^{ab}$	$-1.25(0.03)^{b}$	$-1.01(0.01)^{abc}$	$-1.30(0.02)^{bc}$
Peak Knee Flexion Moment (Nm/kg)	$0.32(0.01)^{a}$	$0.28(0.02)$	$0.21(0.01)^{ab}$	$0.18(0.01)^{b}$	$0.22(0.01)^{abc}$	$0.19(0.01)^{bc}$
Peak Knee Extension Moment (Nm/kg)	$-0.70(0.02)^{a}$	$-1.31(0.06)$	$-0.52(0.03)^{ab}$	$-1.08(0.06)^{b}$	$-0.50(0.02)^{abc}$	$-0.99(0.05)^{bc}$
Peak Hip Flexion Moment (Nm/kg)	$0.45(0.02)^{a}$	$0.53(0.05)$	$0.70(0.02)^{ab}$	$0.63(0.05)^{b}$	$0.65(0.03)^{abc}$	$0.60(0.04)^{bc}$
Peak Hip Extension Moment (Nm/kg)	$-0.59(0.03)^{a}$	$-0.38(0.05)$	$-0.36(0.02)^{ab}$	$-0.34(0.04)^{b}$	$-0.34(0.02)^{abc}$	$-0.37(0.03)^{c}$
Peak vGRF (BW)	$1.05(0.01)^{a}$	$1.02(0.02)$	$0.94(0.01)^{ab}$	$0.99(0.01)^{b}$	$0.95(0.01)^{abc}$	$0.99(0.01)^{b}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

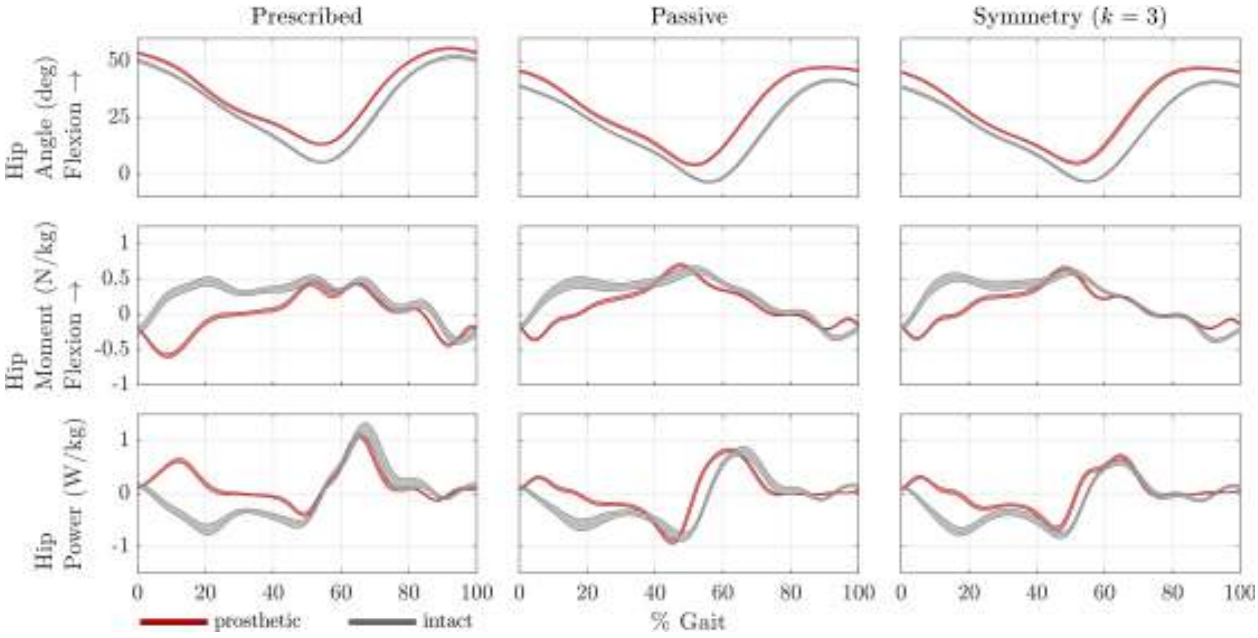


Figure G.8: Experiment 2 sagittal plane hip mechanics for the three experimental conditions.

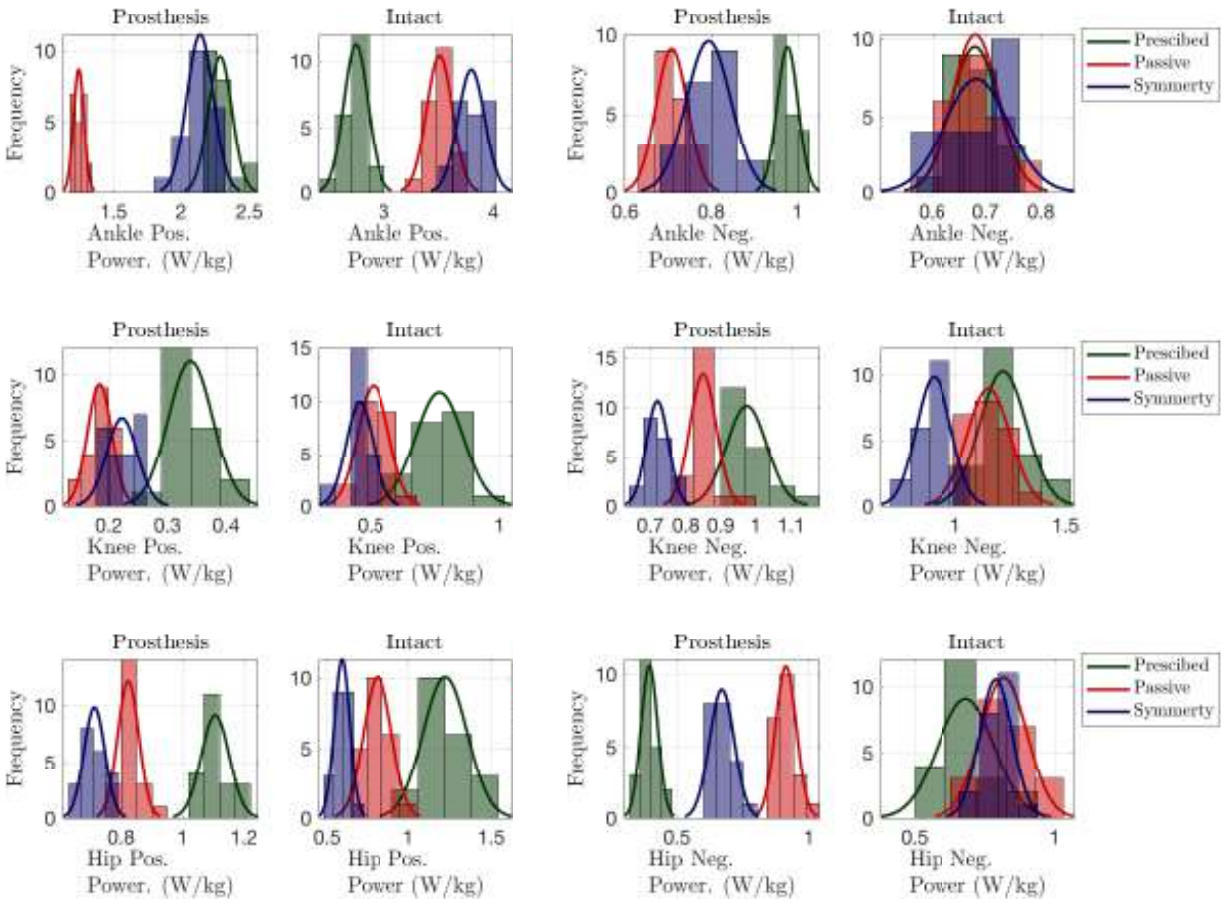


Figure G.9: Experiment 2 histograms for peak power outcomes.

Table G.4: Experiment 2 joint power outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Positive Ankle Power (W/kg)	$2.29(0.09)^{a}$	2.74(0.11)	$1.25(0.04)^{ab}$	$3.51(0.12)^{b}$	$2.14(0.10)^{abc}$	$3.80(0.12)^{bc}$
Peak Negative Ankle Power (W/kg)	$0.98(0.02)^{a}$	0.68(0.04)	$0.71(0.04)^{ab}$	0.68(0.04)	$0.79(0.05)^{abc}$	0.68(0.06)
Peak Positive Knee Power (W/kg)	$0.34(0.04)^{a}$	0.77(0.09)	$0.18(0.02)^{ab}$	$0.51(0.06)^{b}$	$0.22(0.03)^{abc}$	$0.46(0.05)^{bc}$
Peak Negative Knee Power (W/kg)	$0.97(0.06)^{a}$	1.22(0.11)	$0.85(0.04)^{ab}$	$1.15(0.10)^{b}$	$0.72(0.03)^{abc}$	$0.90(0.08)^{bc}$
Peak Positive Hip Power (W/kg)	$1.11(0.05)^{a}$	1.22(0.13)	$0.82(0.03)^{b}$	$0.81(0.09)^{b}$	$0.71(0.03)^{abc}$	$0.59(0.05)^{bc}$
Peak Negative Hip Power (W/kg)	$0.39(0.03)^{a}$	0.69(0.10)	$0.91(0.04)^{ab}$	$0.82(0.08)^{b}$	$0.67(0.05)^{abc}$	$0.79(0.06)^{b}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

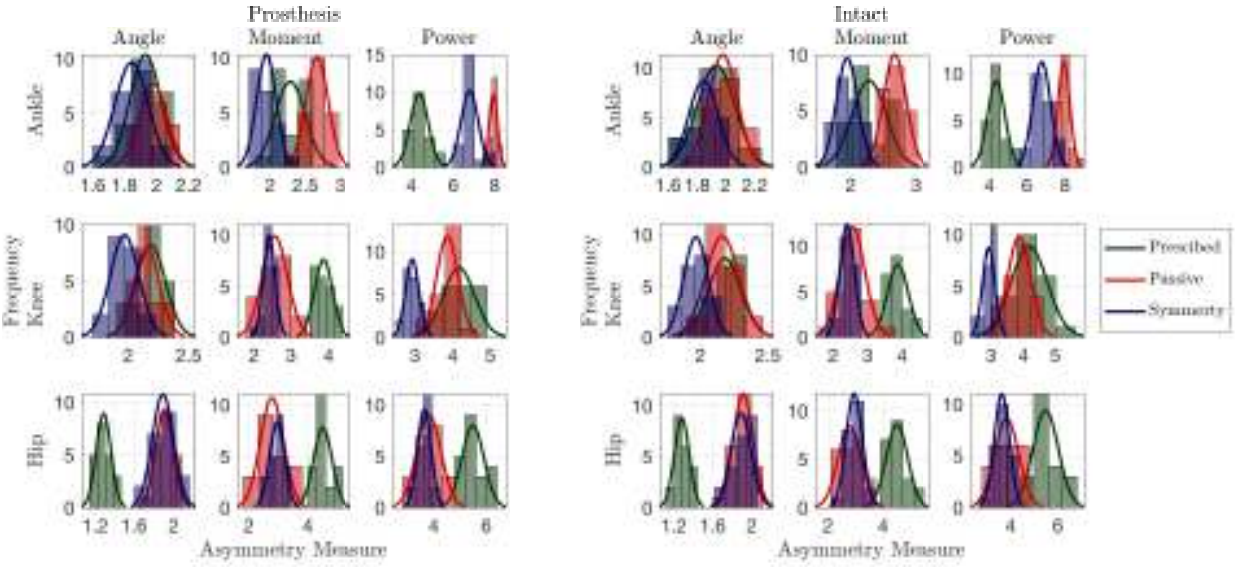


Figure G.10: Experiment 2 histograms for AM outcomes.

Table G.5: Experiment 2 sagittal plane AM outcome measures (standard deviations), defined in (6.2).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
AM GRF (BW)	2.42(0.37)	2.43(0.38)	4.26(0.26)^{b}	4.39(0.40)^{b}	2.03(0.24)^{bc}	2.04(0.24)^{bc}
AM Ankle Angle (rad)	1.93(0.10)	1.93(0.13)	1.98(0.09)	1.98(0.10)	1.84(0.10)^{bc}	1.85(0.10)^{bc}
AM Ankle Moment (Nm/kg)	2.29(0.21)	2.28(0.26)	2.66(0.16)^{b}	2.67(0.18)^{b}	1.95(0.14)^{bc}	1.95(0.16)^{bc}
AM Ankle Power (W/kg)	4.38(0.43)	4.39(0.46)	7.92(0.20)^{b}	7.95(0.31)^{b}	6.79(0.41)^{bc}	6.79(0.47)^{bc}
AM Knee Angle (rad)	2.19(0.12)	2.18(0.12)	2.15(0.11)	2.16(0.12)	1.97(0.12)^{bc}	1.97(0.09)^{bc}
AM Knee Moment (Nm/kg)	3.88(0.24)	3.88(0.30)	2.57(0.34)^{b}	2.56(0.36)^{b}	2.41(0.16)^{b}	2.41(0.21)^{b}
AM Knee Power (W/kg)	4.16(0.41)	4.18(0.57)	3.86(0.29)^{b}	3.87(0.36)^{b}	2.93(0.18)^{bc}	2.92(0.19)^{bc}
AM Hip Angle (rad)	1.29(0.08)	1.28(0.08)	1.91(0.09)^{b}	1.92(0.10)^{b}	1.89(0.11)^{b}	1.89(0.10)^{b}
AM Hip Moment (Nm/kg)	4.50(0.31)	4.50(0.39)	2.78(0.39)^{b}	2.78(0.42)^{b}	2.95(0.25)^{b}	2.95(0.30)^{b}
AM Hip Power (W/kg)	5.50(0.42)	5.49(0.54)	3.81(0.46)^{b}	3.84(0.51)^{b}	3.64(0.26)^{b}	3.64(0.32)^{b}

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

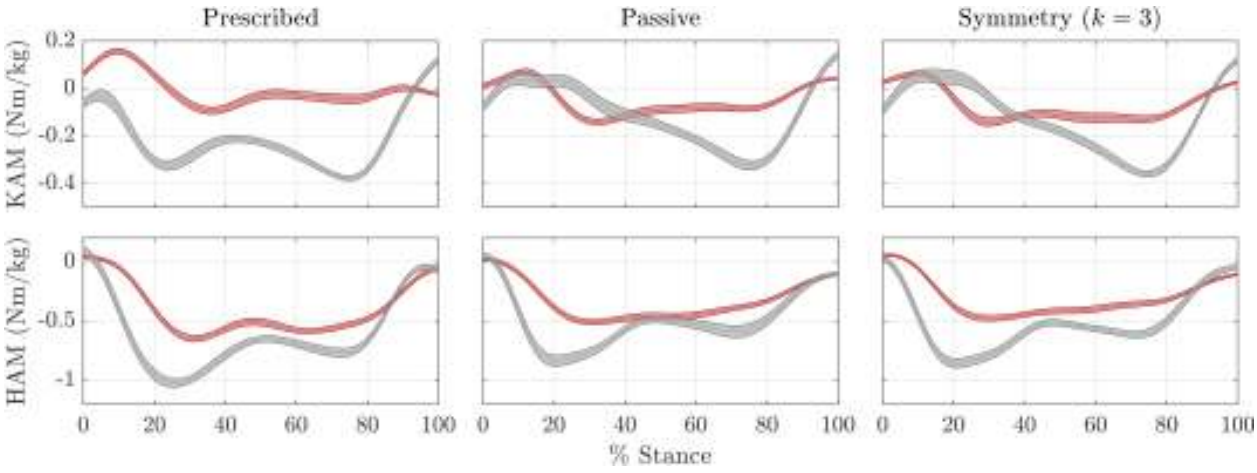


Figure G.11: Experiment 2 frontal plane hip moments (KAM and HAM) for the three experimental conditions.

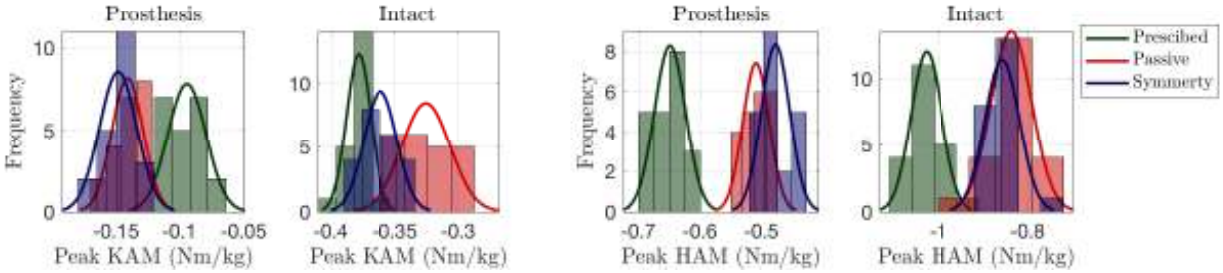


Figure G.12: Experiment 2 histograms for KAM and HAM outcomes.

Table G.6: Experiment 2 OA loading factors outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Knee Adduction Moment (Nm/kg)	$-0.09(0.02)^{a}$	$-0.38(0.01)$	$-0.14(0.01)^{ab}$	$-0.33(0.02)^{b}$	$-0.15(0.01)^{ab}$	$-0.36(0.01)^{bc}$
Peak Hip Adduction Moment (Nm/kg)	$-0.65(0.03)^{a}$	$-1.03(0.03)$	$-0.51(0.02)^{ab}$	$-0.84(0.05)^{b}$	$-0.48(0.02)^{abc}$	$-0.86(0.04)^{b}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

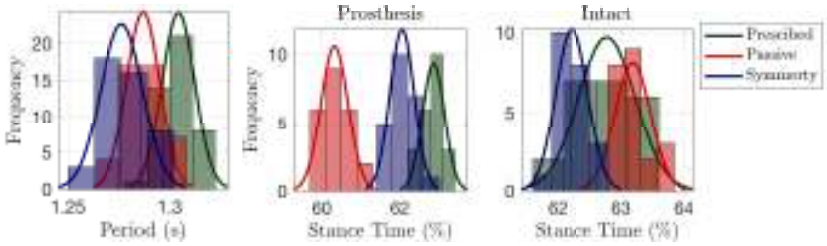


Figure G.13: Experiment 2 histograms for temporal outcomes.

Table G.7: Experiment 2 Temporal symmetry outcomes measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Step Period (s)	1.30(0.01)		1.29(0.01)		1.28(0.01)	
Stance	62.87(0.28)	62.78(0.45)	60.34(0.35)$^{a\,b}$	63.19(0.28)^{b}	62.06(0.31)$^{b\,c}$	62.22(0.26)$^{b\,c}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

Appendix H

EXPERIMENT 3 RESULTS

H.1 Summary

Results for experimental 3 are provided below. The results from this experiment demonstrate how a large ρ can cause issues. However, many of the same outcomes reported in Ch. 6 were also found to be true. One thing to note is an increase in intact negative knee power late in the during swing (see Fig. H.6).

Table H.1: Algorithm and motor parameters for experiment 3.

Gearbox transmission	R_G	19.2
Effective transmission	R_T	1800
Torque constant (Nm/A)	k_τ	0.00775
Speed constant (rpm/V)	k_s	1223
Initial gain	ρ	1
Pre-smooth time (%)	t_1	5
Post-smooth time (%)	t_2	$TO + 20\%$
Max harmonic (rad/s)	ω_{max}	$10 \cdot \omega_0$
Filter cutoff	ω_c	$10 \cdot \omega_0$
HS delay (ms)	Δ_{HS}	40
Decay rate	γ	2
Allowable reference growth factor	α	2

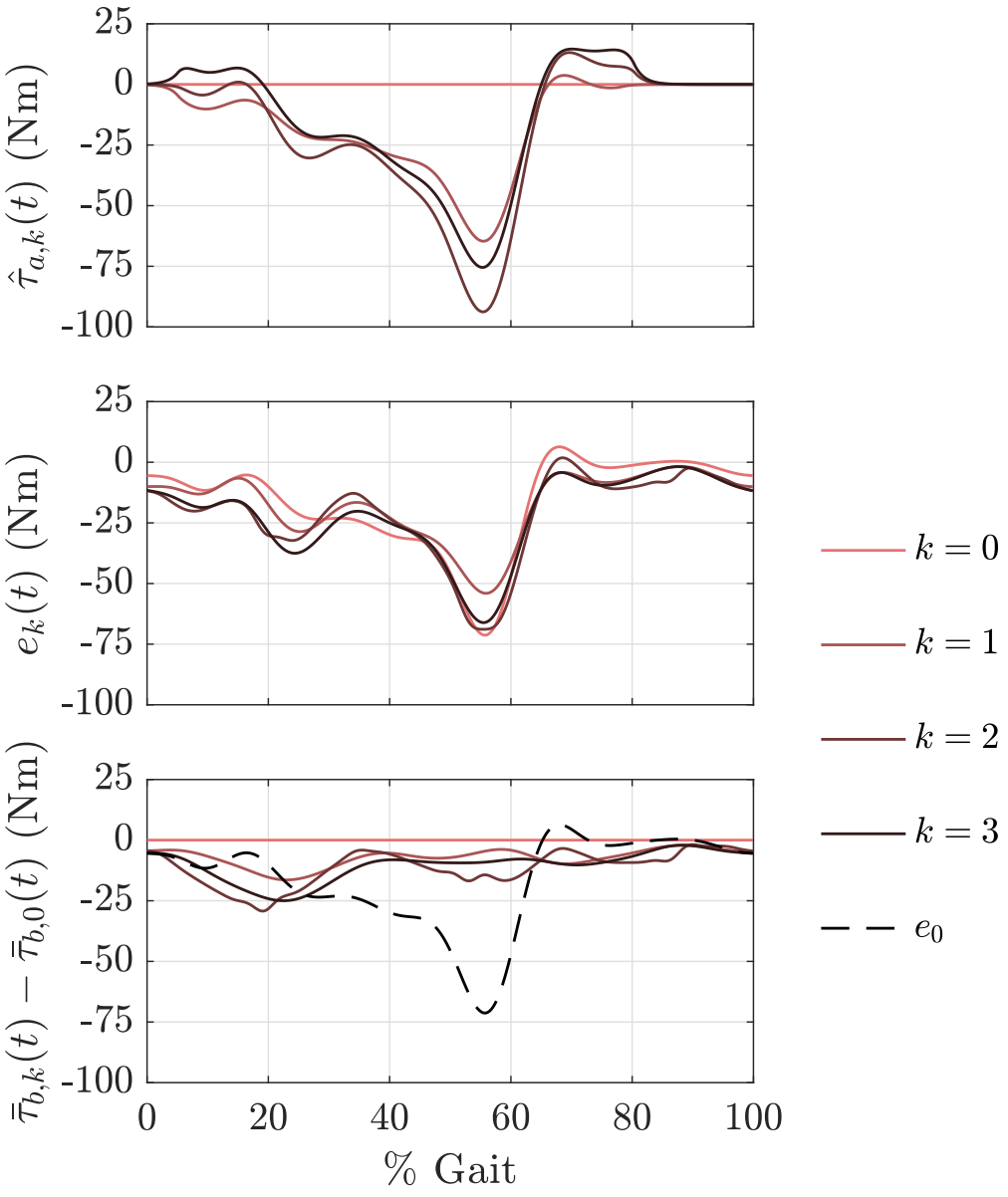


Figure H.1: Experiment 3 time domain evolution of active torque $\hat{\tau}_{a,k}$, error e_k, and the change in reference signal, e.g., $\bar{\tau}_{b,k} - \bar{\tau}_{b,0}$.

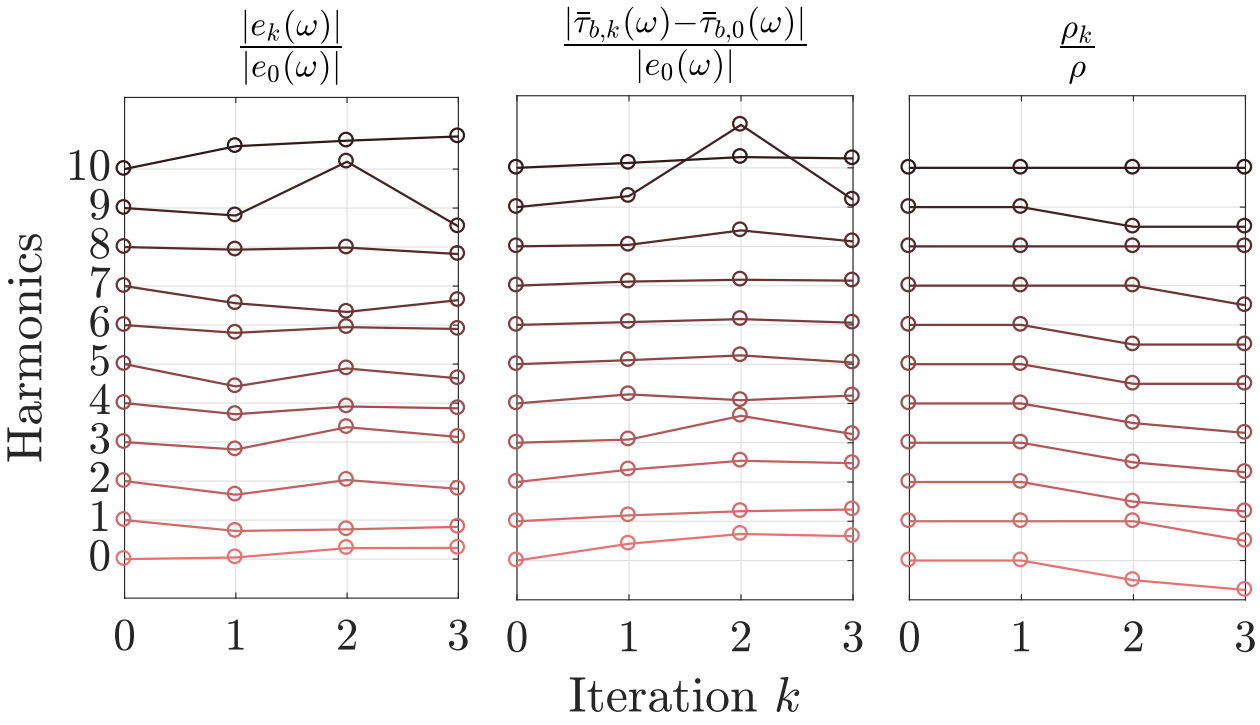


Figure H.2: Experiment 3 frequency domain evolution of the magnitude of error, normalized by the initial error, $|e_k|/|e_0|$, the magnitude of change in reference signal, normalized by the initial error $|\bar{\tau}_{b,k} - \bar{\tau}_{b,0}|/|e_0|$, and adaptation in ρ_k, normalized by the initial ρ_0.

Table H.2: Experiment 3 joint kinematic outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Plantarflexion (deg)	$1.99(0.19)^{a}$	$-16.66(1.85)$	$4.13(0.07)^{ab}$	$-14.73(0.93)^{b}$	$-0.99(0.59)^{abc}$	$-16.84(1.19)^{c}$
Peak Ankle Dorsiflexion (deg)	$24.90(0.13)^{a}$	$19.52(0.51)$	$24.06(0.17)^{ab}$	$17.77(0.48)^{b}$	$21.19(0.20)^{abc}$	$17.34(0.59)^{bc}$
Peak Knee Flexion (deg)	$71.34(0.70)^{a}$	$59.30(1.10)$	$56.00(0.74)^{ab}$	$53.77(0.85)^{b}$	$59.14(0.45)^{abc}$	$55.73(0.66)^{bc}$
Peak Hip Extension (deg)	$13.29(0.36)^{a}$	$5.28(0.26)$	$5.46(0.45)^{ab}$	$-0.22(0.31)^{b}$	$3.11(0.34)^{abc}$	$-2.61(0.38)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

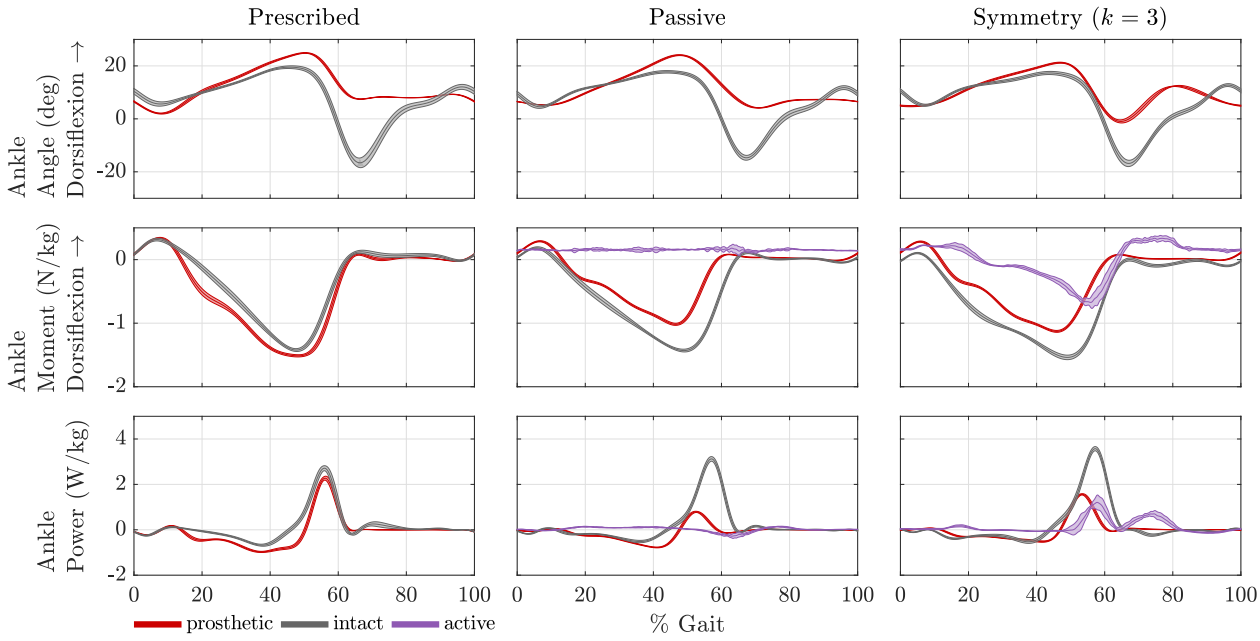


Figure H.3: Experiment 3 sagittal ankle mechanics for the three experimental conditions.

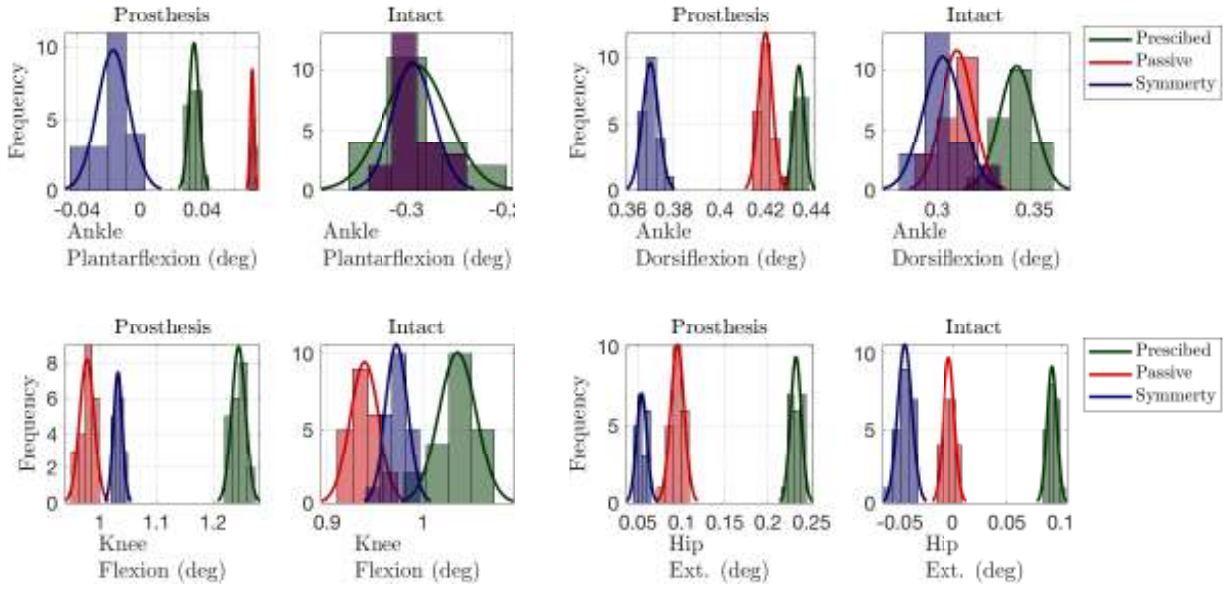


Figure H.4: Experiment 3 histograms for peak kinematic outcomes.

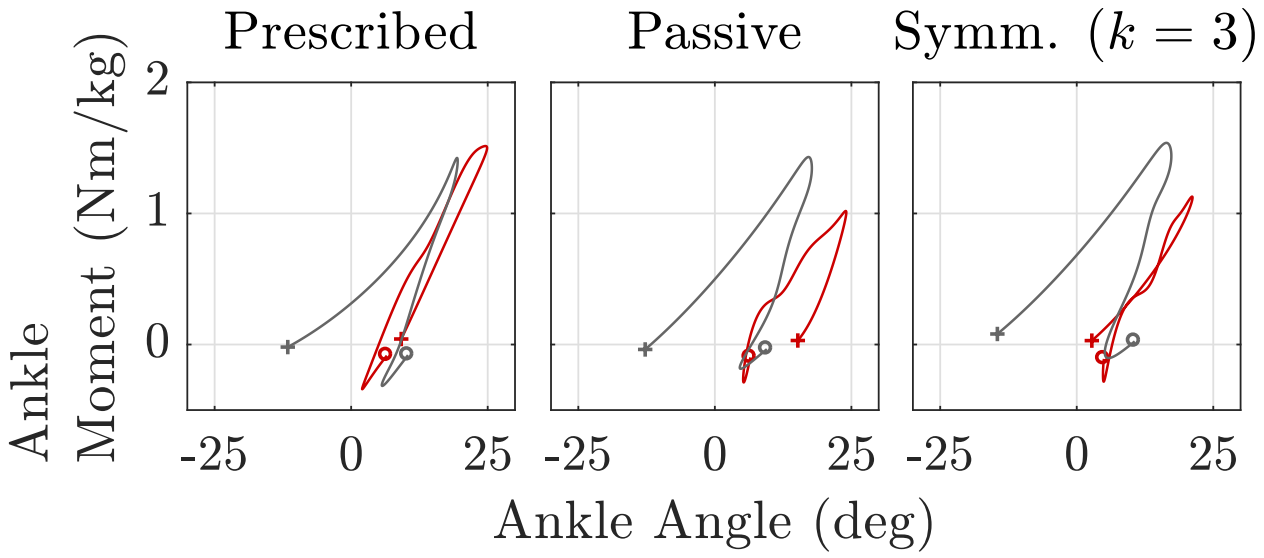


Figure H.5: Experiment 3 ankle work loops for the three experimental conditions.

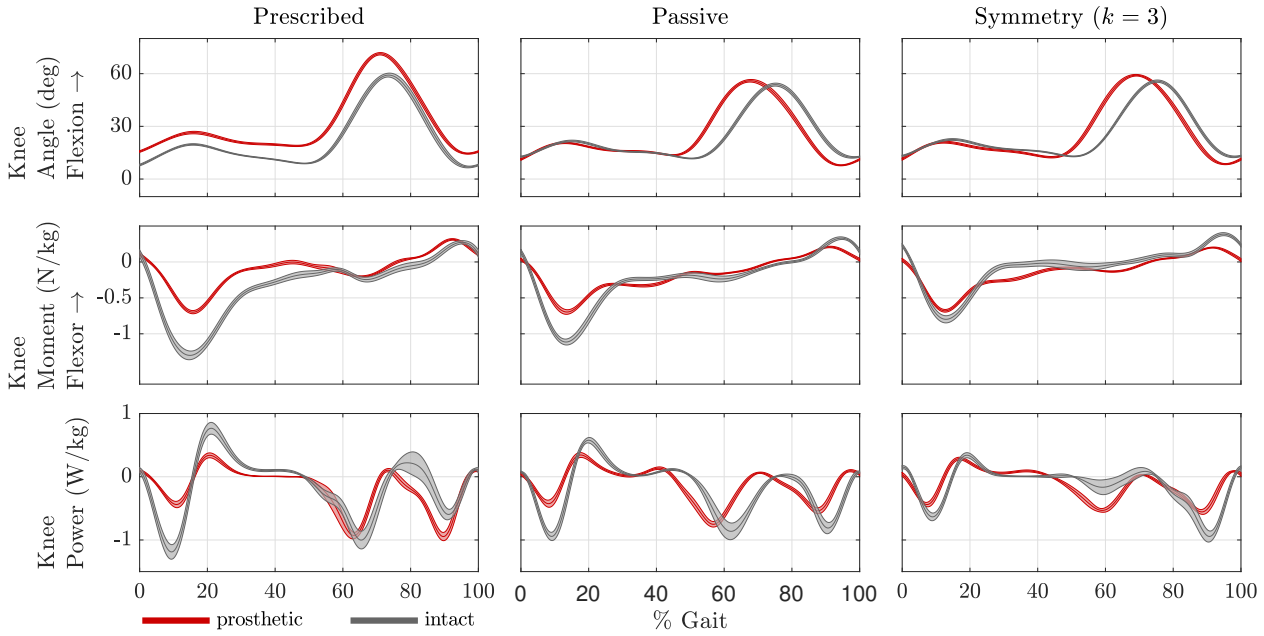


Figure H.6: Experiment 3 sagittal plane knee mechanics for the three experimental conditions.

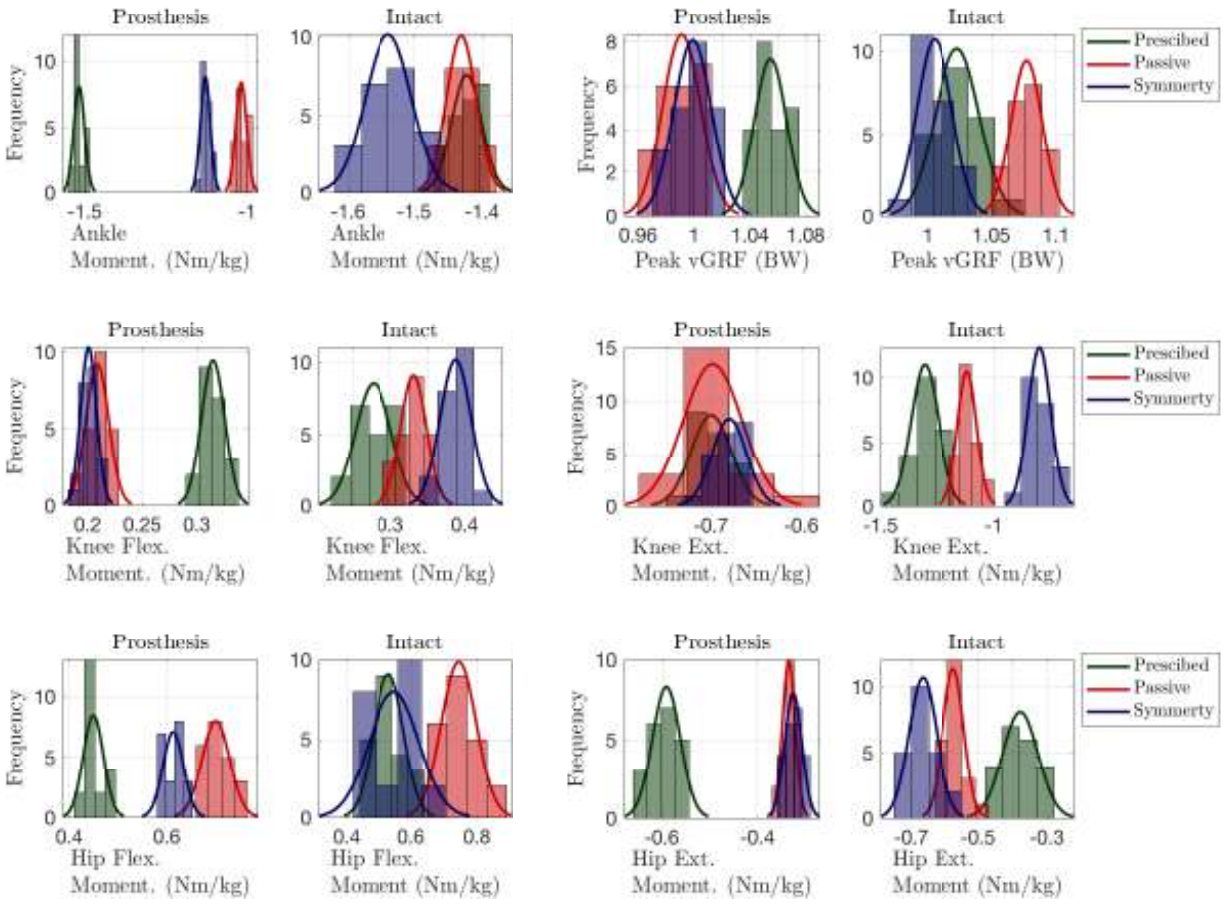


Figure H.7: Experiment 3 histograms for peak kinetic outcomes.

Table H.3: Experiment 3 joint kinetics and vGRF outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Moment (Nm/kg)	$-1.52(0.02)^{a}$	$-1.42(0.02)$	$-1.02(0.02)^{ab}$	$-1.43(0.02)$	$-1.13(0.02)^{abc}$	$-1.54(0.03)^{bc}$
Peak Knee Flexion Moment (Nm/kg)	$0.32(0.01)^{a}$	$0.28(0.02)$	$0.21(0.01)^{ab}$	$0.33(0.02)^{b}$	$0.20(0.01)^{abc}$	$0.39(0.02)^{bc}$
Peak Knee Extension Moment (Nm/kg)	$-0.70(0.02)^{a}$	$-1.31(0.06)$	$-0.70(0.03)^{a}$	$-1.12(0.04)^{b}$	$-0.68(0.02)^{abc}$	$-0.80(0.05)^{bc}$
Peak Hip Flexion Moment (Nm/kg)	$0.45(0.02)^{a}$	$0.53(0.05)$	$0.70(0.03)^{ab}$	$0.75(0.05)^{b}$	$0.61(0.02)^{abc}$	$0.55(0.08)^{c}$
Peak Hip Extension Moment (Nm/kg)	$-0.59(0.03)^{a}$	$-0.38(0.05)$	$-0.33(0.01)^{ab}$	$-0.57(0.03)^{b}$	$-0.33(0.02)^{ab}$	$-0.66(0.04)^{bc}$
Peak vGRF (BW)	$1.05(0.01)^{a}$	$1.02(0.02)$	$0.99(0.01)^{ab}$	$1.08(0.01)^{b}$	$1.00(0.01)^{b}$	$1.01(0.01)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

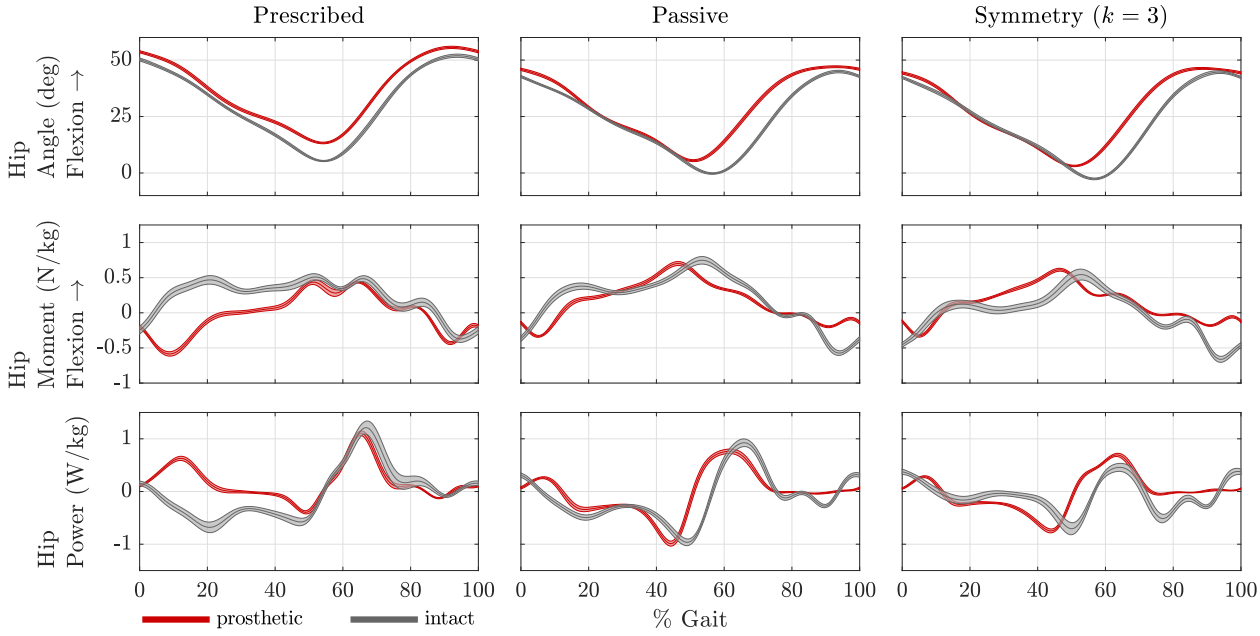


Figure H.8: Experiment 3 sagittal plane hip mechanics for the three experimental conditions.

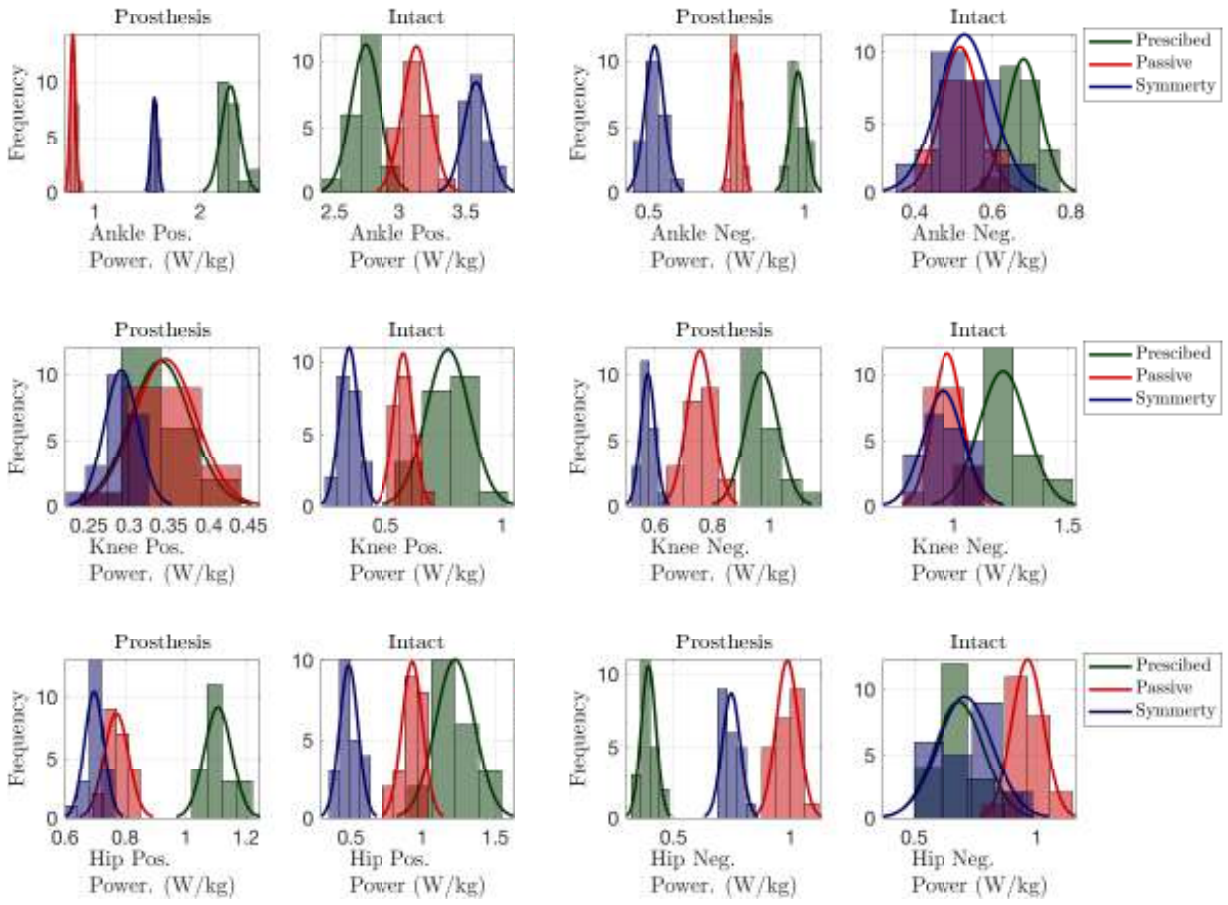


Figure H.9: Experiment 3 histograms for peak power outcomes.

Table H.4: Experiment 3 joint power outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Positive Ankle Power (W/kg)	$2.29(0.09)^{a}$	2.74(0.11)	$0.79(0.02)^{ab}$	$3.13(0.10)^{b}$	$1.56(0.03)^{abc}$	$3.58(0.09)^{bc}$
Peak Negative Ankle Power (W/kg)	$0.98(0.02)^{a}$	0.68(0.04)	$0.78(0.02)^{ab}$	$0.52(0.05)^{b}$	$0.52(0.03)^{bc}$	$0.53(0.07)^{b}$
Peak Positive Knee Power (W/kg)	$0.34(0.04)^{a}$	0.77(0.09)	$0.35(0.04)^{a}$	$0.58(0.04)^{b}$	$0.29(0.02)^{abc}$	$0.34(0.04)^{bc}$
Peak Negative Knee Power (W/kg)	$0.97(0.06)^{a}$	1.22(0.11)	$0.76(0.04)^{ab}$	$0.97(0.07)^{b}$	$0.58(0.03)^{abc}$	$0.95(0.09)^{b}$
Peak Positive Hip Power (W/kg)	$1.11(0.05)^{a}$	1.22(0.13)	$0.77(0.04)^{ab}$	$0.93(0.07)^{b}$	$0.70(0.03)^{abc}$	$0.49(0.06)^{bc}$
Peak Negative Hip Power (W/kg)	$0.39(0.03)^{a}$	0.69(0.10)	$0.99(0.05)^{b}$	$0.96(0.06)^{b}$	$0.75(0.04)^{bc}$	$0.71(0.11)^{c}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

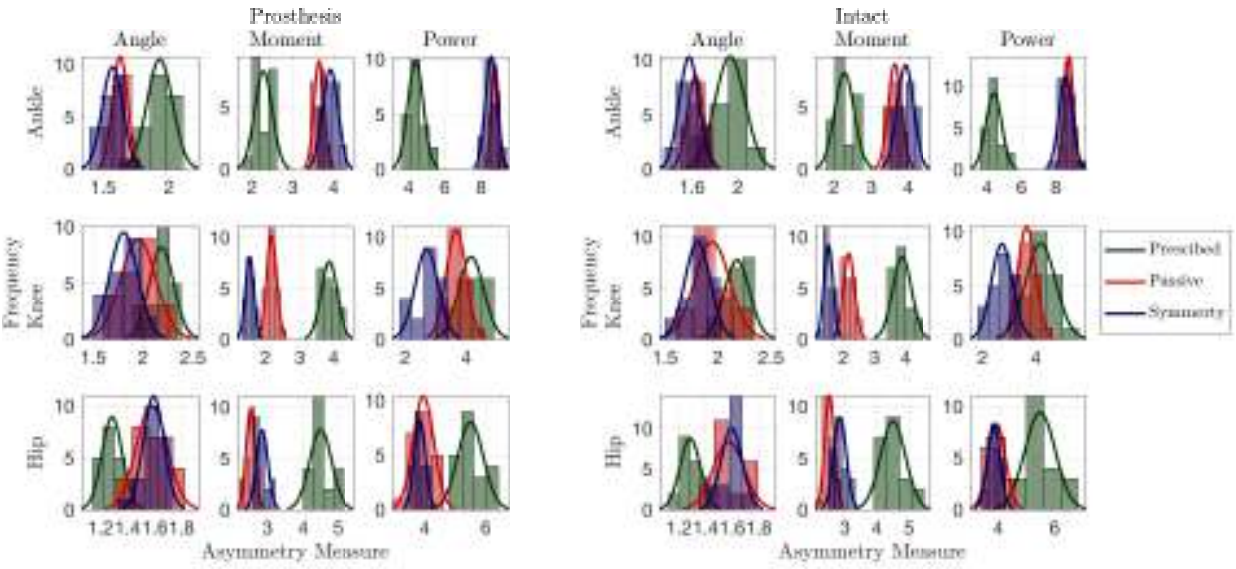


Figure H.10: Experiment 3 histograms for AM outcomes.

Table H.5: Experiment 3 sagittal plane AM outcome measures (standard deviations), defined in (6.2).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
AM GRF (BW)	2.42(0.37)	2.43(0.38)	5.46(0.14)^{ab}	6.15(0.29)^{b}	4.94(0.27)^{abc}	5.17(0.30)^{bc}
AM Ankle Angle (rad)	1.93(0.10)	1.93(0.13)	1.63(0.06)^{b}	1.63(0.07)^{b}	1.57(0.08)^{bc}	1.58(0.09)^{bc}
AM Ankle Moment (Nm/kg)	2.29(0.21)	2.28(0.26)	3.64(0.12)^{b}	3.63(0.18)^{b}	3.91(0.19)^{bc}	3.92(0.22)^{bc}
AM Ankle Power (W/kg)	4.38(0.43)	4.39(0.46)	8.77(0.20)^{b}	8.77(0.26)^{b}	8.56(0.33)^{bc}	8.58(0.39)^{b}
AM Knee Angle (rad)	2.19(0.12)	2.18(0.12)	1.95(0.18)^{b}	1.95(0.17)^{b}	1.81(0.14)^{bc}	1.81(0.12)^{bc}
AM Knee Moment (Nm/kg)	3.88(0.24)	3.88(0.30)	2.17(0.15)^{b}	2.17(0.18)^{b}	1.52(0.12)^{bc}	1.52(0.14)^{bc}
AM Knee Power (W/kg)	4.16(0.41)	4.18(0.57)	3.67(0.33)^{b}	3.65(0.34)^{b}	2.70(0.38)^{bc}	2.72(0.40)^{bc}
AM Hip Angle (rad)	1.29(0.08)	1.28(0.08)	1.58(0.11)^{b}	1.57(0.12)^{b}	1.59(0.08)^{b}	1.59(0.08)^{b}
AM Hip Moment (Nm/kg)	4.50(0.31)	4.50(0.39)	2.51(0.13)^{b}	2.51(0.15)^{b}	2.84(0.16)^{bc}	2.84(0.20)^{bc}
AM Hip Power (W/kg)	5.50(0.42)	5.49(0.54)	3.96(0.34)^{b}	3.95(0.31)^{b}	3.85(0.18)^{b}	3.85(0.22)^{b}

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

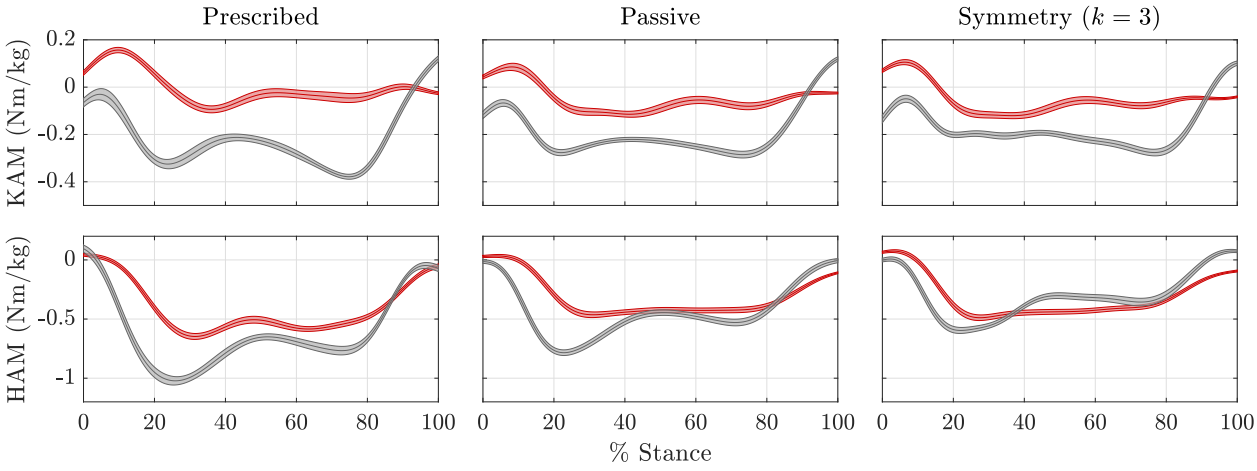


Figure H.11: Experiment 3 frontal plane hip moments (KAM and HAM) for the three experimental conditions.

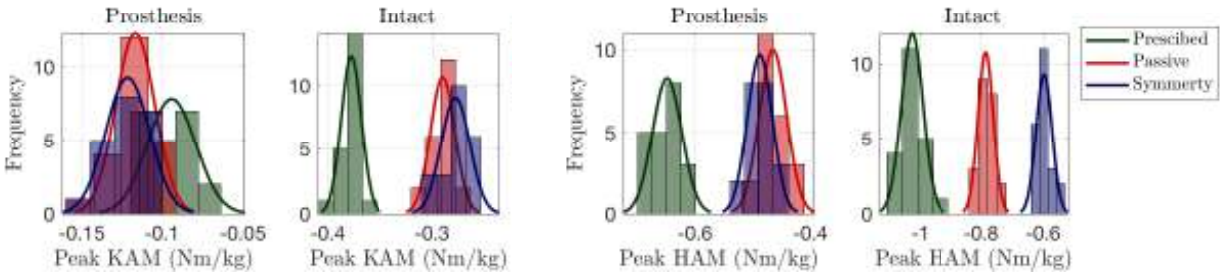


Figure H.12: Experiment 3 histograms for KAM and HAM outcomes.

Table H.6: Experiment 3 OA loading factors outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Knee Adduction Moment (Nm/kg)	$-0.09(0.02)^{a}$	$-0.38(0.01)$	$-0.12(0.01)^{ab}$	$-0.29(0.01)^{b}$	$-0.12(0.01)^{ab}$	$-0.28(0.01)^{bc}$
Peak Hip Adduction Moment (Nm/kg)	$-0.65(0.03)^{a}$	$-1.03(0.03)$	$-0.47(0.02)^{ab}$	$-0.79(0.02)^{b}$	$-0.49(0.02)^{abc}$	$-0.60(0.03)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

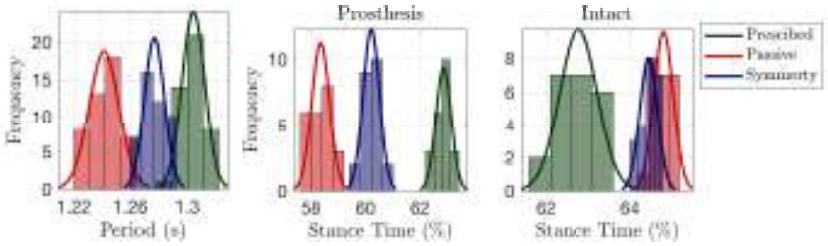


Figure H.13: Experiment 3 histograms for temporal outcomes.

Table H.7: Experiment 3 temporal symmetry outcomes measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 3$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Step Period (s)	1.30(0.01)		1.24(0.01)		1.28(0.01)	
Stance (%)	62.87(0.28)	62.78(0.45)	58.35(0.33)^{ab}	64.79(0.23)^{b}	60.21(0.30)^{abc}	64.44(0.21)^{bc}

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

Appendix I

EXPERIMENT 4 RESULTS

I.1 Summary

Results for experimental 4 are provided below. The results from this experiment demonstrate how a reduced learning harmonics can have adverse effect. In particular, the intact ankle torque grew by a large margin between each iteration, presumably because the control signal lacks sufficient content to properly encode the error (see Fig. I.1).

Table I.1: Algorithm and motor parameters for experiment 4.

Gearbox transmission	R_G	19.2
Effective transmission	R_T	1800
Torque constant (Nm/A)	k_τ	0.00775
Speed constant (rpm/V)	k_s	1223
Initial gain	ρ	1
Pre-smooth time (%)	t_1	4
Post-smooth time (%)	t_2	TO
Max harmonic (rad/s)	ω_{max}	$3 \cdot \omega_0$
Filter cutoff	ω_c	$3 \cdot \omega_0$
HS delay (ms)	Δ_{HS}	40
Decay rate	γ	2
Allowable reference growth factor	α	1

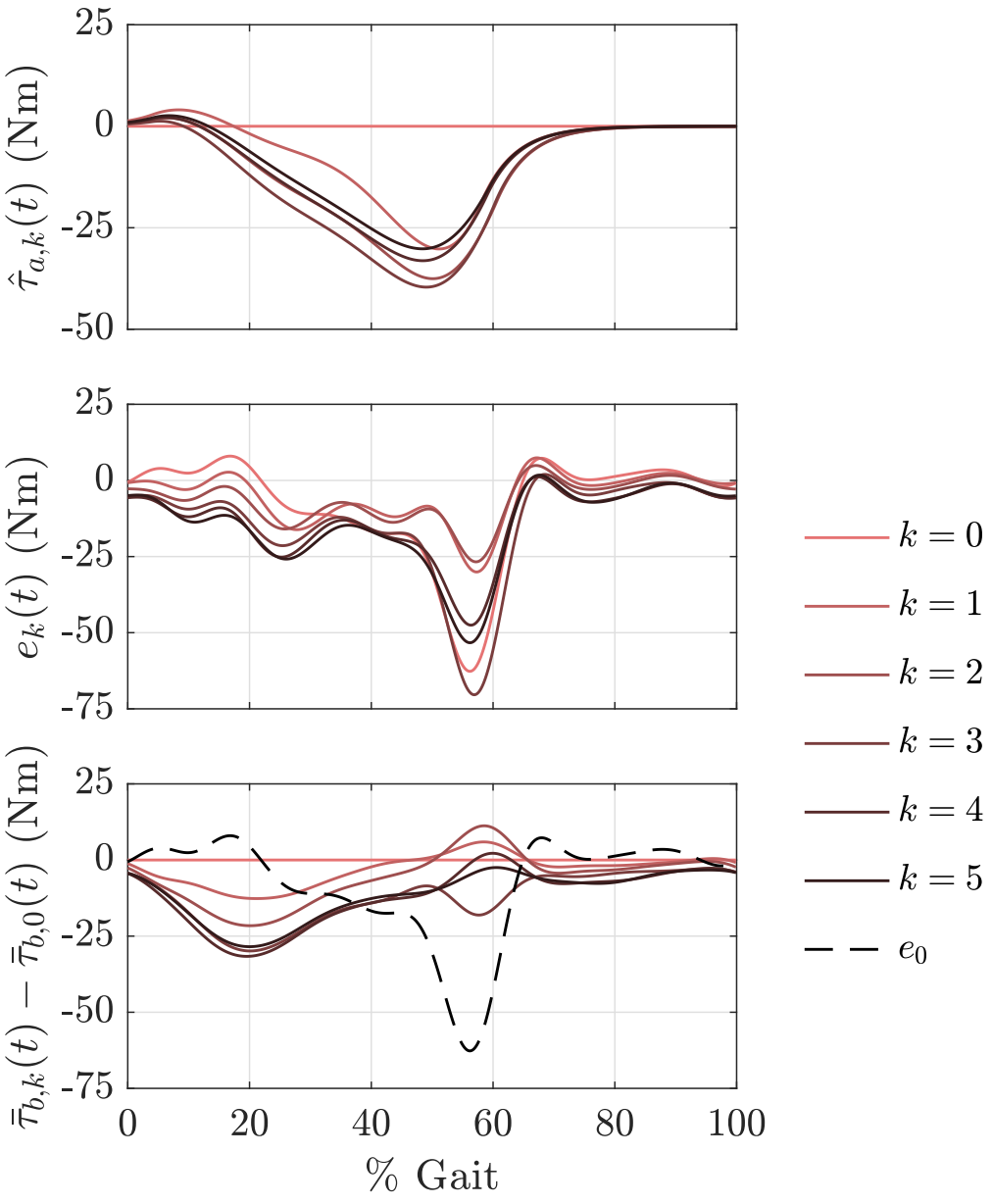


Figure I.1: Experiment 4 time domain evolution of active torque $\hat{\tau}_{a,k}$, error e_k, and the change in reference signal, e.g., $\bar{\tau}_{b,k} - \bar{\tau}_{b,0}$.

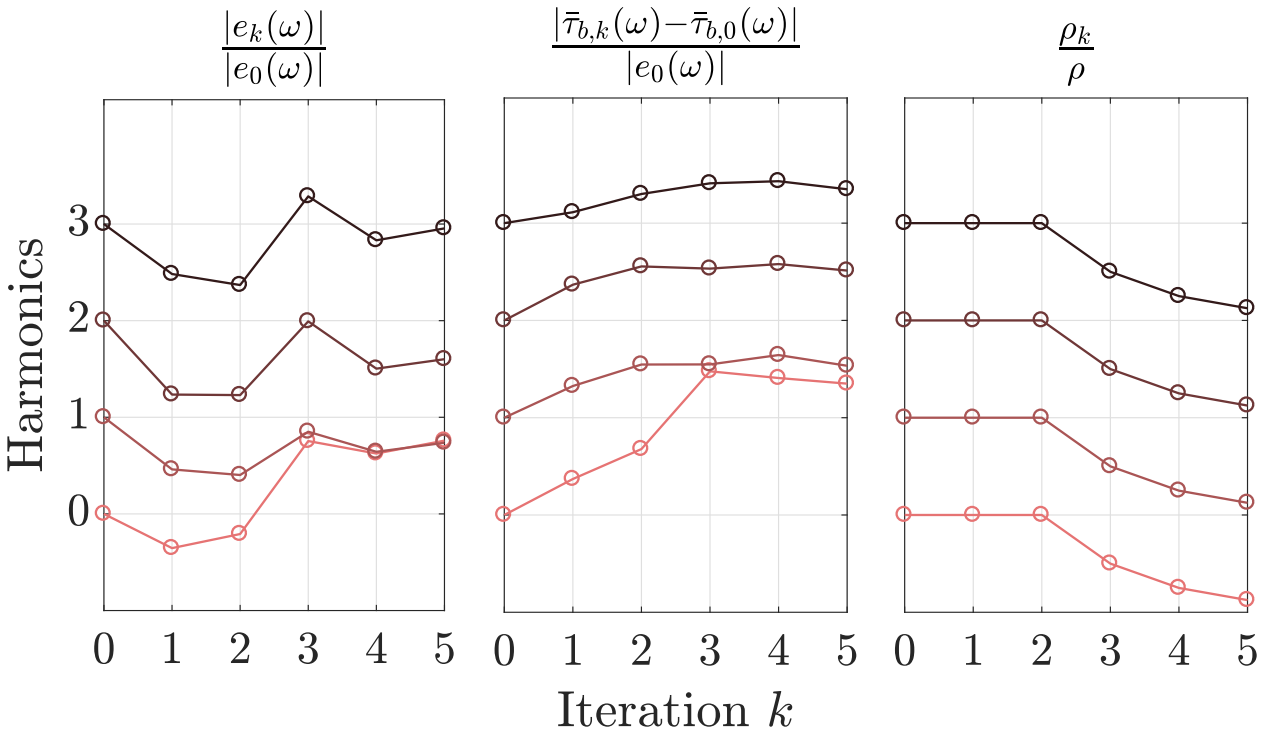


Figure I.2: Experiment 4 frequency domain evolution of the magnitude of error, normalized by the initial error, $|e_k|/|e_0|$, the magnitude of change in reference signal, normalized by the initial error $|\bar{\tau}_{b,k} - \bar{\tau}_{b,0}|/|e_0|$, and adaptation in ρ_k, normalized by the initial ρ_0.

Table I.2: Experiment 4 joint kinematic outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Plantarflexion (deg)	$1.99(0.19)^{a}$	$-16.66(1.85)$	$5.68(0.13)^{ab}$	$-19.77(1.07)^{b}$	$6.03(0.19)^{abc}$	$-20.48(1.18)^{bc}$
Peak Ankle Dorsiflexion (deg)	$24.90(0.13)^{a}$	$19.52(0.51)$	$26.15(0.27)^{ab}$	$18.72(0.58)^{b}$	$23.68(0.22)^{abc}$	$18.80(0.72)^{b}$
Peak Knee Flexion (deg)	$71.34(0.70)^{a}$	$59.30(1.10)$	$62.62(0.87)^{ab}$	$54.54(0.84)^{b}$	$60.43(0.56)^{abc}$	$55.82(0.95)^{bc}$
Peak Hip Extension (deg)	$13.29(0.36)^{a}$	$5.28(0.26)$	$7.15(0.49)^{ab}$	$1.10(0.55)^{b}$	$5.62(0.36)^{abc}$	$0.37(0.33)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

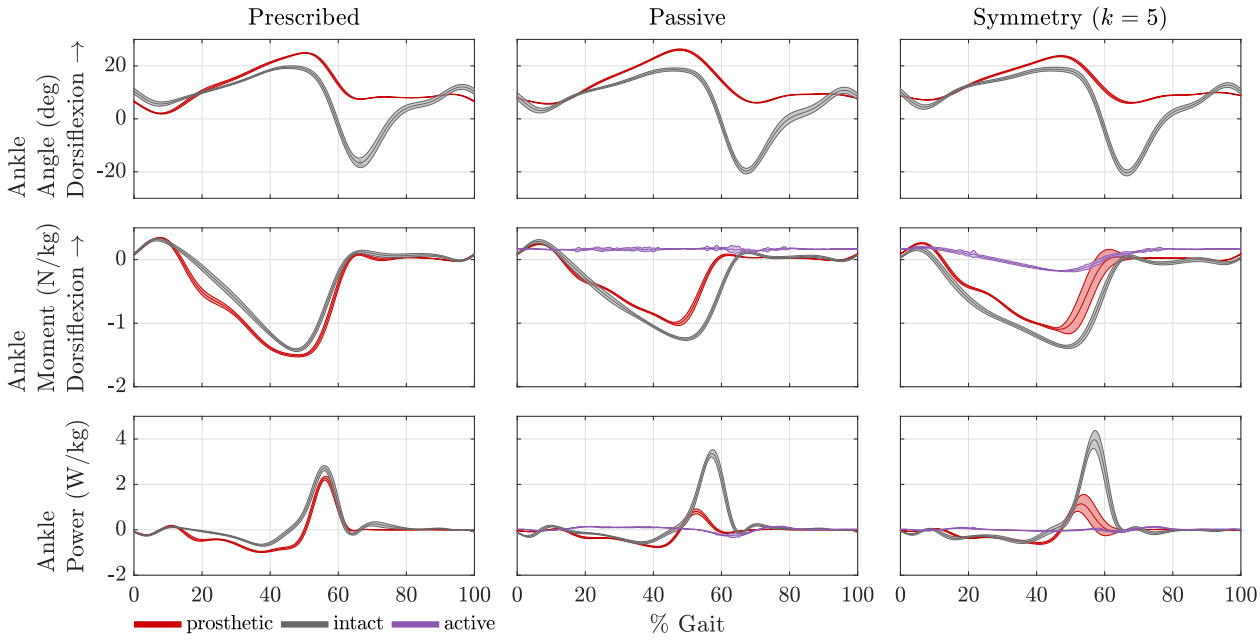


Figure I.3: Experiment 4 sagittal ankle mechanics for the three experimental conditions.

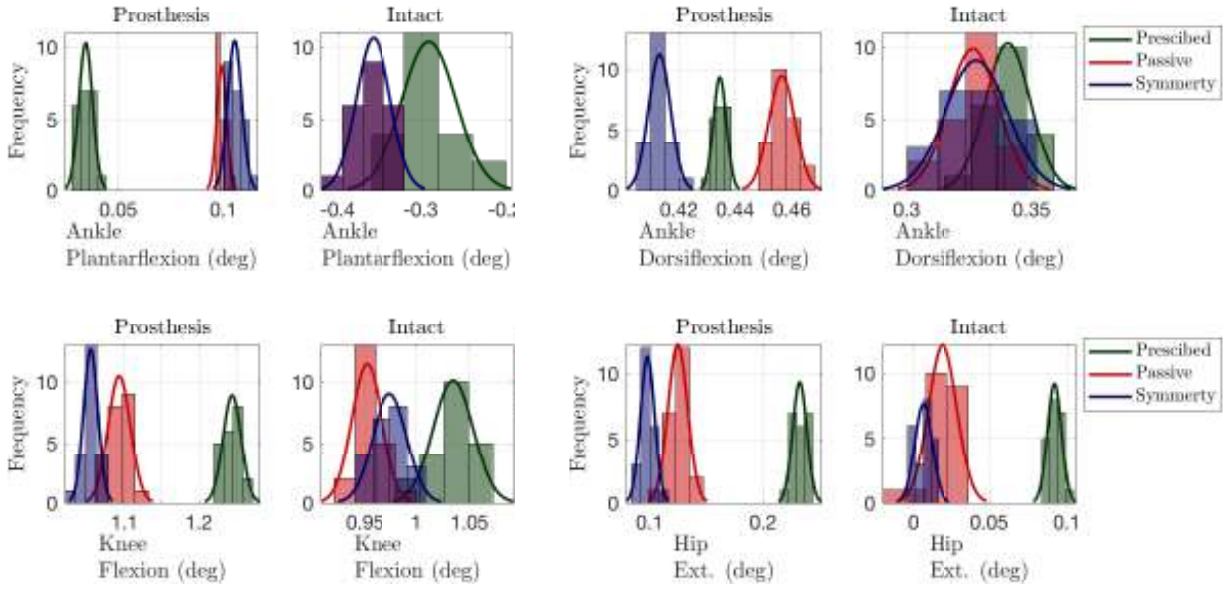


Figure I.4: Experiment 4 histograms for peak kinematic outcomes.

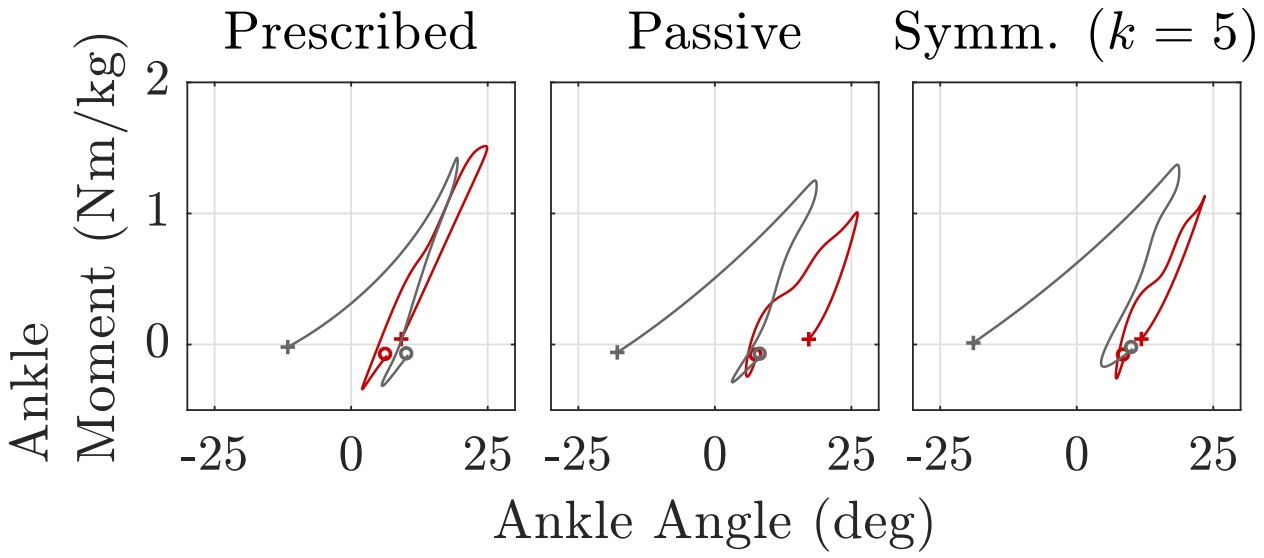


Figure I.5: Experiment 4 ankle work loops for the three experimental conditions.

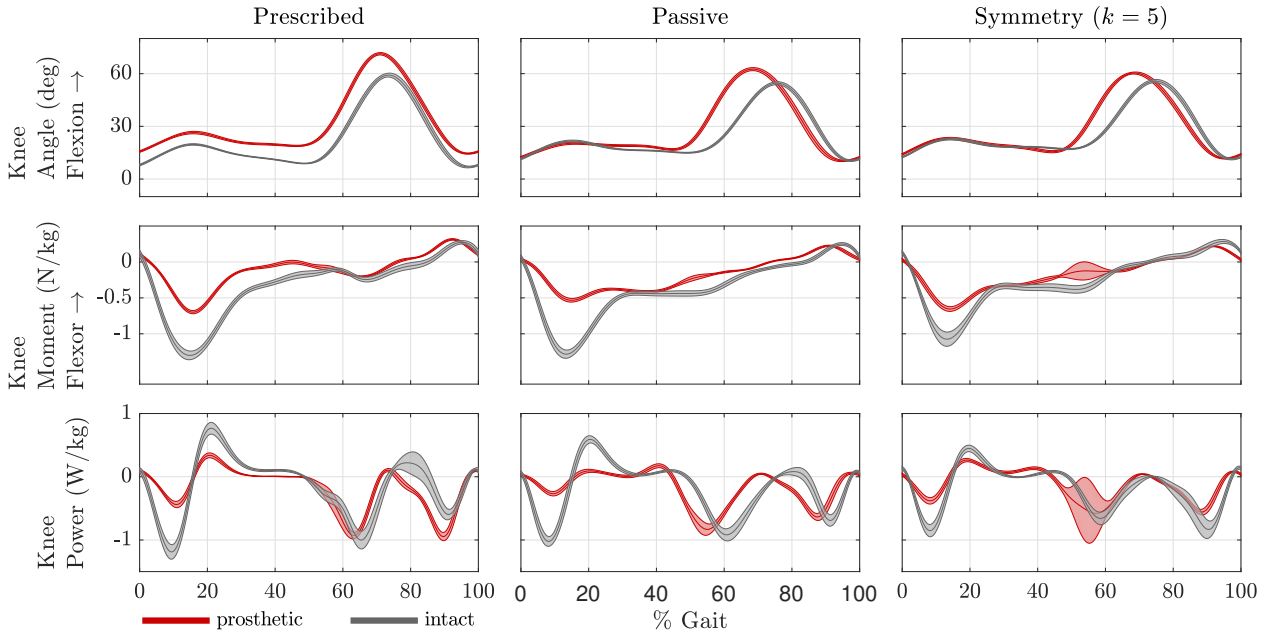


Figure I.6: Experiment 4 sagittal plane knee mechanics for the three experimental conditions.

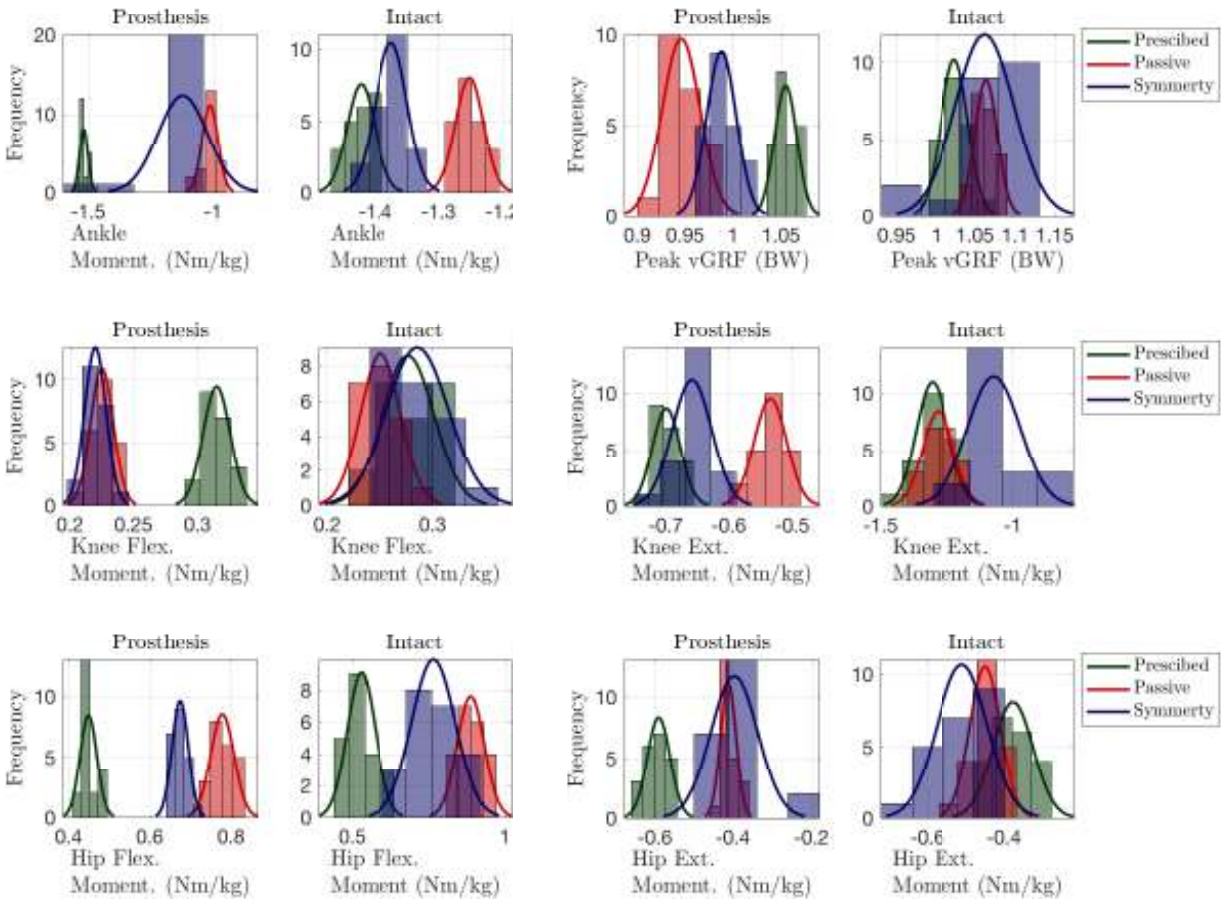


Figure I.7: Experiment 4 histograms for peak kinetic outcomes.

Table I.3: Experiment 4 joint kinetics and vGRF outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Moment (Nm/kg)	$-1.52(0.02)^{a}$	$-1.42(0.02)$	$-1.01(0.03)^{ab}$	$-1.25(0.02)^{b}$	$-1.12(0.10)^{abc}$	$-1.37(0.03)^{bc}$
Peak Knee Flexion Moment (Nm/kg)	$0.32(0.01)^{a}$	$0.28(0.02)$	$0.23(0.01)^{ab}$	$0.25(0.02)^{b}$	$0.22(0.01)^{ab}$	$0.29(0.03)^{c}$
Peak Knee Extension Moment (Nm/kg)	$-0.70(0.02)^{a}$	$-1.31(0.06)$	$-0.53(0.03)^{ab}$	$-1.28(0.06)$	$-0.66(0.03)^{abc}$	$-1.07(0.10)^{bc}$
Peak Hip Flexion Moment (Nm/kg)	$0.45(0.02)^{a}$	$0.53(0.05)$	$0.78(0.03)^{ab}$	$0.89(0.04)^{b}$	$0.67(0.02)^{abc}$	$0.77(0.07)^{bc}$
Peak Hip Extension Moment (Nm/kg)	$-0.59(0.03)^{a}$	$-0.38(0.05)$	$-0.41(0.02)^{ab}$	$-0.45(0.04)^{b}$	$-0.40(0.06)^{ab}$	$-0.51(0.07)^{bc}$
Peak vGRF (BW)	$1.05(0.01)^{a}$	$1.02(0.02)$	$0.95(0.02)^{ab}$	$1.06(0.01)^{b}$	$0.99(0.02)^{abc}$	$1.06(0.04)^{b}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

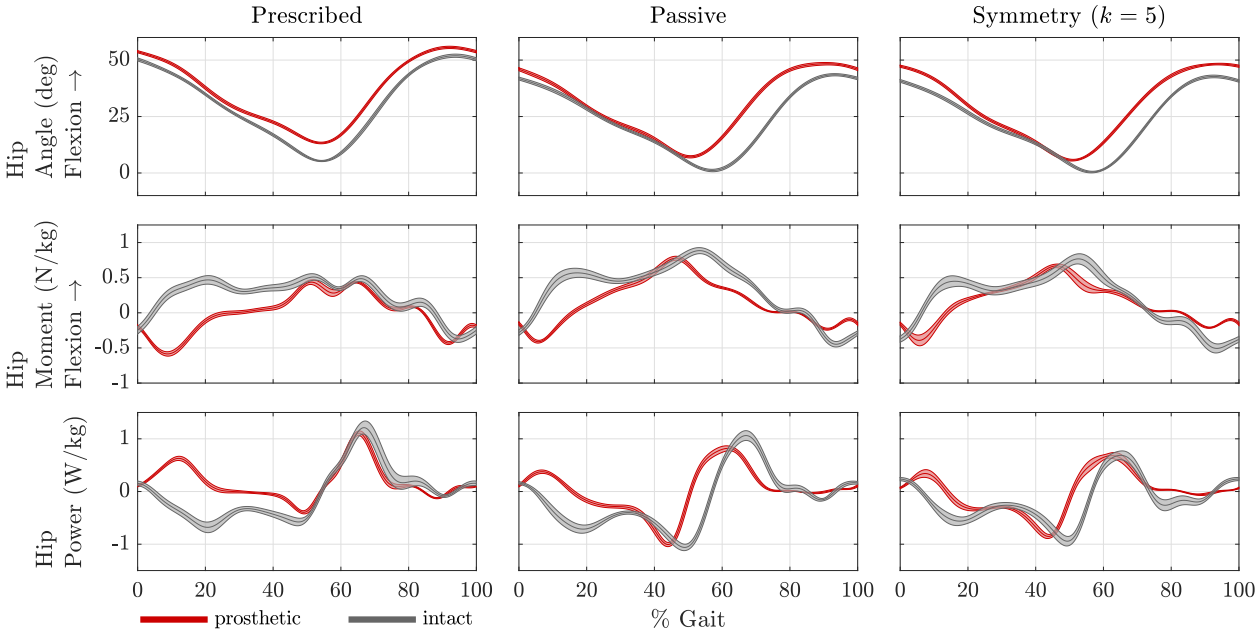


Figure I.8: Experiment 4 sagittal plane hip mechanics for the three experimental conditions.

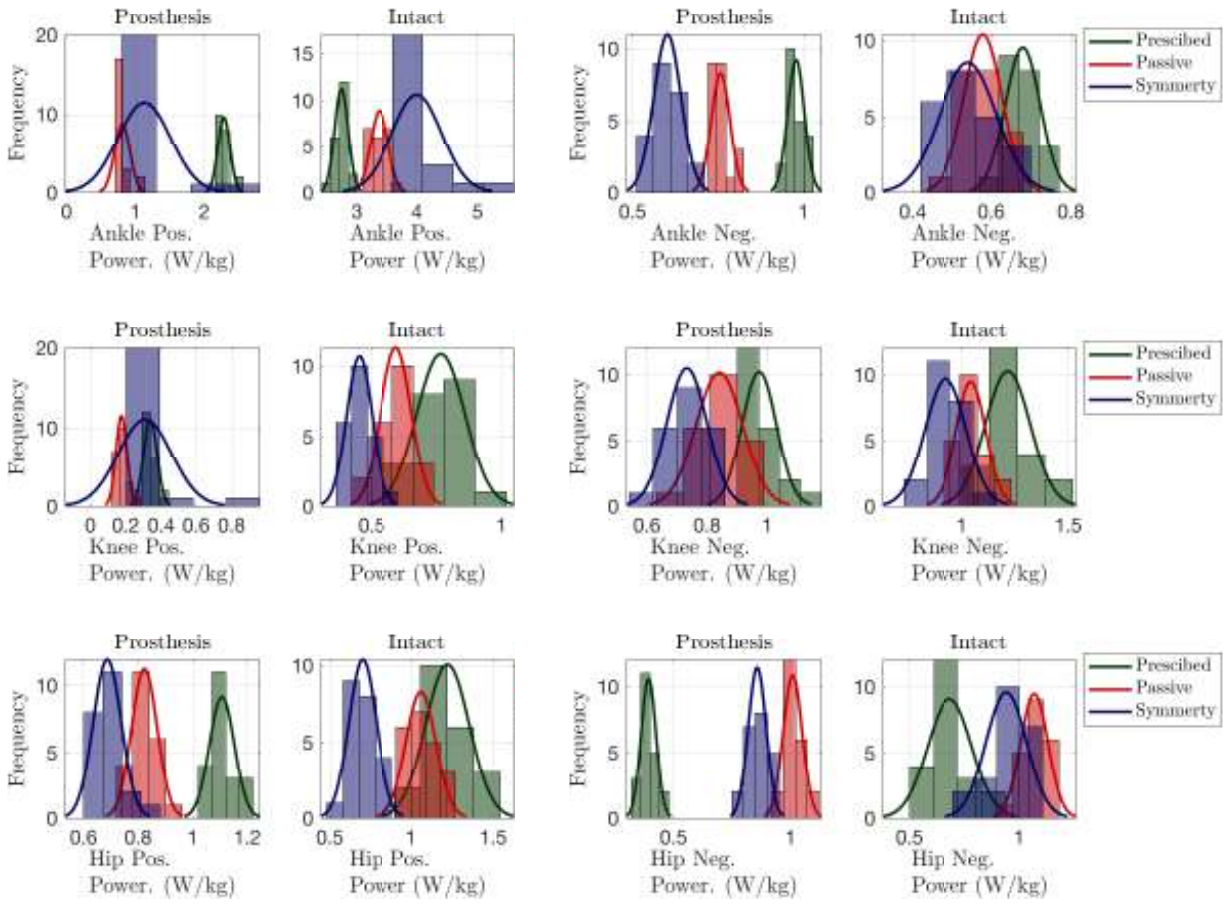


Figure I.9: Experiment 4 histograms for peak power outcomes.

Table I.4: Experiment 4 joint power outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Positive Ankle Power (W/kg)	$2.29(0.09)^{a}$	2.74(0.11)	$0.81(0.11)^{ab}$	$3.37(0.15)^{b}$	$1.13(0.38)^{abc}$	$4.00(0.41)^{bc}$
Peak Negative Ankle Power (W/kg)	$0.98(0.02)^{a}$	0.68(0.04)	$0.76(0.03)^{ab}$	$0.58(0.05)^{b}$	$0.60(0.04)^{abc}$	$0.54(0.07)^{b}$
Peak Positive Knee Power (W/kg)	$0.34(0.04)^{a}$	0.77(0.09)	$0.18(0.03)^{ab}$	$0.60(0.06)^{b}$	$0.31(0.15)^{ac}$	$0.45(0.05)^{bc}$
Peak Negative Knee Power (W/kg)	$0.97(0.06)^{a}$	1.22(0.11)	$0.84(0.08)^{ab}$	$1.04(0.07)^{b}$	$0.73(0.07)^{abc}$	$0.92(0.10)^{bc}$
Peak Positive Hip Power (W/kg)	$1.11(0.05)^{a}$	1.22(0.13)	$0.82(0.05)^{ab}$	$1.06(0.09)^{b}$	$0.69(0.05)^{bc}$	$0.70(0.08)^{bc}$
Peak Negative Hip Power (W/kg)	$0.39(0.03)^{a}$	0.69(0.10)	$1.01(0.04)^{ab}$	$1.07(0.06)^{b}$	$0.86(0.04)^{abc}$	$0.94(0.09)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

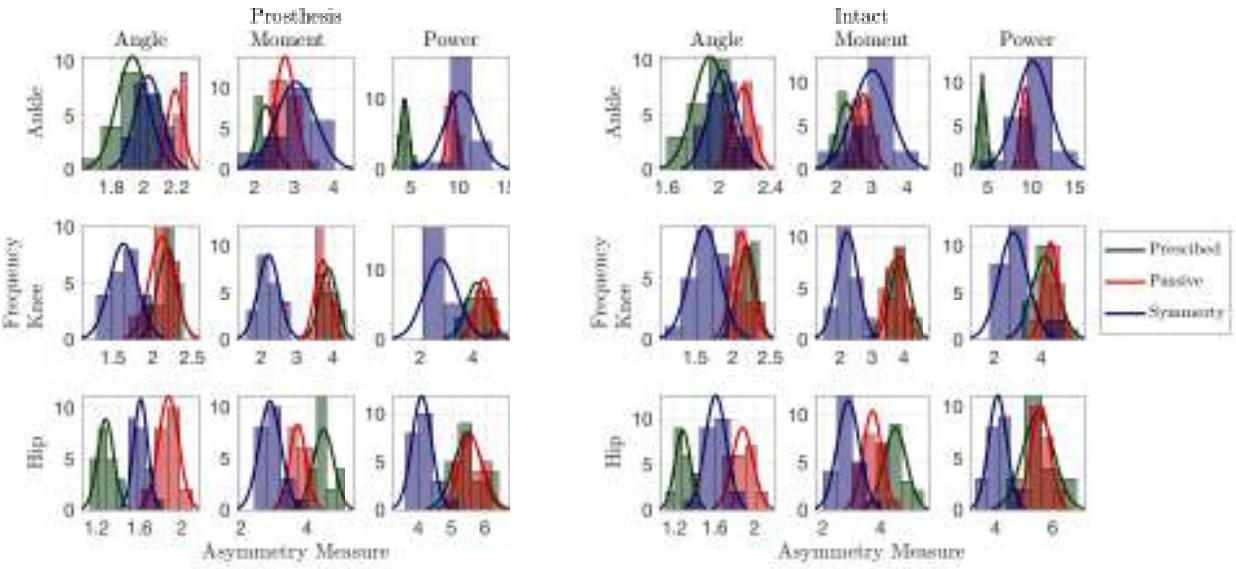


Figure I.10: Experiment 4 histograms for AM outcomes.

Table I.5: Experiment 4 sagittal plane AM outcome measures (standard deviations), defined in (6.2).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
AM GRF (BW)	2.42(0.37)	2.43(0.38)	5.76(0.23)$^{a\,b}$	6.64(0.35)^{b}	5.00(0.59)$^{a\,b\,c}$	5.47(0.85)$^{b\,c}$
AM Ankle Angle (rad)	1.93(0.10)	1.93(0.13)	2.19(0.05)^{b}	2.19(0.08)^{b}	2.03(0.08)$^{b\,c}$	2.03(0.11)$^{b\,c}$
AM Ankle Moment (Nm/kg)	2.29(0.21)	2.28(0.26)	2.77(0.25)^{b}	2.77(0.24)^{b}	3.04(0.49)$^{b\,c}$	3.01(0.54)^{b}
AM Ankle Power (W/kg)	4.38(0.43)	4.39(0.46)	9.32(0.48)^{b}	9.32(0.53)^{b}	10.30(1.69)$^{b\,c}$	10.19(1.97)^{b}
AM Knee Angle (rad)	2.19(0.12)	2.18(0.12)	2.12(0.15)	2.12(0.12)	1.63(0.18)$^{b\,c}$	1.63(0.20)$^{b\,c}$
AM Knee Moment (Nm/kg)	3.88(0.24)	3.88(0.30)	3.72(0.22)^{b}	3.71(0.29)	2.21(0.29)$^{b\,c}$	2.22(0.34)$^{b\,c}$
AM Knee Power (W/kg)	4.16(0.41)	4.18(0.57)	4.43(0.30)^{b}	4.43(0.32)	2.79(0.61)$^{b\,c}$	2.81(0.62)$^{b\,c}$
AM Hip Angle (rad)	1.29(0.08)	1.28(0.08)	1.88(0.09)^{b}	1.88(0.10)^{b}	1.62(0.06)$^{b\,c}$	1.61(0.10)$^{b\,c}$
AM Hip Moment (Nm/kg)	4.50(0.31)	4.50(0.39)	3.72(0.25)^{b}	3.72(0.32)^{b}	2.88(0.33)$^{b\,c}$	2.88(0.38)$^{b\,c}$
AM Hip Power (W/kg)	5.50(0.42)	5.49(0.54)	5.57(0.34)	5.57(0.33)	4.10(0.30)$^{b\,c}$	4.08(0.32)$^{b\,c}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

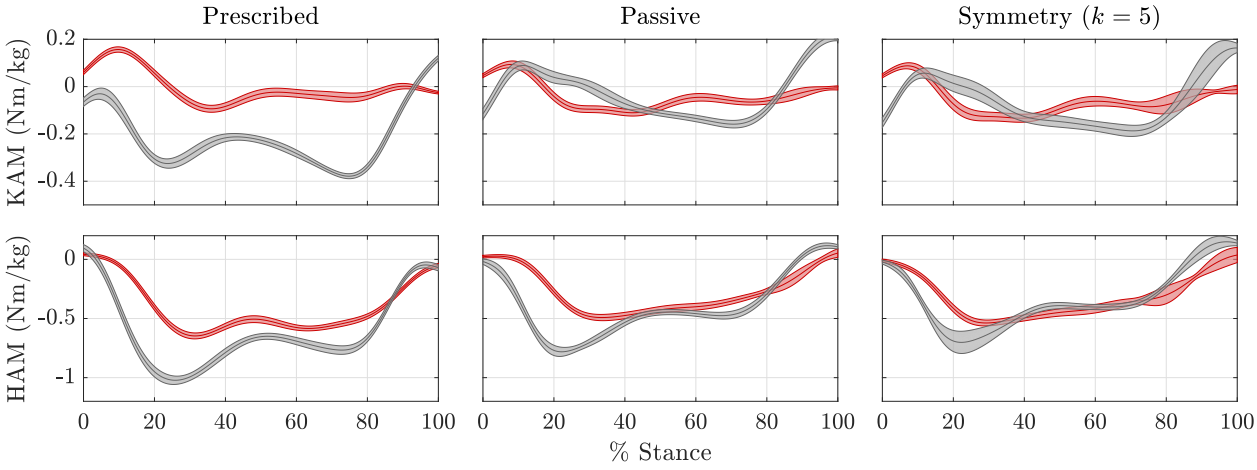


Figure I.11: Experiment 4 frontal plane hip moments (KAM and HAM) for the three experimental conditions.

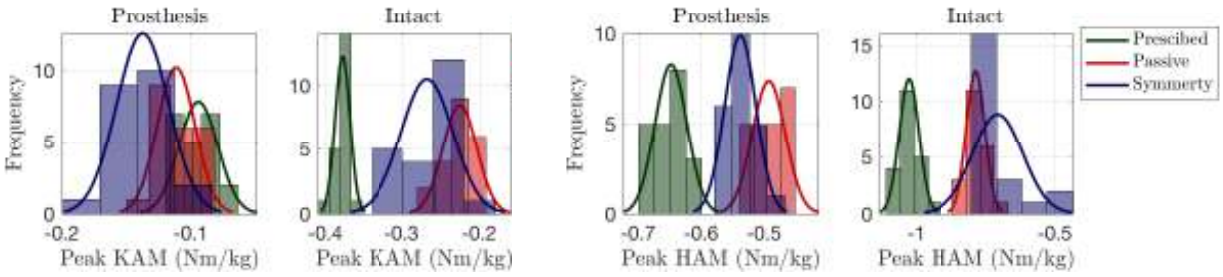


Figure I.12: Experiment 4 histograms for KAM and HAM outcomes.

Table I.6: Experiment 4 OA loading factors outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Knee Adduction Moment (Nm/kg)	$-0.09(0.02)^{a}$	$-0.38(0.01)$	$-0.11(0.01)^{ab}$	$-0.23(0.02)^{b}$	$-0.14(0.02)^{abc}$	$-0.27(0.03)^{bc}$
Peak Hip Adduction Moment (Nm/kg)	$-0.65(0.03)^{a}$	$-1.03(0.03)$	$-0.49(0.03)^{ab}$	$-0.78(0.03)^{b}$	$-0.54(0.02)^{abc}$	$-0.71(0.09)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

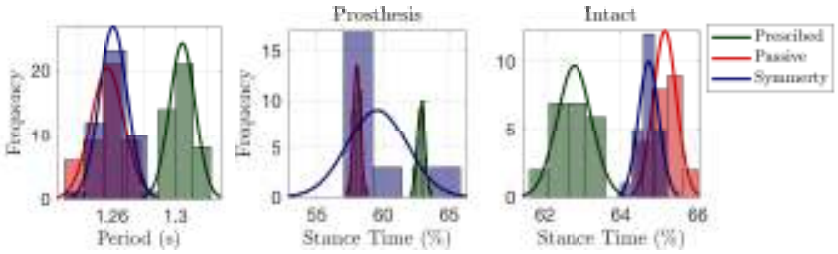


Figure I.13: Experiment 4 histograms for temporal outcomes.

Table I.7: Experiment 4 temporal symmetry outcomes measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 5$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Step Period (s)	1.30(0.01)		1.26(0.01)		1.26(0.01)	
Stance (%)	62.87(0.28)	62.78(0.45)	58.01(0.26)$^{a\,b}$	65.15(0.30)^{b}	59.55(2.25)$^{a\,b\,c}$	64.72(0.27)$^{b\,c}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

Appendix J

EXPERIMENT 5 RESULTS

J.1 Summary

Results for experimental 5 are provided below. The results from this experiment again demonstrate how reduced learning harmonics can have adverse effect. In particular, the intact ankle torque grew by a large margin between each iteration, presumably because the control signal lacks sufficient content to properly encode the error (see Fig. I.1). Additionally, this is the clearest example of the major flaw of the leaning algorithm: throwing out information that increasing the error. From Figs. J.1-J.2 it can be seen that the first control signal ($k = 1$) resulted in a large increase in error and a large deviation in the reference signal. At the next iteration, the algorithm (see Ch. 4) backsteps to the previous error at harmonics where the error grew, however for harmonics that did not grow, the algorithm keeps the control signal and adds the new error. The adverse effect is substantial growth in the signal early in stance (Fig. J.1).

Table J.1: Algorithm and motor parameters for experiment 5.

Gearbox transmission	R_G	19.2
Effective transmission	R_T	1800
Torque constant (Nm/A)	k_τ	0.00775
Speed constant (rpm/V)	k_s	1223
Initial gain	ρ	1
Pre-smooth time (%)	t_1	4
Post-smooth time (%)	t_2	TO
Max harmonic (rad/s)	ω_{max}	$6 \cdot \omega_0$
Filter cutoff	ω_c	$3 \cdot \omega_0$
HS delay (ms)	Δ_{HS}	40
Decay rate	γ	2
Allowable reference growth factor	α	1

Table J.2: Experiment 5 joint kinematic outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 4$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Plantarflexion (deg)	$1.99(0.19)^a$	$-16.66(1.85)$	$5.68(0.13)^{ab}$	$-19.77(1.07)^b$	$3.11(0.47)^{abc}$	$-16.73(1.21)^c$
Peak Ankle Dorsiflexion (deg)	$24.90(0.13)^a$	$19.52(0.51)$	$26.15(0.27)^{ab}$	$18.72(0.58)^b$	$23.72(0.21)^{abc}$	$21.49(0.57)^{bc}$
Peak Knee Flexion (deg)	$71.34(0.70)^a$	$59.30(1.10)$	$62.62(0.87)^{ab}$	$54.54(0.84)^b$	$61.65(0.85)^{abc}$	$58.68(0.64)^{bc}$
Peak Hip Extension (deg)	$13.29(0.36)^a$	$5.28(0.26)$	$7.15(0.49)^{ab}$	$1.10(0.55)^b$	$2.67(0.42)^{abc}$	$-1.32(0.39)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

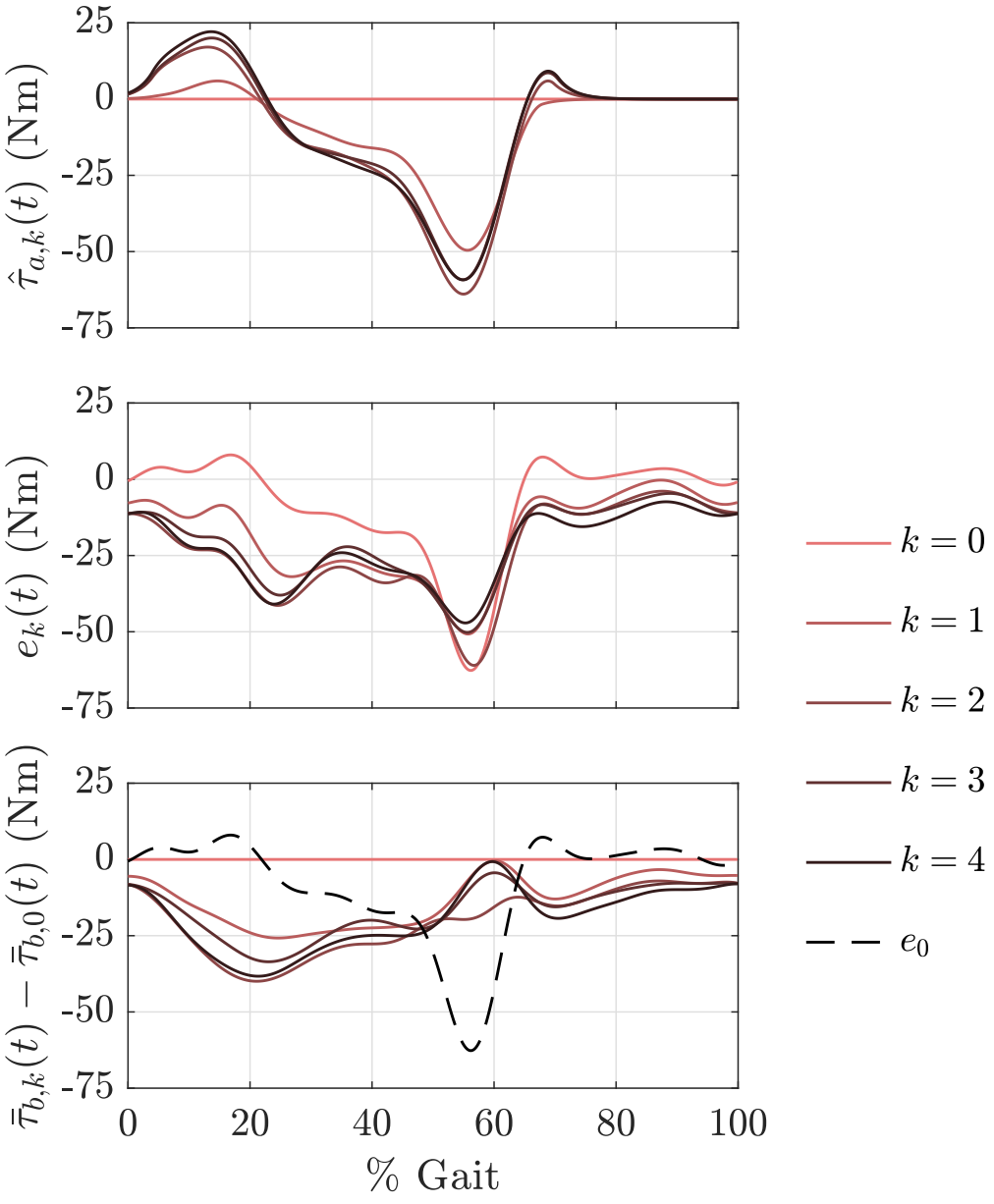


Figure J.1: Experiment 5 time domain evolution of active torque $\hat{\tau}_{a,k}$, error e_k, and the change in reference signal, e.g., $\bar{\tau}_{b,k} - \bar{\tau}_{b,0}$.

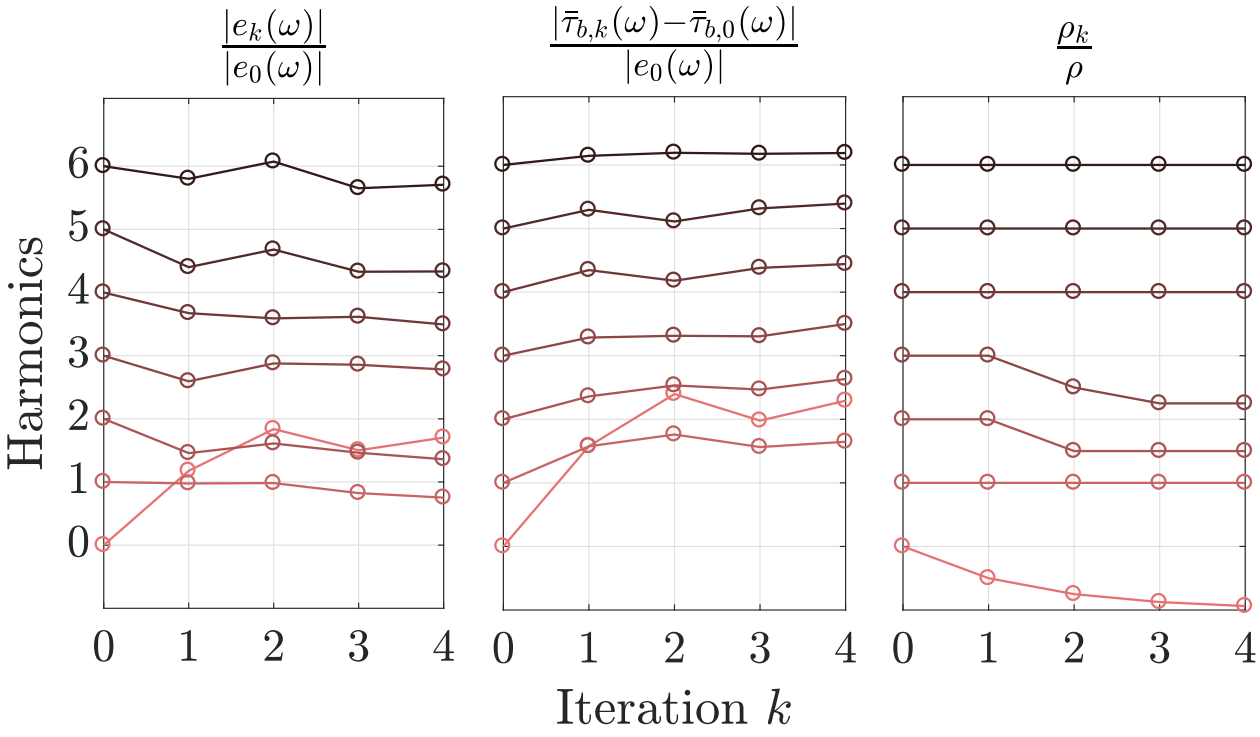


Figure J.2: Experiment 5 frequency domain evolution of the magnitude of error, normalized by the initial error, $|e_k|/|e_0|$, the magnitude of change in reference signal, normalized by the initial error $|\bar{\tau}_{b,k} - \bar{\tau}_{b,0}|/|e_0|$, and adaptation in ρ_k, normalized by the initial ρ_0.

Table J.3: Experiment 5 joint kinetics and vGRF outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 4$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Moment (Nm/kg)	$-1.52(0.02)^{a}$	$-1.42(0.02)$	$-1.01(0.03)^{ab}$	$-1.25(0.02)^{b}$	$-1.15(0.01)^{abc}$	$-1.52(0.03)^{bc}$
Peak Knee Flexion Moment (Nm/kg)	$0.32(0.01)^{a}$	$0.28(0.02)$	$0.23(0.01)^{ab}$	$0.25(0.02)^{b}$	$0.21(0.01)^{abc}$	$0.40(0.02)^{bc}$
Peak Knee Extension Moment (Nm/kg)	$-0.70(0.02)^{a}$	$-1.31(0.06)$	$-0.53(0.03)^{ab}$	$-1.28(0.06)$	$-0.70(0.03)^{ac}$	$-0.95(0.06)^{bc}$
Peak Hip Flexion Moment (Nm/kg)	$0.45(0.02)^{a}$	$0.53(0.05)$	$0.78(0.03)^{ab}$	$0.89(0.04)^{b}$	$0.63(0.02)^{abc}$	$0.53(0.07)^{c}$
Peak Hip Extension Moment (Nm/kg)	$-0.59(0.03)^{a}$	$-0.38(0.05)$	$-0.41(0.02)^{ab}$	$-0.45(0.04)^{b}$	$-0.36(0.02)^{abc}$	$-0.68(0.03)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

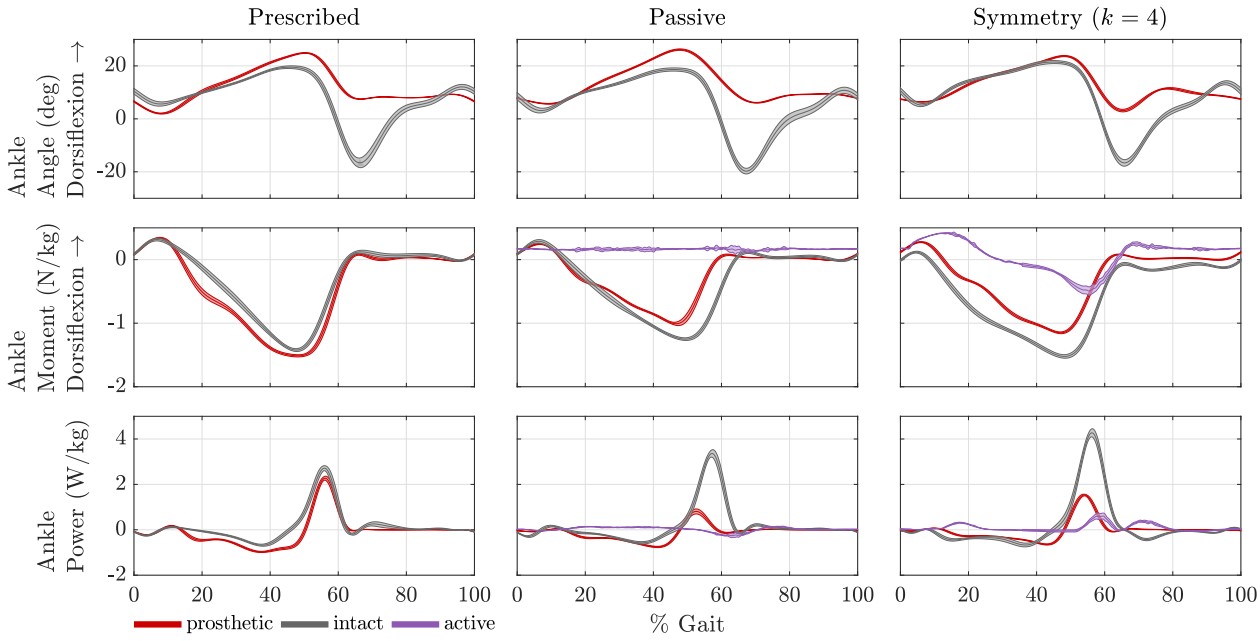


Figure J.3: Experiment 5 sagittal ankle mechanics for the three experimental conditions.

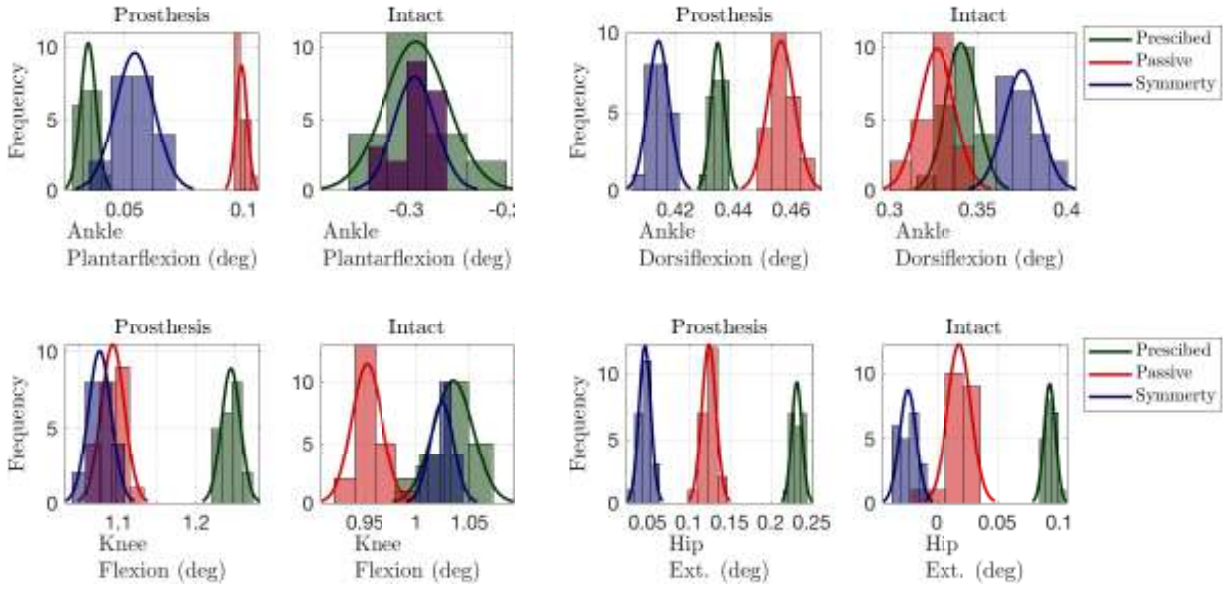


Figure J.4: Experiment 5 histograms for peak kinematic outcomes.

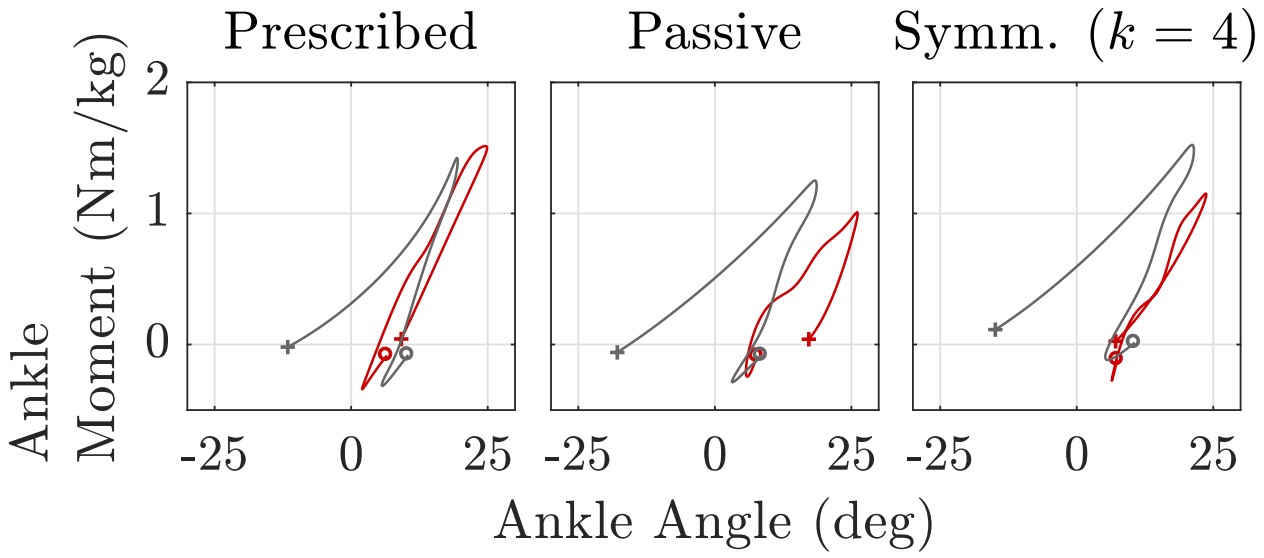


Figure J.5: Experiment 5 ankle work loops for the three experimental conditions.

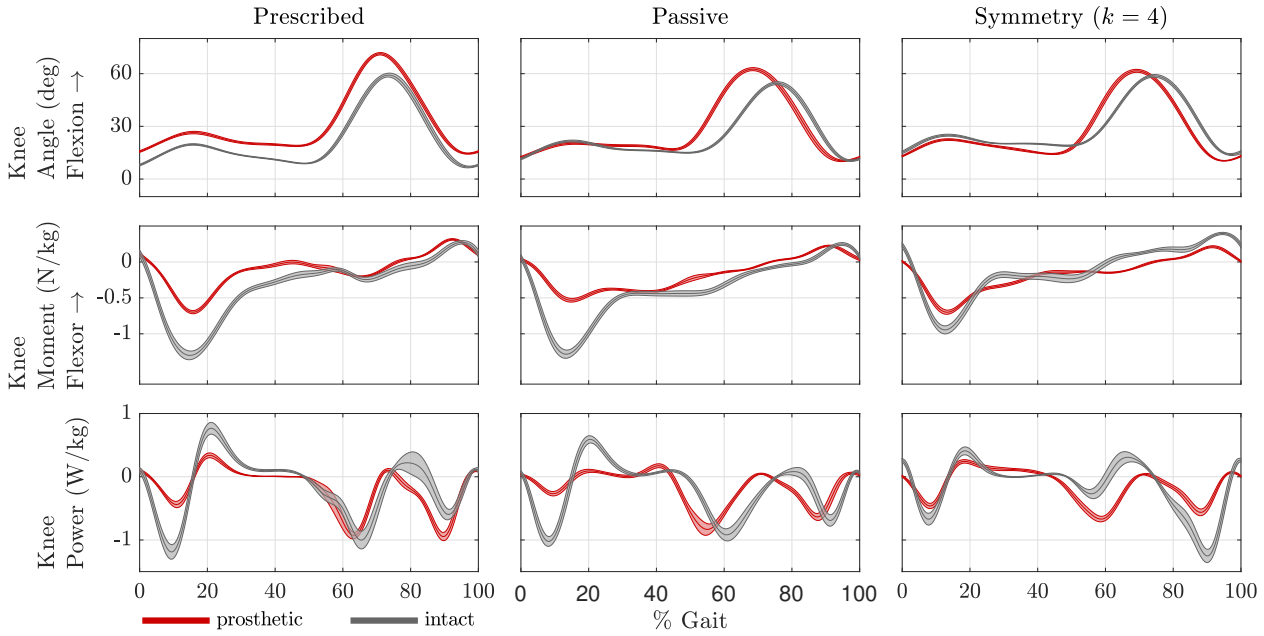


Figure J.6: Experiment 5 sagittal plane knee mechanics for the three experimental conditions.

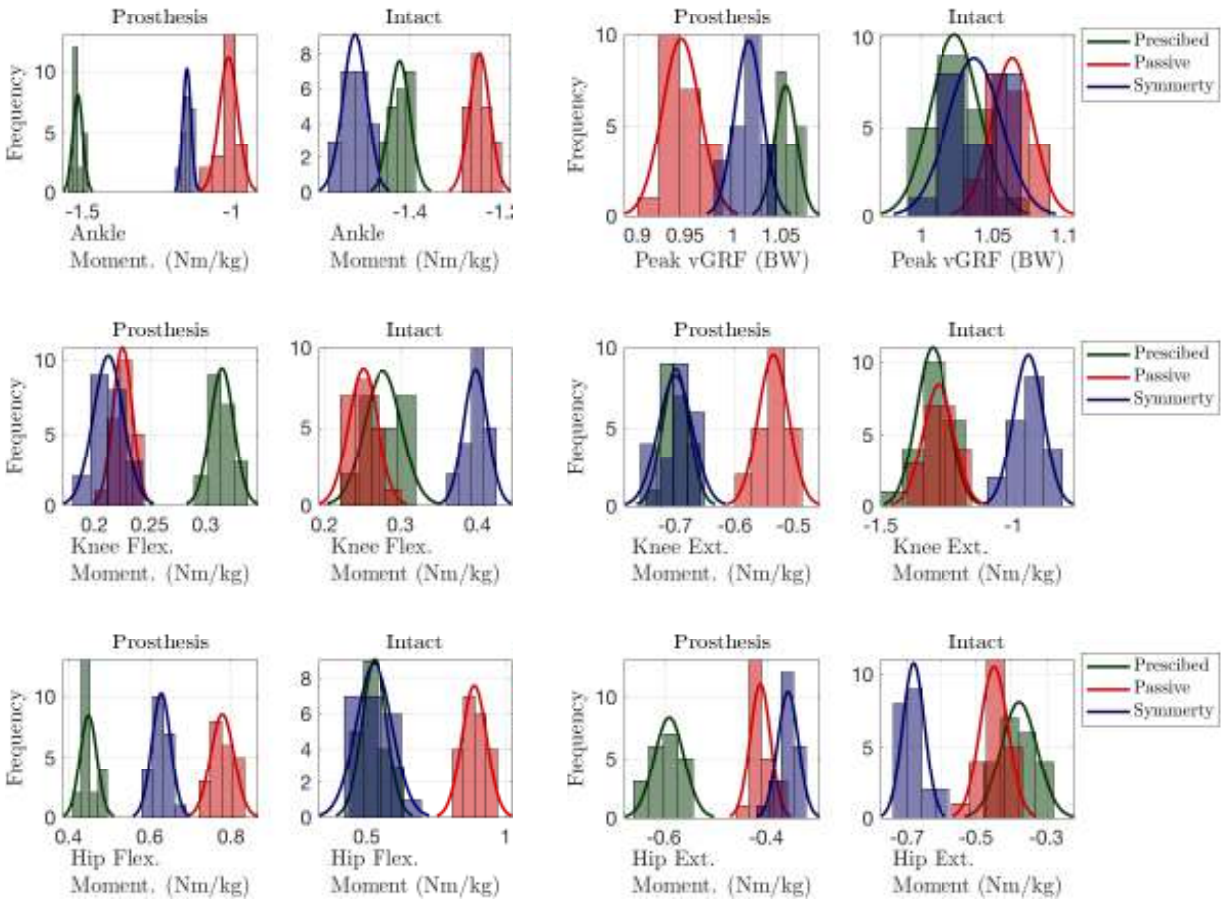


Figure J.7: Experiment 5 histograms for peak kinetic outcomes.

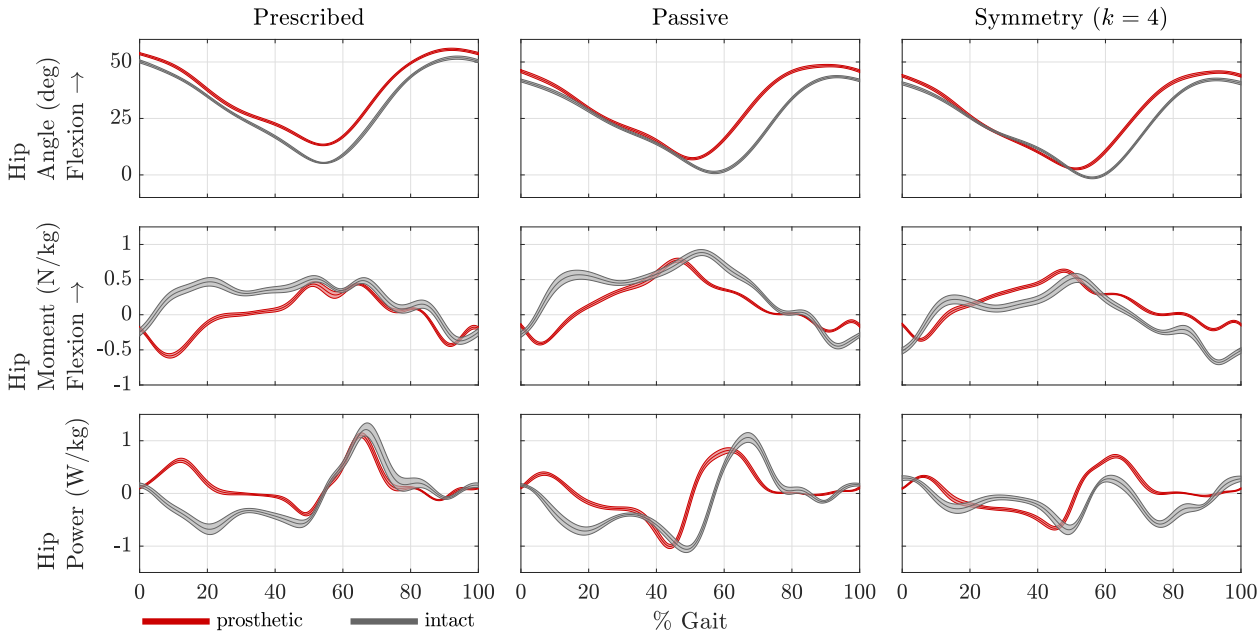


Figure J.8: Experiment 5 sagittal plane hip mechanics for the three experimental conditions.

Table J.4: Experiment 5 joint power outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 4$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Positive Ankle Power (W/kg)	2.29(0.09)^{a}	2.74(0.11)	0.81(0.11)^{ab}	3.37(0.15)^{b}	1.54(0.04)^{abc}	4.27(0.19)^{bc}
Peak Negative Ankle Power (W/kg)	0.98(0.02)^{a}	0.68(0.04)	0.76(0.03)^{ab}	0.58(0.05)^{b}	0.65(0.03)^{bc}	0.67(0.08)^{c}
Peak Positive Knee Power (W/kg)	0.34(0.04)^{a}	0.77(0.09)	0.18(0.03)^{ab}	0.60(0.06)^{b}	0.25(0.03)^{abc}	0.41(0.06)^{bc}
Peak Negative Knee Power (W/kg)	0.97(0.06)^{a}	1.22(0.11)	0.84(0.08)^{ab}	1.04(0.07)^{b}	0.67(0.04)^{abc}	1.26(0.11)^{c}
Peak Positive Hip Power (W/kg)	0.34(0.04)^{a}	0.77(0.09)	0.18(0.03)^{ab}	0.60(0.06)^{b}	0.25(0.03)^{abc}	0.41(0.06)^{bc}
Peak Negative Hip Power (W/kg)	0.97(0.06)^{a}	1.22(0.11)	0.84(0.08)^{ab}	1.04(0.07)^{b}	0.67(0.04)^{abc}	1.26(0.11)^{c}

^{a} Indicates significant difference between limbs within the same condition.

^{b} Indicates significant difference when compared with the prescribed condition for the same limb.

^{c} Indicates significant difference when compared with the passive mode condition for the same limb.

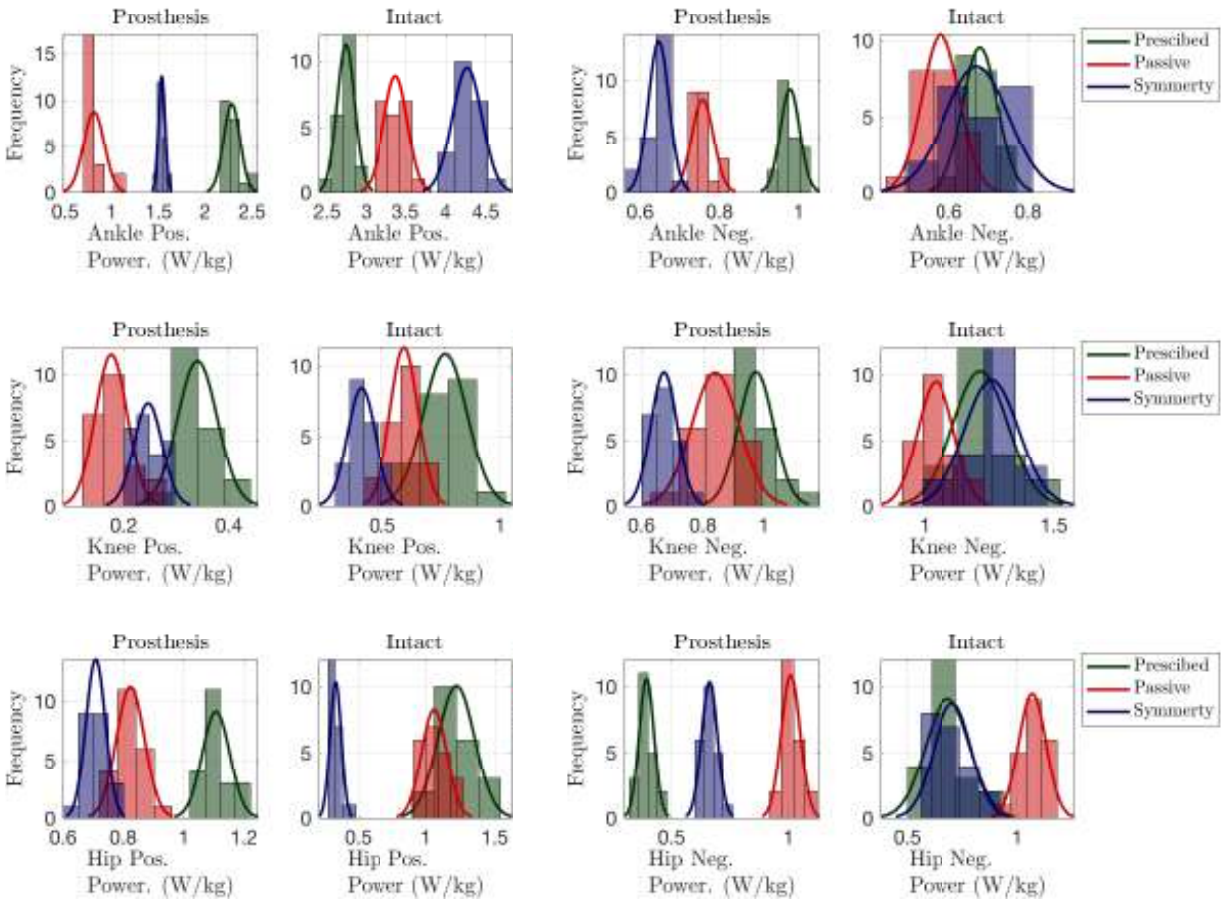


Figure J.9: Experiment 5 histograms for peak power outcomes.

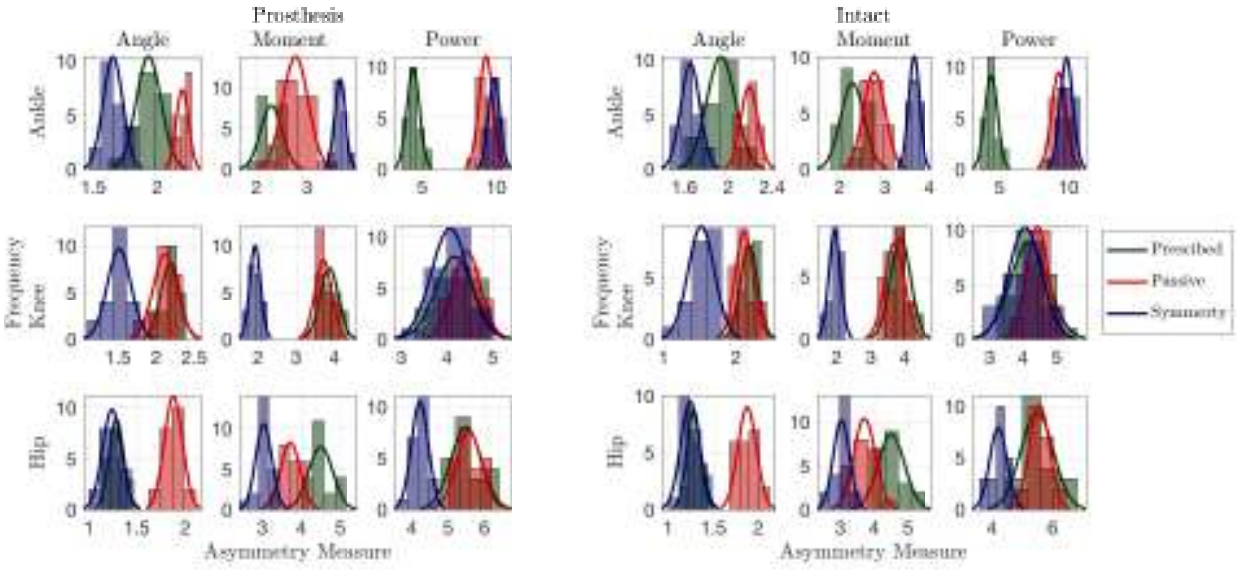


Figure J.10: Experiment 5 histograms for AM outcomes.

Table J.5: Experiment 5 sagittal plane AM outcome measures (standard deviations), defined in (6.2).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 4$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
AM GRF (BW)	2.42(0.37)	2.43(0.38)	5.76(0.23)$^{a\,b}$	6.64(0.35)^{b}	3.63(0.28)^{bc}	3.74(0.32)^{bc}
AM Ankle Angle (rad)	1.93(0.10)	1.93(0.13)	2.19(0.05)^{b}	2.19(0.08)^{b}	1.65(0.08)^{bc}	1.66(0.09)^{bc}
AM Ankle Moment (Nm/kg)	2.29(0.21)	2.28(0.26)	2.77(0.25)^{b}	2.77(0.24)^{b}	3.66(0.11)^{bc}	3.65(0.12)^{bc}
AM Ankle Power (W/kg)	4.38(0.43)	4.39(0.46)	9.32(0.48)^{b}	9.32(0.53)^{b}	9.88(0.39)^{bc}	9.87(0.46)^{bc}
AM Knee Angle (rad)	2.19(0.12)	2.18(0.12)	2.12(0.15)	2.12(0.12)	1.52(0.15)^{bc}	1.52(0.18)^{bc}
AM Knee Moment (Nm/kg)	3.88(0.24)	3.88(0.30)	3.72(0.22)^{b}	3.71(0.29)	1.93(0.14)^{bc}	1.94(0.17)^{bc}
AM Knee Power (W/kg)	4.16(0.41)	4.18(0.57)	4.43(0.30)^{b}	4.43(0.32)	4.06(0.40)^{c}	4.06(0.49)^{c}
AM Hip Angle (rad)	1.29(0.08)	1.28(0.08)	1.88(0.09)^{b}	1.88(0.10)^{b}	1.24(0.10)^{c}	1.25(0.11)^{c}
AM Hip Moment (Nm/kg)	4.50(0.31)	4.50(0.39)	3.72(0.25)^{b}	3.72(0.32)^{b}	3.04(0.21)^{bc}	3.04(0.24)^{bc}
AM Hip Power (W/kg)	5.50(0.42)	5.49(0.54)	5.57(0.34)	5.57(0.33)	4.22(0.24)^{bc}	4.21(0.29)^{bc}

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

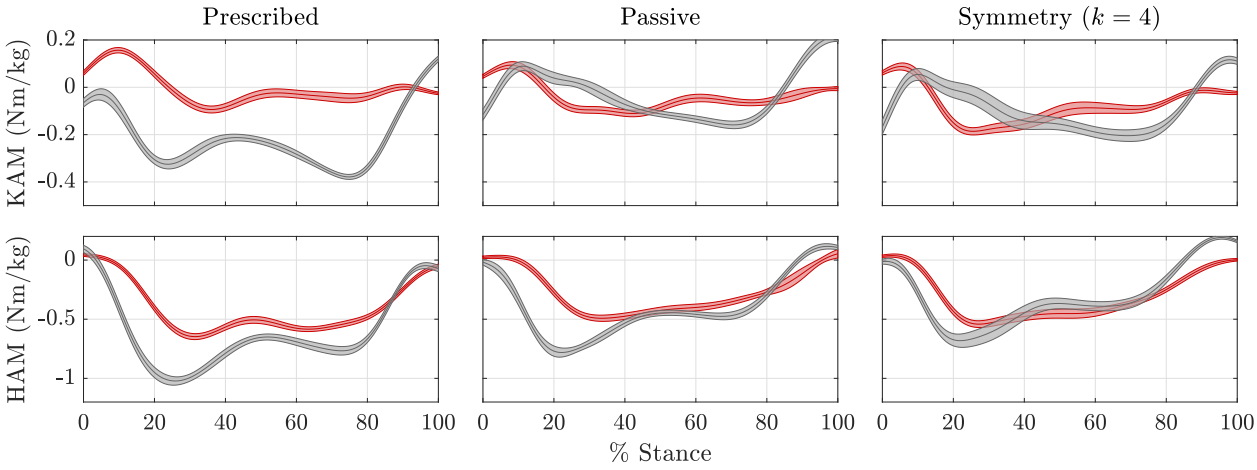


Figure J.11: Experiment 5 frontal plane hip moments (KAM and HAM) for the three experimental conditions.

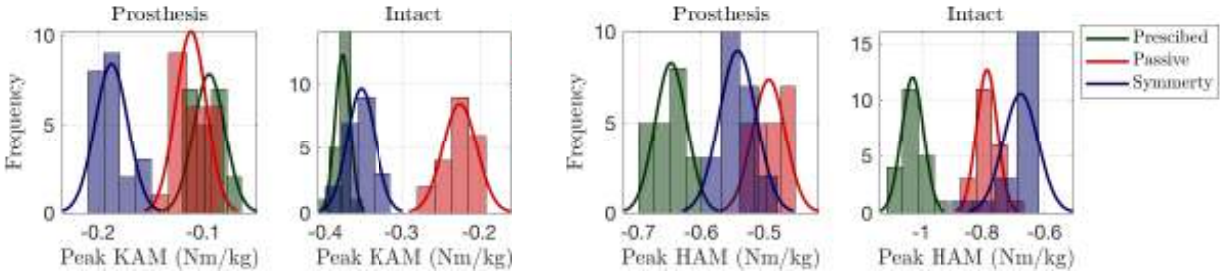


Figure J.12: Experiment 5 histograms for KAM and HAM outcomes.

Table J.6: Experiment 5 OA loading factors outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 4$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Knee Adduction Moment (Nm/kg)	$-0.09(0.02)^{a}$	$-0.38(0.01)$	$-0.11(0.01)^{ab}$	$-0.23(0.02)^{b}$	$-0.19(0.02)^{abc}$	$-0.35(0.02)^{bc}$
Peak Hip Adduction Moment (Nm/kg)	$-0.65(0.03)^{a}$	$-1.03(0.03)$	$-0.49(0.03)^{ab}$	$-0.78(0.03)^{b}$	$-0.54(0.03)^{abc}$	$-0.68(0.06)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

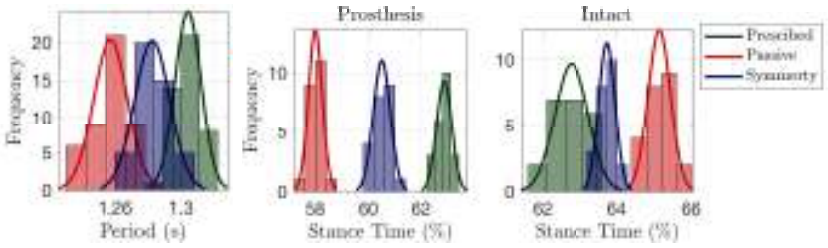


Figure J.13: Experiment 5 histograms for temporal outcomes.

Table J.7: Experiment 5 temporal symmetry outcomes measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 4$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Step Period (s)	1.30(0.01)		1.26(0.01)		1.28(0.01)	
Stance	62.87(0.28)	62.78(0.45)	58.01(0.26)^{ab}	65.15(0.30)^{b}	60.52(0.32)^{abc}	63.74(0.22)^{bc}

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

Appendix K

EXPERIMENT 6 RESULTS

K.1 Summary

Results for experimental 6 are provided below. The results from this experiment again demonstrate the effect of a delay in the control signal. In this case a bug in the code caused a delay, instead of compensating for the actually HS delay. Note that even with the delay still doesn't diverge. Results show marginal improvements in gait mechanics with a delayed control signal.

Table K.1: Algorithm and motor parameters for experiment 6.

Gearbox transmission	R_G	19.2
Effective transmission	R_T	1800
Torque constant (Nm/A)	k_τ	0.00775
Speed constant (rpm/V)	k_s	1223
Initial gain	ρ	1
Pre-smooth time (%)	t_1	4
Post-smooth time (%)	t_2	TO
Max harmonic (rad/s)	ω_{max}	$10 \cdot \omega_0$
Filter cutoff	ω_c	$10 \cdot \omega_0$
HS delay (ms)	Δ_{HS}	-40
Decay rate	γ	2
Allowable reference growth factor	α	1

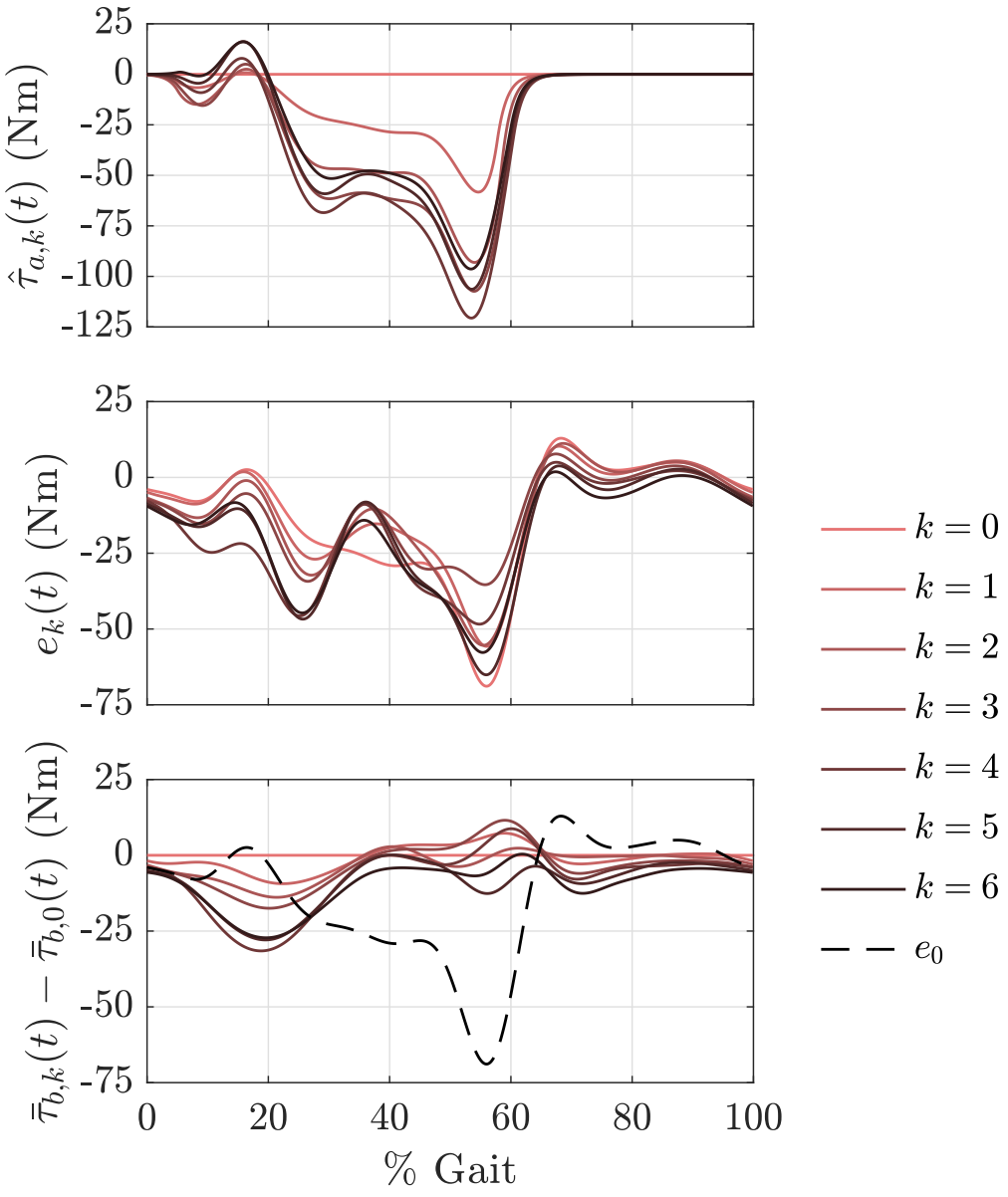


Figure K.1: Experiment 6 time domain evolution of active torque $\hat{\tau}_{a,k}$, error e_k, and the change in reference signal, e.g., $\bar{\tau}_{b,k} - \bar{\tau}_{b,0}$.

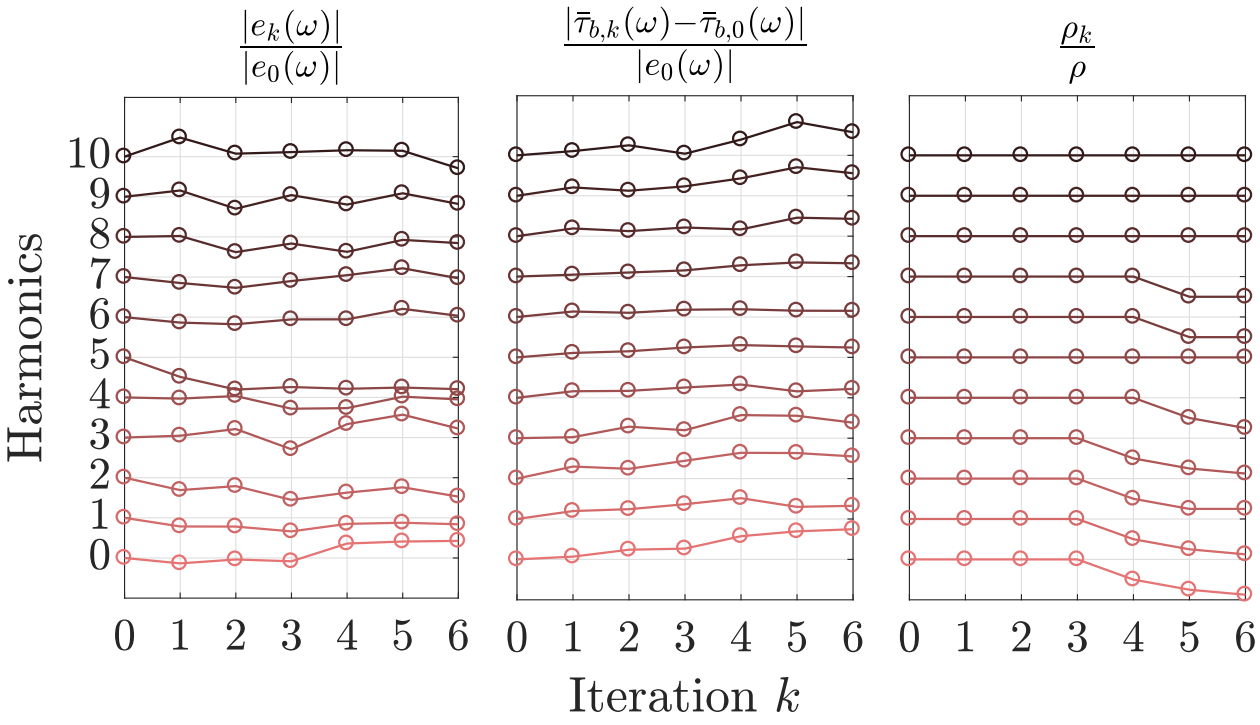


Figure K.2: Experiment 6 frequency domain evolution of the magnitude of error, normalized by the initial error, $|e_k|/|e_0|$, the magnitude of change in reference signal, normalized by the initial error $|\bar{\tau}_{b,k} - \bar{\tau}_{b,0}|/|e_0|$, and adaptation in ρ_k, normalized by the initial ρ_0.

Table K.2: Experiment 6 joint kinematic outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 6$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Plantarflexion (deg)	$1.99(0.19)^{a}$	$-16.66(1.85)$	$1.85(0.11)^{ab}$	$-13.41(1.31)^{b}$	$-12.96(0.92)^{abc}$	$-13.67(1.17)^{b}$
Peak Ankle Dorsiflexion (deg)	$24.90(0.13)^{a}$	$19.52(0.51)$	$22.39(0.18)^{ab}$	$17.96(0.48)^{b}$	$17.20(0.26)^{abc}$	$17.92(0.61)^{b}$
Peak Knee Flexion (deg)	$71.34(0.70)^{a}$	$59.30(1.10)$	$54.99(0.78)^{ab}$	$51.84(0.97)^{b}$	$57.14(0.65)^{abc}$	$52.04(1.06)^{b}$
Peak Hip Extension (deg)	$13.29(0.36)^{a}$	$5.28(0.26)$	$7.39(0.43)^{ab}$	$1.09(0.51)^{b}$	$7.75(0.27)^{abc}$	$2.08(0.41)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

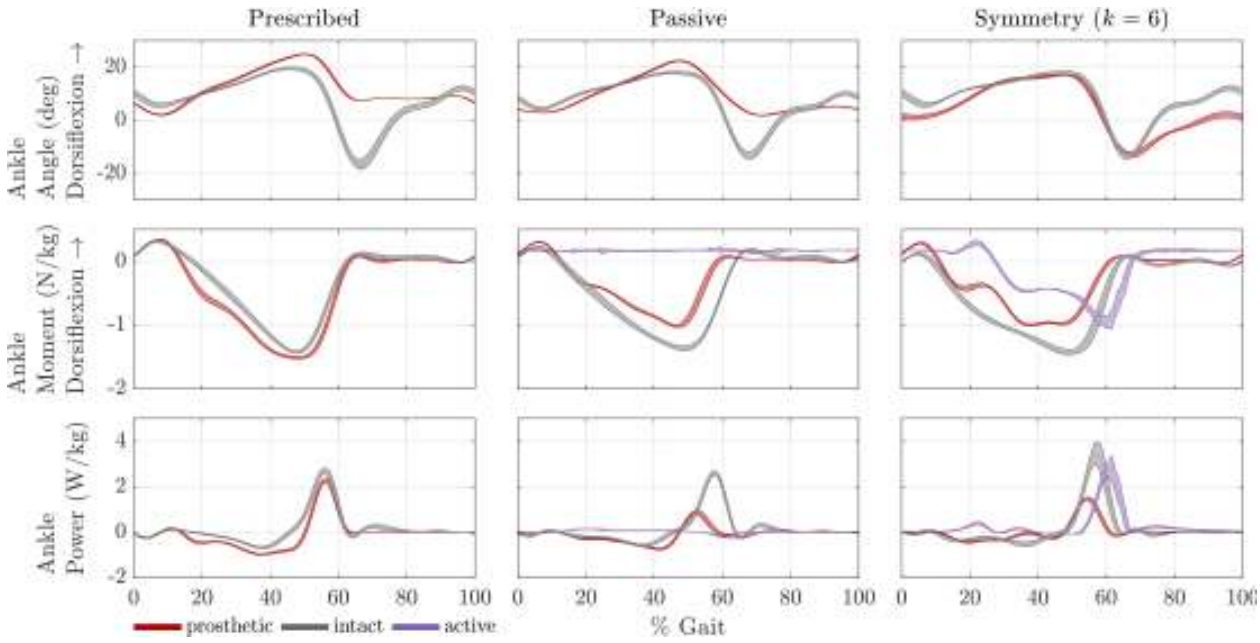


Figure K.3: Experiment 6 sagittal ankle mechanics for the three experimental conditions.

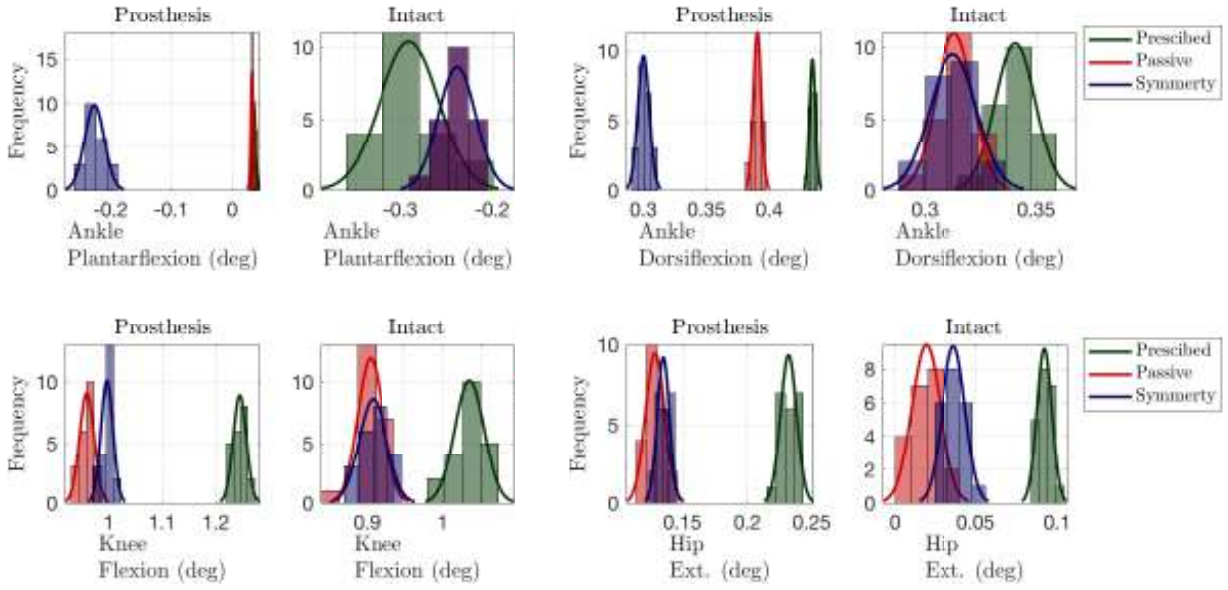


Figure K.4: Experiment 6 histograms for peak kinematic outcomes.

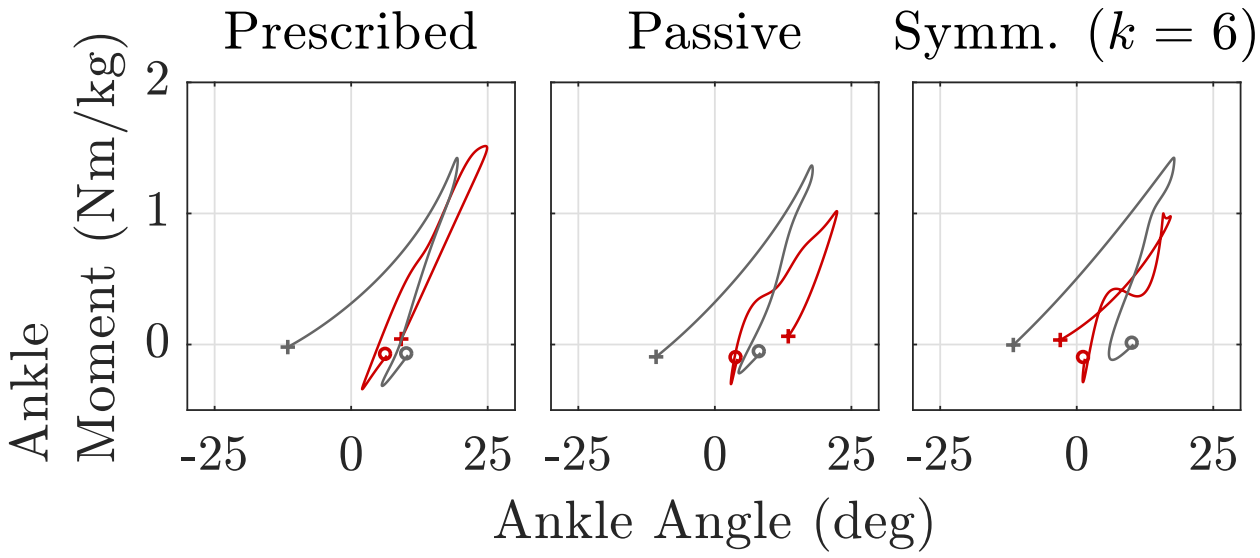


Figure K.5: Experiment 6 ankle work loops for the three experimental conditions.

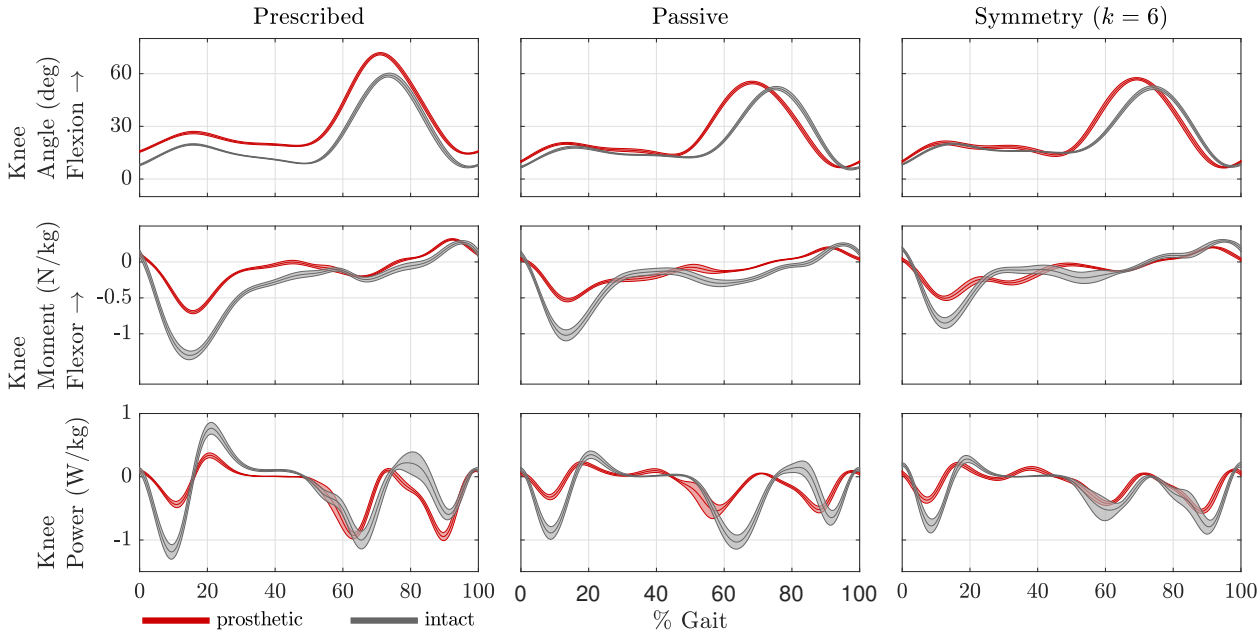


Figure K.6: Experiment 6 sagittal plane knee mechanics for the three experimental conditions.

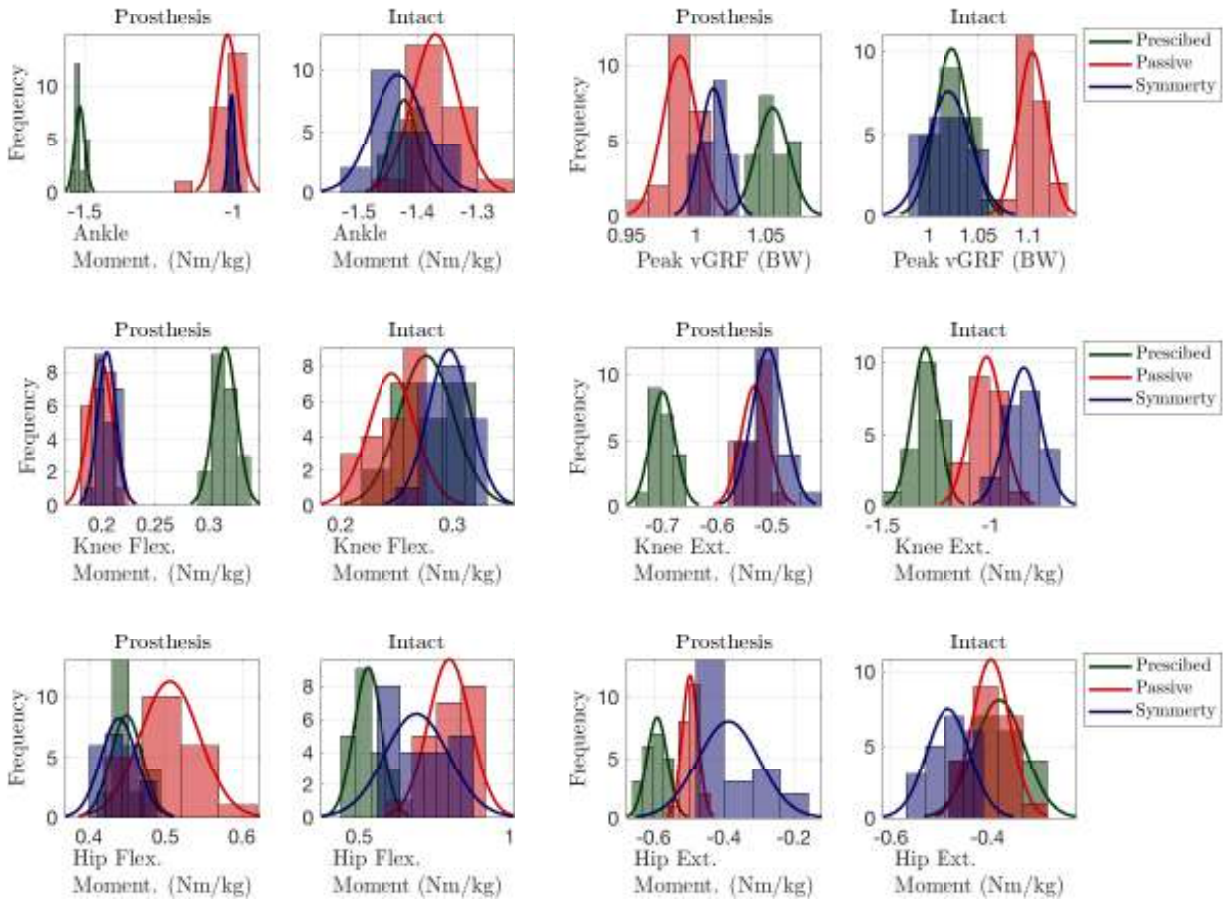


Figure K.7: Experiment 6 histograms for peak kinetic outcomes.

Table K.3: Experiment 6 joint kinetics and vGRF outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 6$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Ankle Moment (Nm/kg)	$-1.52(0.02)^{a}$	$-1.42(0.02)$	$-1.02(0.04)^{ab}$	$-1.37(0.04)^{b}$	$-1.01(0.01)^{ab}$	$-1.43(0.04)^{c}$
Peak Knee Flexion Moment (Nm/kg)	$0.32(0.01)^{a}$	$0.28(0.02)$	$0.20(0.01)^{ab}$	$0.25(0.02)^{b}$	$0.21(0.01)^{ab}$	$0.30(0.02)^{bc}$
Peak Knee Extension Moment (Nm/kg)	$-0.70(0.02)^{a}$	$-1.31(0.06)$	$-0.53(0.03)^{ab}$	$-1.02(0.07)^{b}$	$-0.51(0.03)^{abc}$	$-0.85(0.08)^{bc}$
Peak Hip Flexion Moment (Nm/kg)	$0.45(0.02)^{a}$	$0.53(0.05)$	$0.51(0.04)^{ab}$	$0.80(0.07)^{b}$	$0.44(0.02)^{ac}$	$0.69(0.11)^{bc}$
Peak Hip Extension Moment (Nm/kg)	$-0.59(0.03)^{a}$	$-0.38(0.05)$	$-0.50(0.02)^{ab}$	$-0.39(0.04)$	$-0.39(0.09)^{abc}$	$-0.48(0.04)^{bc}$
Peak vGRF (BW)	$1.05(0.01)^{a}$	$1.02(0.02)$	$0.99(0.01)^{ab}$	$1.10(0.01)^{b}$	$1.01(0.01)^{bc}$	$1.02(0.02)^{c}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

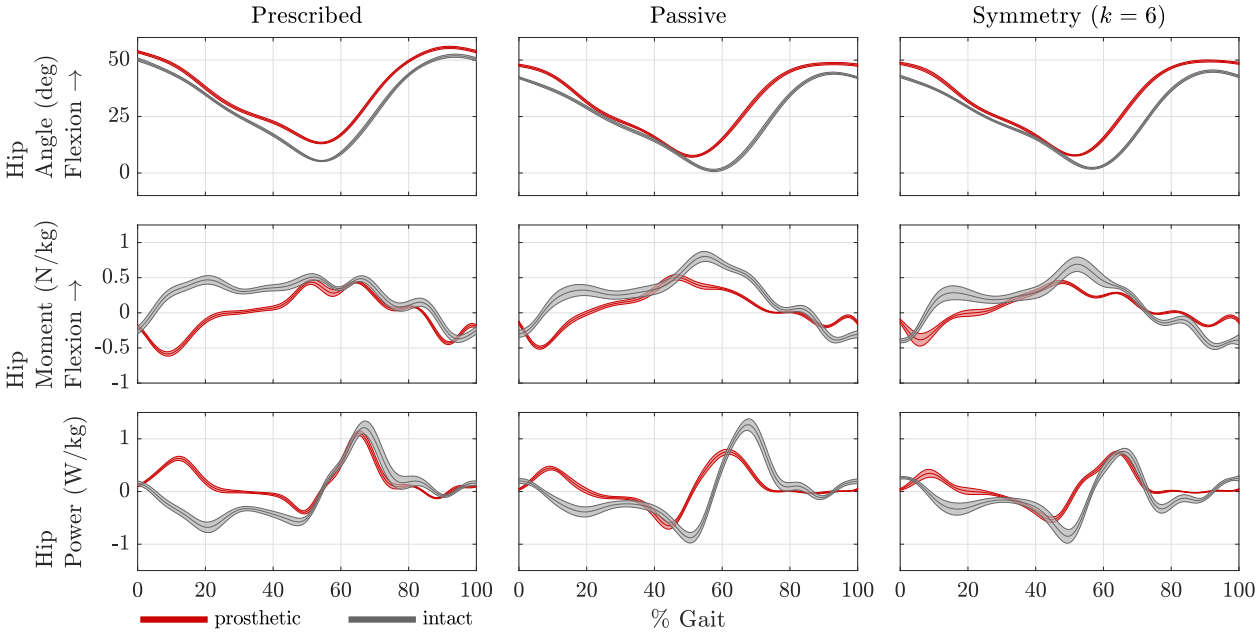


Figure K.8: Experiment 6 sagittal plane hip mechanics for the three experimental conditions.

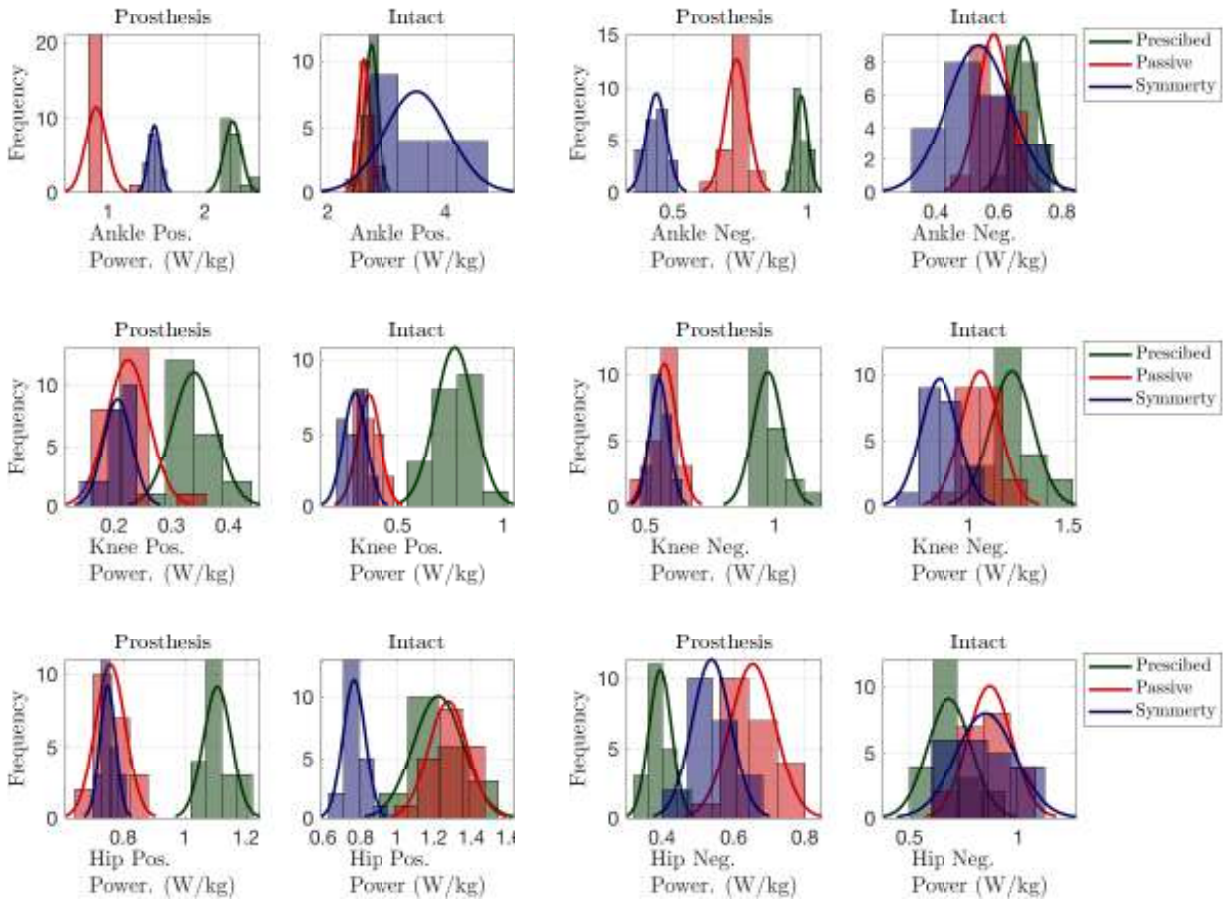


Figure K.9: Experiment 6 histograms for peak power outcomes.

Table K.4: Experiment 6 joint power outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 6$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Positive Ankle Power (W/kg)	$2.29(0.09)^{a}$	2.74(0.11)	$0.88(0.11)^{ab}$	$2.61(0.10)^{b}$	$1.48(0.06)^{abc}$	$3.52(0.54)^{bc}$
Peak Negative Ankle Power (W/kg)	$0.98(0.02)^{a}$	0.68(0.04)	$0.74(0.04)^{ab}$	$0.58(0.05)^{b}$	$0.44(0.04)^{abc}$	$0.54(0.10)^{b}$
Peak Positive Knee Power (W/kg)	$0.34(0.04)^{a}$	0.77(0.09)	$0.22(0.04)^{ab}$	$0.36(0.06)^{b}$	$0.21(0.02)^{ab}$	$0.29(0.05)^{bc}$
Peak Negative Knee Power (W/kg)	$0.97(0.06)^{a}$	1.22(0.11)	$0.57(0.05)^{ab}$	$1.06(0.10)^{b}$	$0.55(0.04)^{ab}$	$0.85(0.10)^{bc}$
Peak Positive Hip Power (W/kg)	$0.34(0.04)^{a}$	0.77(0.09)	$0.22(0.04)^{ab}$	$0.36(0.06)^{b}$	$0.21(0.02)^{ab}$	$0.29(0.05)^{bc}$
Peak Negative Hip Power (W/kg)	$0.97(0.06)^{a}$	1.22(0.11)	$0.57(0.05)^{ab}$	$1.06(0.10)^{b}$	$0.55(0.04)^{ab}$	$0.85(0.10)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

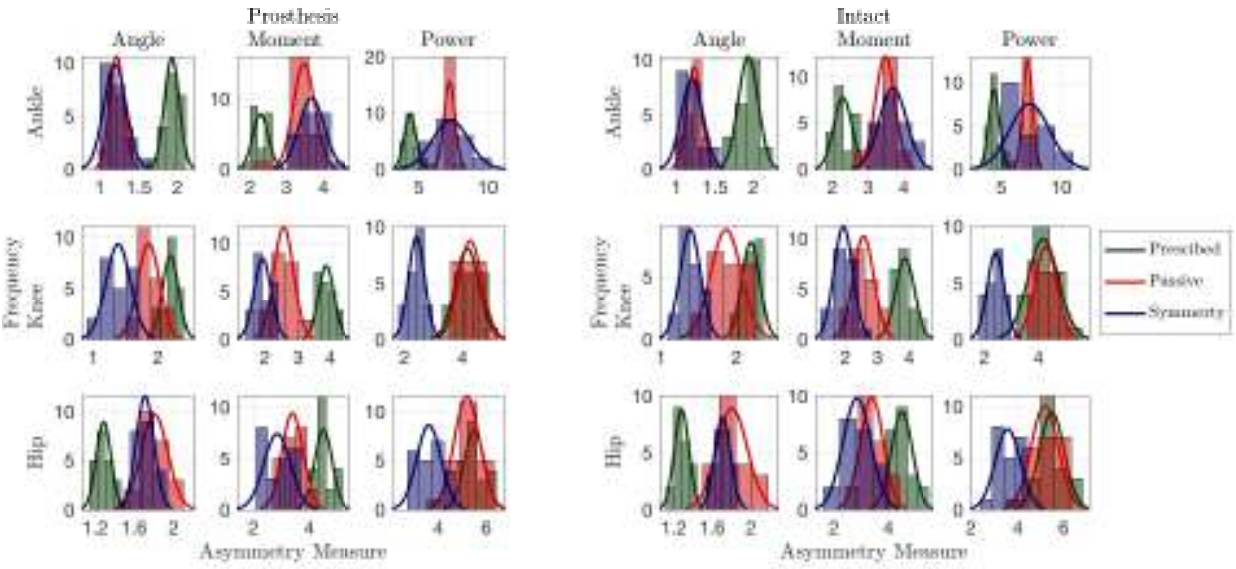


Figure K.10: Experiment 6 histograms for AM outcomes.

Table K.5: Experiment 6 sagittal plane AM outcome measures (standard deviations), defined in (6.2).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 6$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
AM GRF (BW)	2.42(0.37)	2.43(0.38)	5.58(0.25)$^{a\,b}$	6.59(0.40)^{b}	5.34(0.44)$^{b\,c}$	5.51(0.39)$^{b\,c}$
AM Ankle Angle (rad)	1.93(0.10)	1.93(0.13)	1.22(0.09)^{b}	1.23(0.10)^{b}	1.20(0.14)^{b}	1.20(0.14)^{b}
AM Ankle Moment (Nm/kg)	2.29(0.21)	2.28(0.26)	3.48(0.29)^{b}	3.47(0.28)^{b}	3.69(0.35)$^{b\,c}$	3.70(0.38)$^{b\,c}$
AM Ankle Power (W/kg)	4.38(0.43)	4.39(0.46)	7.23(0.40)^{b}	7.22(0.40)^{b}	7.31(1.31)^{b}	7.38(1.66)^{b}
AM Knee Angle (rad)	2.19(0.12)	2.18(0.12)	1.85(0.17)^{b}	1.86(0.20)^{b}	1.38(0.19)$^{b\,c}$	1.39(0.13)$^{b\,c}$
AM Knee Moment (Nm/kg)	3.88(0.24)	3.88(0.30)	2.57(0.30)^{b}	2.56(0.33)^{b}	1.92(0.26)$^{b\,c}$	1.92(0.30)$^{b\,c}$
AM Knee Power (W/kg)	4.16(0.41)	4.18(0.57)	4.27(0.40)	4.28(0.41)	2.42(0.28)$^{b\,c}$	2.42(0.31)$^{b\,c}$
AM Hip Angle (rad)	1.29(0.08)	1.28(0.08)	1.81(0.14)^{b}	1.81(0.16)^{b}	1.71(0.09)$^{b\,c}$	1.71(0.08)$^{b\,c}$
AM Hip Moment (Nm/kg)	4.50(0.31)	4.50(0.39)	3.40(0.37)^{b}	3.39(0.42)^{b}	2.86(0.48)$^{b\,c}$	2.84(0.51)$^{b\,c}$
AM Hip Power (W/kg)	5.50(0.42)	5.49(0.54)	5.19(0.53)^{b}	5.18(0.58)	3.60(0.51)$^{b\,c}$	3.59(0.54)$^{b\,c}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

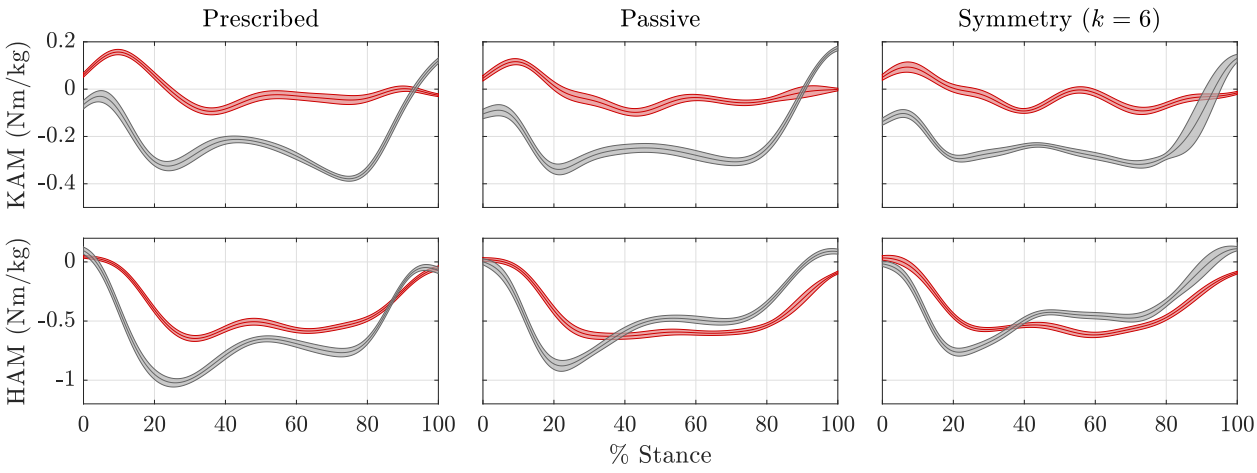


Figure K.11: Experiment 6 frontal plane hip moments (KAM and HAM) for the three experimental conditions.

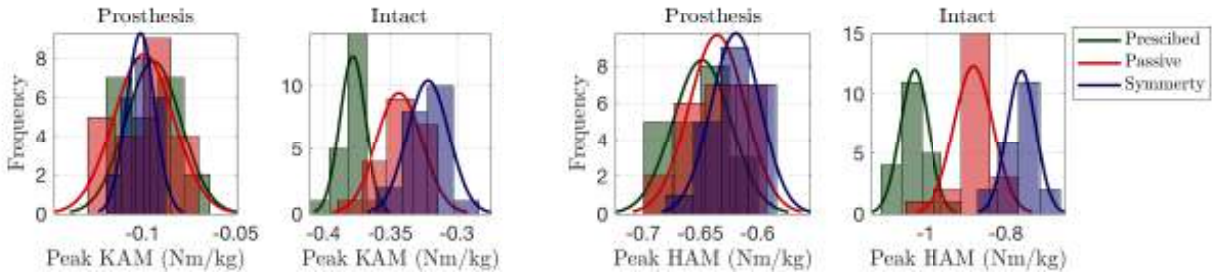


Figure K.12: Experiment 6 histograms for KAM and HAM outcomes.

Table K.6: Experiment 6 OA loading factors outcome measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 6$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Peak Knee Adduction Moment (Nm/kg)	$-0.09(0.02)^{a}$	$-0.38(0.01)$	$-0.10(0.02)^{a}$	$-0.34(0.02)^{b}$	$-0.10(0.01)^{a}$	$-0.32(0.02)^{bc}$
Peak Hip Adduction Moment (Nm/kg)	$-0.65(0.03)^{a}$	$-1.03(0.03)$	$-0.64(0.02)^{a}$	$-0.88(0.05)^{b}$	$-0.62(0.02)^{abc}$	$-0.76(0.04)^{bc}$

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.

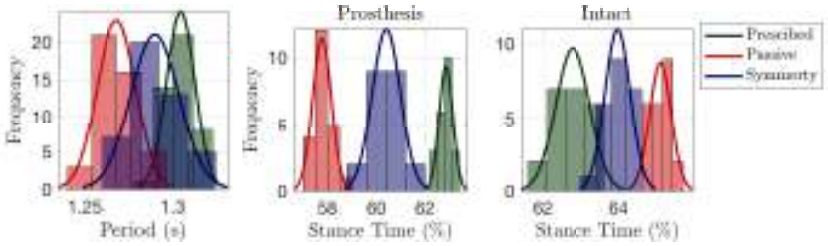


Figure K.13: Experiment 6 histograms for temporal outcomes.

Table K.7: Experiment 6 temporal symmetry outcomes measures (standard deviations).

	Prescribed		Passive Mode ($k = 0$)		Symm. Control ($k = 6$)	
	Prosthesis	Intact	Prosthesis	Intact	Prosthesis	Intact
Step Period (s)	1.30(0.01)		1.27(0.01)		1.29(0.01)	
Stance (%)	62.87(0.28)	62.78(0.45)	57.78(0.38)^{ab}	65.09(0.28)^{b}	60.42(0.58)^{abc}	63.97(0.33)^{bc}

[a] Indicates significant difference between limbs within the same condition.

[b] Indicates significant difference when compared with the prescribed condition for the same limb.

[c] Indicates significant difference when compared with the passive mode condition for the same limb.