SINGLE-AXIS SIMULATED HIP-JOINT MOVEMENT CONTROLLED BY BODY-MOUNTED IMUS FOR ELECTROMECHANICAL PROSTHESIS APPLICATIONS

by

Austin Lawrence Fifield

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Applied Energy and Electromechanical Systems

Charlotte

2020

Approved by:

_____
Dr. Aidan Browne

_____
Dr. Luke Donovan

_____
Dr. Michael Smith

_____
Dr. Wesley Williams

©2020
Austin Lawrence Fifield
ALL RIGHTS RESERVED

ABSTRACT

AUSTIN LAWRENCE FIFIELD. Single-axis simulated hip-joint movement controlled by body-mounted IMUs for electromechanical prosthesis applications. (Under the direction of DR. AIDAN BROWNE)

Survivors of cancerous and traumatic transpelvic amputations have difficulty in transitioning into and out of a chair, as well as sitting upright with correct posture. The addition of special pads and chair inserts can help the amputee; however, through time, the amputee may develop spinal deformities, which pose a major health concern. In an effort to improve the quality of life of transpelvic amputees, a representative single-axis hip-joint prosthesis analogue is designed and tested. A motor, loaded with mass characteristics similar to that of a plausible thigh prosthesis, performs bidirectional motion; it is commanded by an embedded controller that calculates position commands from two body-mounted IMUs. Body acceleration data is obtained from the IMUs, and real-time calculations are performed to generate positional commands for the hip-joint motor. A 3-D motion camera system confirms that digital data obtained from system measurement devices accurately corresponds to the physical position and orientation of the body and representative prosthesis. A separate computer is used to simultaneously log raw data, generate data analysis visuals, and produce system stability and response characteristics. After completing a thorough analysis of the system's operation and response, the prosthesis analogue produces motion and control characteristics feasible for autonomous electromechanical prosthesis motion. The study, as a whole, successfully presented a control system that measures data accurately from body-mounted IMUs and controls a closed-feedback, single-axis, rotational device, representative of a plausible electromechanical thigh-joint prosthesis. With further research towards a complete prosthesis prototype, system design, and control characteristics of this study could be integrated and used in the advancement of the quality of life of many transpelvic amputation survivors.

DEDICATION

This thesis is dedicated to my twin brother, Adam. Your courage, motivation, and strength are unmatched and wholly admired. I am so thankful you fought and made it through that fateful night in July and am optimistic for your future. This thesis took me a lot of time to complete. Could you take just a little time to read it?

ACKNOWLEDGEMENTS

It is absolutely imperative that I acknowledge the guidance, help, and criticism from Dr. Aidan Browne. Dr. Browne spent many hours during a time of international crisis to ensure all necessary resources were available to me in order to complete this thesis. His efforts were completely voluntary and very much appreciated as the completion of this thesis would not be possible without his help.

I'd like to send out a special thank you to Dr. Luke Donovan who confidently reached into a new field of study and helped apply kinesiology to engineering design.

I would also like to acknowledge the exceptional guidance and reliability of Dr. Michael Smith, and Dr. Wesley Williams during an especially complex time in history. Their expertise was very much appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3-D	3-Dimensional
CAD	Computer Aided Design
DHPF	Digital High Pass Filter
DIO	Digital Input/Output
DLPF	Digital Low Pass Filter
DoF	Degrees of Freedom
EMG	Electromyography
FPE	Final Prediction Error
FPS	Frames per second
I^2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
MEMS	Micro-electromechanical System
MRAC	Model Reference Adaptive Control
MSE	Mean Root Square
MXP	myRIO Expansion Port
n4sid	Subspace state-space estimation
NI	National Instruments
OPC	Open Platform Communications
PD	Proportional-Derivative

PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation
RPM	Revolutions per minute
SPI	Serial Peripheral Interface

CHAPTER 1: INTRODUCTION

1.1 Background

In recent years, upper-limb prosthesis research has advanced quickly, moving towards total limb control through myoelectric sensing. Lower-limb prosthetics, specifically below-knee, have seen improved research in variable dampers, complex joints and springs, and even microprocessor control. As the quality of life for most amputees has increased through novel prosthetic research, amputees with rare conditions, such as forequarter and transpelvic (hemipelvectomy) amputations, have been left with outdated prosthetic devices that do not contain modern dynamic-control mechanisms. Newer upper-limb prosthetics have been designed with electromyography and microprocessors, specifically for finger and hand motion. More advanced lower-limb prosthetics have utilized springs, variable dampers, and hydraulics. Most microprocessor-controlled prosthesis research and development has been targeted towards upper-limb devices. This research aims to produce a microprocessor-controlled model hip prosthesis with single-axis sagittal plane motion controlled via an embedded system for application in hemipelvectomy amputees.

The transpelvic amputation is a procedure in which the pelvis and surrounding muscle, tissue, nerves, and bone have been medically or traumatically amputated. A similar but different procedure called the hip disarticulation involves amputation at the hip joint. During these operations, the amputee retains their entire pelvis but loses all three (ankle, knee, and hip) lower-limb joints on the affected side of the body. This research aims at providing a prosthesis for transpelvic amputees but could also aid in the development of a similar device for hip disarticulation amputees. This hemipelvectomy is typically performed medically in response to a cancerous tumor in the pelvic region or traumatically due to high-speed trauma in the same region. Referencing Figure 1.1, amputation of the pelvis includes removal of all or

part of the acetabulum, ilium, pubis, and ischium and may also include small sections of the sacrum, gluteal, and other surrounding muscles. Hemipelvectomy amputees, more often than not, have injuries to include the bladder, bowel, and sexual organs, atypical of hip disarticulation amputees [5]. The hemipelvectomy is the last resort for lower-limb amputations and requires considerably more rehabilitation than other lower-limb amputations. The first recorded hemipelvectomy was performed in 1891, with the patient only living hours after the operation [6]. By 1902, 13 recorded hemipelvectomy operations had been performed with a 60% mortality rate [7] [8]. By 2013, the survival rate increased to 66%, with a mean postoperative lifespan of 32.8 months [9]. As the transpelvic amputation occurs more frequently with a higher survival rate and postoperative lifespan, a more significant focus in the research of lower-limb prosthetics should turn to the quality of life of hemipelvectomy survivors.

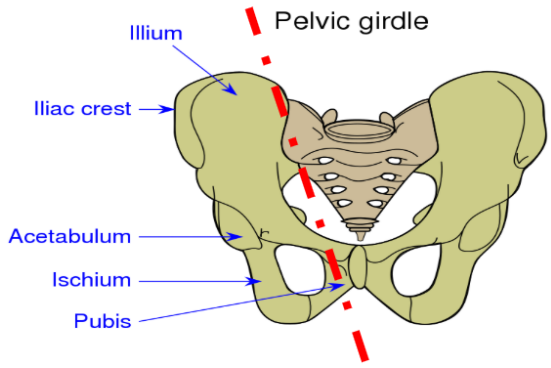


Figure 1.1: Possible separation division of transpelvic amputation.

1.2 Problem Statement

For hemipelvectomy amputees, sitting down with correct, balanced posture, transitioning to standing and sitting positions, and reclining in a chair are all difficult tasks. With time, curvature of the spine (scoliosis), or other spinal deformities will form and could cause medical problems for the amputee. To prevent this, the amputee must sit upright with the amputated side supported; this can be done with a fixed prosthesis. For the hip joint to assist with reclining and moving comfortably in a chair, it

must be dynamic. A rigid thigh is required to add support as well as help guide the amputee into a seat. Ideally, an amputee should have the ability to sit, relax, and reposition intuitively; however, with current lower-limb prosthesis development, this is not possible. The technology exists to create advanced lower-limb prosthetics as seen in common, complex upper-limb prosthetics; however, research and development in connecting the technologies and building advanced artificial lower limbs is behind. There is currently a need for dynamically controlled hip-joint prosthetics and with the survival rates of hemipelvectomy patients increasing, there will be a much greater need for such prosthetics in the future [7] [8] [9].

1.3 Conceptual Design

A single-axis hip-joint prosthesis would need to be designed with rotational ability across the sagittal plane, most easily performed by a motor or hydraulic system. Using Figure 1.2 for reference, motion across the sagittal plane involves the extension and flexion of the lower limbs.

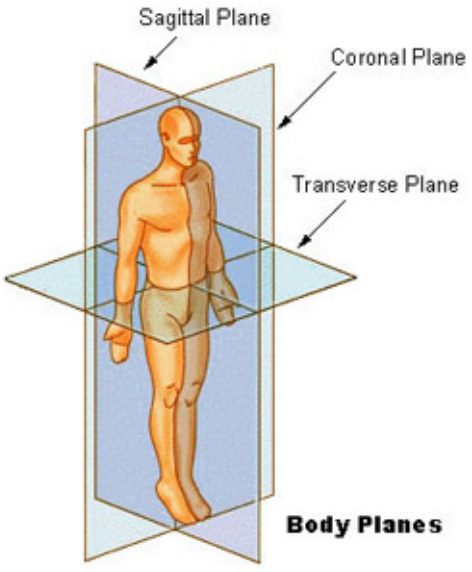


Figure 1.2: Anatomical planes of the human body [1]

The use of a stepper motor or large servo motor could be affordable and ideal for joint control, although more advanced and expensive servo-hydraulic systems such as

those used by Boston Dynamics in their humanoid robot, Atlas, have been proven reliable and very capable [10]. The rotational portion of the prosthesis (the artificial thigh) would house all components with the motor located towards the location of rotation, most likely fitted along the length of the artificial thigh with a bevel gear, or 90 degrees gearbox attached such that a shaft can be connected to the static, body-mounted portion of the prosthesis. The prosthesis would operate without power cords and through the use of a chargeable battery pack housed in the artificial limb. The artificial thigh should allow for socket assembly (static portion of the prosthesis) interchangeability as prosthesis sockets are unique and custom-fitted to every amputee. Transpelvic amputees vary significantly in shape and size, especially at the amputation site. The artificial thigh should be attachable and operational with all unique socket variants. A robust and structural rigid thigh should be constructed to protect control components and be able to support the weight of the individual utilizing the prosthesis. Prototype designs can use wired communication devices located on the non-amputated side of the body to determine limb and body orientation; however, wireless devices would be ideal in eliminating the need for wires to be run across the body to the prosthesis. Pairing this single-axis prosthesis with an above-knee 2-axis prosthesis would be the direct final research target. A dynamic and autonomously controlled 3-axis prosthesis could potentially solve many problems lower-limb amputees face.

This thesis focuses on control system aspects of the conceptual design as well as testing an autonomously controlled model representative of a thigh prosthesis that can enable advanced design and construction of a producible prosthesis in future work. Materials, form, and aesthetic properties of the prosthesis are not examined in this study and are left for future research and development.

CHAPTER 2: LITERATURE REVIEW

2.1 Human Motion and Prosthesis Analysis

The analysis of human motion aided with lower-limb prosthetics is well documented at and below the knee. Individuals with rare conditions such as trans-femoral and transpelvic amputations lack extensive gait analysis in part because most of these amputees choose not to wear a prosthesis [11]. The rarity of these conditions has left lower-limb prosthetics, particularly triple-joint, under-researched. The triple-joint mechanical prosthesis seen in Figure 2.1 is a typical prosthesis used by transpelvic amputees that want to maintain a lifestyle of self-sufficient locomotion.

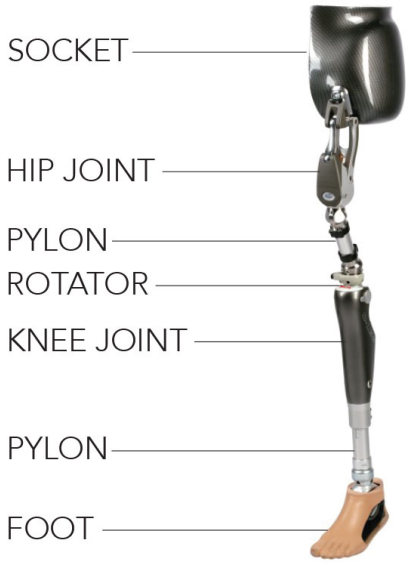


Figure 2.1: Typical triple joint lower-limb prosthesis [2]

Though this prosthesis has many advantages, it also has some downfalls, particularly when amputees want to comfortably sit in deep chairs but want to keep the prosthesis attached to their bodies. The artificial buttocks is static in size and shape and does not conform to thigh motion. The mechanical thigh joint allows a portion of the prosthesis to move up and out of the way when taking a seat; however, whenever the amputee is sitting in a deep chair or sofa, the leg portion of the prosthesis will

begin to contact the front crest of the seat. A dynamic motion controlling the entire length of an artificial thigh, as well as cooperation with the artificial buttocks portion of the prosthesis, is necessary and in need of research.

A study performed in 2014 by M.T. Karimi, et al., analyzed gait characteristics of a hemipelvectomy subject walking with a triple-joint Canadian prosthesis, as seen in Figure 2.2 [12].

Figure 2.2: Triple joint Canadian lower-limb prosthesis [2]

The subject's prosthesis included mechanical-only joints with electronic control devices. For this study, the subject walked with his prosthesis while the "spatiotemporal gait parameters, moments applied on the lower-limb joints, three planar motions of the lower-limb joints and trunk, and force applied on the legs during walking" were examined [12]. The study concluded that the subject's gait characteristics of the mechanical prosthesis were vastly different and asymmetrical to the gait characteristics of the patient's non-amputated leg. Though the triple-joint prosthesis allowed the subject to walk, the motion created by the subject was asymmetrical to expected normal movements; the range of motion was also significantly reduced in the prosthesis side. Range of motion in the pelvis showed no significant difference to the

non-prosthesis pelvic side due to the entire pelvis range of motion being restricted by the prosthesis.

Another study, performed by Morgan Redfield, et al., recorded data from transtibial amputees' motion during specific activities to better understand the effects of prosthetic devices [13]. Accelerometers were attached to the prosthetic limb of ten transtibial amputees, and limb acceleration data was obtained during walking, stair climbing, standing, and sitting. One accelerometer was placed on the subject's thigh, while another accelerometer was placed on the prosthesis near the ankle. Human pose phase was determined for each subject in the range of 90.1% - 99.6%. The study concluded that the use of two accelerometers mounted as such in the study produced equal to better results than a single mounted accelerometer. This research is significant for future projects that aim at identifying lower-limb acceleration and movement. An IMU attached to the rigid body between joints helps identify motion characteristics and should not be overlooked.

When measuring body motion using IMUs, long term measurements have continuously added error through accelerometer and gyroscope drift. Some studies have aimed to identify superior filtering algorithms for maximum accuracy. The Kalman filter is widely known to reduce drift in motion tracking devices. Some researchers have experimented with IMU angle estimation without the use of a magnetometer to help produce accurate heading values in magnetized environments [14]. In this study, Jung Keun Lee, et al. used a new Kalman filter that takes data from an IMU gyroscope, corrects the data with kinematic constraints, and estimates yaw heading to establish an accurate joint angle. An optical camera system was used to verify test results against the calculated positional data. As with this experiment, the thesis performed in this document also utilizes an optical camera system to validate test data. Without a stand-alone system to validate data, such as a separate motion capture system, motion characteristics identified from IMU data cannot be trusted

especially with the knowledge that IMUs drift. A 2012 study, performed by George Thomas and Daniel Simon, used a Kalman filter along with an IMU to estimate thigh angle for control of a semi-active knee prosthesis [15]. A human subject, wearing the IMU, walked five paces, performed a 90-degree turn, and walked five more paces. Upon examination, Thomas determined that the thigh angle of the leg containing the prosthesis heavily depends on the knee when standing (stance phase). While walking (swing phase), the thigh angle is fully controlled by the user, and the prosthesis has little effect on this angle. The Kalman filter helped remove IMU data error while performing the pivot. This study concluded that the Kalman filter is a useful tool in determining thigh angle from IMU data for control of a prosthetic leg. Clement Duraffourg, et al. investigated a possible algorithm that can be used to determine the pose of a prosthesis, an important part of gait mode recognition [16]. This algorithm used Kalman filter characteristics as well as other drift correction methods to obtain consistent velocity data. The output of this algorithm is compared against optoelectronic measurements of a transfemoral amputee. Kalman-based algorithms are well known for their reliability and success in IMU data estimation. The study concluded this algorithm is useful in determining gait mode as well as prosthesis control; however, the formula may not be optimal due to the computed starting point being that of when the heel strikes the ground. This produces poor data due to the integration drift of the first cycle. A study performed in 2018 by Hamza Benzerrouk and Alexander Nebylov, used modern filtering algorithms instead of the standard Kalman filter to determine the precise attitude, position, and direction of pedestrian navigation [17]. The study focused on indoor walking motion, specifically the detection of foot stance phase, and did not aim to produce any data on body motion above the ankle joint.

Some studies, such as one performed by Guanglin Li, et al., have tested body position and motion with non-IMU approaches, such as this study, which experimented with the arm angle of an amputee [18]. A small magnet was placed into the end of a

simulated residual arm bone. In theory, the magnet's magnetic field can be measured, thus generating a calculated and precise arm angle. This method generated data that is consistent with human bone structure, thus creating accurate data not manipulated by muscle, skin deformities, or drifts in IMU data. Magnetic flux, as well as magnetic field direction, were used along with the finite-element method to calculate arm rotational angle. The study concluded that this method can be reliable when the residual limb is placed in the center of the prosthetic limb. If the limb and magnet are placed off-center by at least 15 mm, data inaccuracies are possible with shifts in both the y and z directions. This study produced results that may be beneficial for above-knee amputees; however, amputees with transpelvic amputations do not have any residual bone left and are unable to control residual hip joint motion.

2.2 Prosthesis Improvement Designs

The supermajority of lower-limb prosthetics utilizes mechanical linkages, springs, dampers, and complex combinations thereof. M. Pina Aragon, et al. evaluated a prototype for a hip prosthesis for applications in hip disarticulation or hemipelvectomy patients [3]. Using gait characteristics of a non-amputee, a possible hip prosthesis was designed using Computer-Aided Design (CAD); it can be viewed below in Figure 2.3.

Figure 2.3: CAD of a possible hip joint prosthesis [3]

The design's CAD model was strength tested in simulation and its operation compared to that of typical gait movements. A polycentric prosthesis was designed similar to a four-link knee mechanism with pneumatic control and operation at 6 km/h simulated walking speed. The study concluded that to implement a prosthesis that mirrors human movement, the devices must respond to the movement of other joints associated with the prosthesis. It was recommended to develop a prosthesis that includes "automatic actuators, or actuators controlled by automatic systems, capable of making instant decisions." This is where the bridge between robotics and microprocessor-controlled prosthetics come together. This thesis aims at autonomously controlling an actuator that can perform movements and make decisions much quicker than a human.

Mechanical characteristics are important aspects when designing a prosthesis. In 2017, Marco Cempini, et al. designed and tested a prosthesis that minimizes size and weight while maintaining power and torque for a polycentric ankle prosthesis application [19]. The design included a magnetic angular encoder that measures the angle of inclination, semi-autonomously actuating the ankle joint to help the user walk while

also reducing the amount of power required to actuate the prosthesis. The prosthesis moved on a standard 4-bar joint mechanism common in a knee prosthesis. A human subject's ankle positions were analyzed and compared to that of the prosthesis for testing to include standing and walking exercises. The test data showed a very similar movement of the prosthesis compared to the human ankle. The study concluded that this particular prosthesis displayed accurate regulation of torque while also performing very similar movement to an actual human ankle. As the location of the prosthesis moves up along the body, its size and weight increase, causing larger loads and higher necessary torques. Thus, optimization in joint movement is essential when designing a multi-joint prosthesis. Another study, performed by Matthew Ryder and Frank Sup, aimed to provide hip-joint torque measurements with the use of an upgraded exoskeleton device [20]. The device improves hip sagittal plane movements utilizing a DC motor and Scotch-Yoke mechanism. Hip torque, angle, and power were mapped to analyze typical hip movements during the gait cycle. Maximum torque was observed at nearly 1.3 Nm/kg normalized to subject mass in kilograms with the hip at approximately 12 degrees flexion. In this design, utilization of a scotch-yoke mechanism allowed the rotary motion to be converted into linear motion, similar to that of an engine crank. This approach allowed the DC motor driving the mechanism to maintain a forward rotating angle throughout the entire gait cycle. This design, along with an actuated spring and gear reduction, allows for smoother operation of the leg and fewer peaks in the required torque curves. This design also produced a gradual increase in motor angle with positive and negative changes in the required wheel angle.

Moving towards electronically controlled prosthesis, such as the one shown in Figure 2.4, Srinivas Pandit, et al. proposed a design of a transfemoral prosthesis to which is controlled by electronic controls and a passive damping system [4].

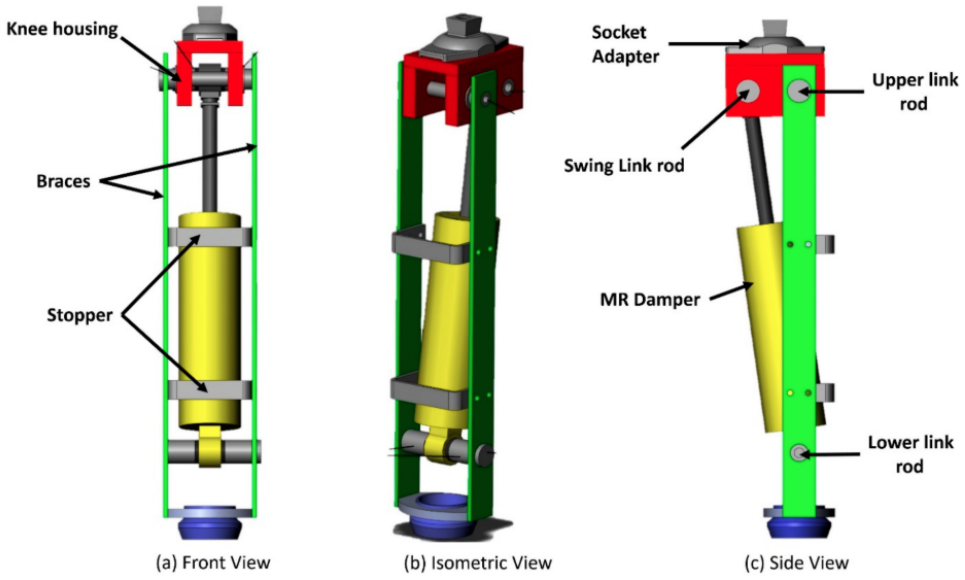


Figure 2.4: Electronic below-knee prosthesis [4]

For testing of this prosthesis, sensors were embedded on the plantar insole of the foot to achieve gait position data for prosthesis control. The prosthesis would need to provide stance phase stability as well as flexion and extension of the limb during the swing phase. A magneto-rheological damper and hinge knee-joint were included in the prosthesis to ensure controlled movement was obtained. A modular magnetic field was used to allow variable control of the damper. For plantar sensing, 24 sensors were located in the insole in five distinct groups. Actuation of these sensors created detailed data for gait cycle assumption. Proportional-integral-derivative (PID) control and filtration was achieved to control magneto-rheological variable damping. A unilateral transfemoral amputee tested the device and performed simple walking tasks. The study concluded that stance and swing gait phase detection was reliably possible with the use of insole embedded sensors inside of a low-cost prosthesis. The addition of sensors and a processor to control autonomous damping with PID control was very successful in this study and is a primary reason for its use in the project performed in this thesis.

2.3 Prosthesis Control

As previously noted, there are many different ways to control a prosthesis. Research has advanced significantly in upper-limb prosthesis control which, as a side effect, has pushed lower-limb prosthesis research further behind. Some researchers have used a combination of electromyography (EMG) and kinematic signals to control a prosthesis replicating a possible solution to transhumeral amputees [21]. A time-delayed artificial neural network was trained to predict elbow and forearm movements. The neural network used data from six proximal EMG signals as well as linear acceleration and angular velocity signal data from three IMUs. Subjects were to perform reaching movements with their arms. These movements were then transcribed into a virtual reality environment. The real-life movements were compared to movements created in the virtual reality space. The test results showed task efficiency in 78% of subjects and concluded that using a combination of EMG and kinematic signals can provide sufficient prosthesis control. This study used EMG signals from subjects with no amputated limbs, thus creating the possibility of inaccurate data when compared to EMG signals of amputees. A requirement for this research is that there needs to be some amount of controlled muscle left around the amputated limb. With some transpelvic amputations, all muscles used to control the hip joint and thigh have been removed, thus disallowing the use of a similar EMG controlled prosthesis. It is also important to realize the effects of surrounding muscle action during specific movements of the hip. Donald Nuemann describes the muscular actions of the hip to include detailed information on local hip muscle groups and their actions in all three anatomical planes [22]. Sagittal plane muscle action is a focus of this research. Muscle groups include the hip flexors and extensors. These muscles are attached to the pelvis via various tendons, all of which can be lost in a transpelvic amputation. Bilateral movements of the flexors produce rotational movements of the femur, generating multi-axis movements. These movements are created from the ball-in-socket hip joint

design of the human body, a feature not implementable in single-axis artificial hip design.

Lower-limb microcontrollers based prosthetics have also been designed and analyzed. One study analyzed the effectiveness of a microcontroller-based pneumatic knee prosthesis [23]. The study, performed by F. Pelisse, et al., included embedding a microcontroller inside of a knee prosthesis that measures stance phase duration and then adjusts a pneumatic damper to increase patient comfort. An amputee who was equipped with the device had their gait phase duration in stance and swing phases measured. The device was deemed advantageous but required additional research and development.

In 2019 using MATLAB, Fermin Aragon, et al. analyzed three prosthesis models (PID, model reference adaptive control, and sliding mode control) for lower-limb prosthetic control [24]. All three systems were modeled in MATLAB, and their stability was analyzed. System torque was measured against time for each system. The PID system showed consistent torque across the entire time interval. The Model Reference Adaptive Control (MRAC) system displayed some initial overshoot but increased torque compared to the PID system. The Sliding Mode Control (SMC) system produced far less overshoot and a more steady increase in torque but overall produced less torque than the MRAC system. The study concluded that the use of non-linear control systems (MRAC and SMC) generated more torque but also some unwanted side effects. The combination of these control systems would ultimately produce ideal results. When designing a simple single control type prosthesis, utilizing a PID controller seems more advantageous as it contains less overshoot and more consistent torque. These are ideal characteristics for a prosthetic control system.

A powered prosthetic ankle joint with dynamic capabilities that allowed the user to walk as well as run was designed [25]. This is much different from typical prosthetic ankle joints as they are typically designed for either walking or running. The

design by Martin Grimmer, et al., used a 200 W DC brushless motor as well as a spring for joint stiffness. A microcontroller was implemented to control the motor. Two gyroscopes were used to measure shank velocity. Accelerometers were used to determine walking direction (forward or backward). These sensors measured single-axis motion in the sagittal plane (flexion and extension). Raw signals were filtered to determine gait stance and behavior. Human subjects were used to measure standard ankle joint torque and angles. This data was used for movement of the prosthetic ankle joint where motion was compared to that of a healthy subject. The human subject's healthy ankle joint was bypassed with an orthotics device while the prosthetic joint walked for them. It was determined that the ankle joint angle and torque values were similar to that of a healthy subject's data. Stride length data when walking was slightly high compared to actual, and torque and speed data were identical to actual. When running, stride length was much shorter than actual. The prosthetic ankle joint created dissimilar results when running and produced data not consistent with actual movement characteristics. The study concluded that this prosthetic model worked well when walking (1.6 m/s) and running slowly (2.6 m/s) but when running faster (4.0 m/s), the model produced unwanted results. Complexity is exponential when adding multiple joints to a prosthetic. An ideal prosthesis should be manufactured to allow the user to perform tasks as well as they might with no amputation. As seen through research, the gait cycle produces many complicated movements that are difficult to replicate even with an advanced prosthesis. In another study, Mevlut Saracoglu, et al. designed an above-knee prosthesis that is controlled by an embedded microcontroller that moves an actuator-controlled knee joint [26] for walking. The microcontroller used input data from gait sensors to determine data when the individual is walking; an MPU-6050 Inertial Measurement Unit (IMU) was used to capture this data. The test prosthesis was fitted to a single test subject, and data was obtained. It was concluded that the device, after many revisions, can reliably obtain data and

send it wirelessly to a microcontroller for data processing. Torque, hip angle, and knee angle were all determined using the IMU data obtained wirelessly. This study concluded the reliability of a wireless data acquisition system for prosthesis applications in individuals performing simple walking tasks. Even in much simpler scenarios such as the previously mentioned study, data obtained from IMUs has been reliable and proven to produce accurate data; however, the fundamental control system that creates artificial movement is lacking.

In the early 1990s, some prosthetics were designed with the use of hybrid prosthetic control systems that utilize electronic components [27]. Externally powered prosthetics can be compared to that of hand-operated bicycle brake systems, electric-powered windows, lighting systems, power tools, flight controls, and heating, ventilation, and air conditioning systems. When integrating these operations into prosthetics, there must be subconscious control, be user friendly, have multifunctional control, instantaneous response, no sacrifice of human function, and maintain a natural appearance. Biomechanical input, as well as transducers, can be used to generate position, velocity, force, and locking and unlocking of the joint. Some control mechanisms include electromyography, myoacoustic signals, and neuroelectric. Though cable-driven prosthetics are well known, these advanced control mechanisms can advance the outcome of prosthetics and bionics. Electronically-controlled artificial limbs create a subconscious control mechanism lacking in cable-driven prosthetics. The study concluded that there is much-needed research before true artificial limbs can be created and that they must contain subconscious control systems that allow the user to move about their daily lives intuitively.

When determining the physical control of a prosthesis system, many parameters about human motion must be known. From previous studies, maximum thigh angular velocity can be consistently measured during athletic activities [28] [29]. A test was performed with IMUs located on the thigh and shank to measure thigh and knee

angle, power, and velocity (in deg/s) [28]. The data obtained measured thigh angular velocity at a maximum of approximately 260 deg/s (43 RPM) allowing for proper motor sizing of a motor actuated prosthesis. Moreover, this research measured this velocity data at a maximum when the thigh angle is between 25 and 50 degrees flexion. With this velocity known for a single limb loading test, an artificial joint can be confidently created with a motor that should rotate at 43 RPM or faster. This test required athletic movement that produces much higher velocities as compared to sitting and transitioning, thus providing confidence in prosthesis motion control with a motor sized at or slightly below the produced velocities in this study. Another study was performed that measured the instantaneous maximum velocity of thigh motion. The study obtained thigh and shank velocity data about the sagittal, frontal, and transverse planes while subjects performed a drop jump test. This experiment required subjects to jump off of a 36 cm high box and land on both legs, then jump back into the air. IMUs were used to obtain thigh and shank velocity data during the test. Thigh velocity across the sagittal plane is the most important data from this study as it related directly to the movement of the artificial single-axis prosthesis being created. Maximum positive and negative velocities of approximately 400 deg/s and -300 deg/s were measured. Designing a prosthesis with a motor capable of rotating at this speed would provide greater confidence in being able to mirror sagittal-plane thigh-joint movement.

2.4 Gait Pose Analysis

Research has been widely performed on gait pose detection and analysis. Though the prosthesis designed in this thesis does not involve gait motion, it is essential to understand some of these characteristics as they may be necessary for future design if the current prosthesis were implemented in a multi-joint system or used to help amputees walk. Gait pose is especially important when a microprocessor-controlled prosthesis is designed that must know the current location of all limbs and make adjustments to

allow the individual to make ideal future movements. A study by Gerasimos Bastas, et al. from 2018 included using three IMU-based algorithms, zero-crossing, proximal peak, and maximal acceleration to compare gait movements between 14 healthy, 17 transtibial, and 16 transfemoral subjects [30]. The algorithms used signals from a 3-axis IMU mounted in the lumbopelvic region to track acceleration, walking speed, step duration, and body symmetry. Subjects walked 20 meters to create a consistent data log for result analysis. The study concluded that the zero-crossing algorithm produced lower standard deviations in acceleration in all three groups (71.4%, 64.7%, and 81.2%) as well as more consistent results. For this thesis project, these algorithms are not particularly important; however, the location of the IMU used to determine gait poses is. This study chose to locate the IMU in the lumbopelvic region, which displays strong support and provides accurate data during human motion. Similarly, a study performed by Nimsiri Abhayasinghe and Iain Murray utilized peak detection and zero-crossing detection algorithms along with a single thigh-mounted IMU to determine walking, standing, and sitting gait poses of two male and two female human subjects [31]. The study analyzed readings from a 9-axis IMU while the subject performed various indoor gait activities. The time-curves were analyzed, and gait pose was determined based on computed thigh angle, peaks (human subject's steps), and time-delay between changes in the thigh angle curve. The study was able to determine subject standing, walking, and sitting poses with 78%, 92%, and 100% accuracy, respectively.

In 2017, research was performed on the determinability of gait phases and events for healthy and amputee subjects from a shank-mounted IMU [32]. Foot switches were integrated into the shoe of each subject for measuring delay in initial contact (IC) and toe-off (TO). An algorithm written in MATLAB was used to detect gait events. Subjects walked 10 meters at three speeds before then walking up and down a 5.8-meter incline at 5 degrees. Foot switch and IMU data were used to determine the

gait phase (stance and swing) of each subject. The study concluded that amputee subjects had a significant delay between IC and TO compared to that of healthy subjects. Gait phase detection rate was 100% for level and ramp activities proving the effectiveness of an IMU paired with external pressure sensors.

When determining the body position and angle of a human solely with the use of IMUs, it is crucial to provide a means of calibrating the measurement devices. Counter-intuitive to this logic, a study in 2014 by Thomas Seel, Jorg Raisch, and Thomas Schauer was performed that examined joint angle position to include joint axis identification and extension and flexion angle measurements without additional calibration of the IMUs [33]. The study compares IMU data with 3-D motion data to that of a transfemoral amputee. Instead of traditional calibration methods that require the human subject to stand straight and sit at 90 degrees, this study employs estimation equations that create body angle assumptions and do not require calibration. Both IMUs are referenced to the same point, and their values, along with algorithms, are contrasted to generate an accurate joint angle. The study concluded IMU errors to be less than 1 degree on the prosthesis and three degrees on the human. It was also concluded that the methods expressed in this study produced joint location and angular data more consistent with human subject motion than other methods that require human subject movement calibration.

CHAPTER 3: DESIGN OVERVIEW

The goal of this thesis was to improve a transpelvic amputee's ability to sit and recline in a chair. As previously mentioned, transpelvic amputees have difficulty transitioning smoothly into a chair and maintaining an upright sitting position. Naturally, when transitioning to a sitting position, both the left and right lower limbs follow a symmetrical path. When sitting, the body typically sits upright with both thighs parallel to each one another and supporting the lateral motion of the body. The symmetrical position of the thighs prevents damage to the spine and provides a comfortable position. To create an autonomous prosthesis that replicates the symmetrical motion of the right thigh, the right thigh's position must be known or correctly assumed. It is also known that the torso moves when transitioning and reclining into a sitting position. The hip area, where a prosthesis would be mounted to, rotates across the sagittal plane via thigh movement and lumbar torso movement. For this project, the torso and right thigh positions are accurately measured in real-time to identify necessary output data for hip-joint reflection. To simulate the sagittal plane motion of the hip-joint, a motor is used and set along the transverse plane with the shaft turning around the transverse axis, producing rotation across the sagittal plane. The motion produced replicates flexion and extension of the hip joint and is restricted in both directions, similar to the necessary range of motion for a human's hip-joint. Knee and ankle joints are not designed nor analyzed in this project as the goal aims at producing a hip joint that would allow motion of a prosthetic thigh; however, the design and findings could, with some uncertainty, be of use in a multi-joint prosthesis.

To create a proof of concept, single-axis, motor-controlled hip joint, two significant aspects were designed, prototyped, and tested: human joint angle determination and simulated joint actuation. Human joint angle determination involves obtaining real-time body positional data and computing a relative joint angle. Joint actuation is to be performed by a stepper motor and controlled via an embedded microcon-

troller with a feedback control system. The entire system includes many components which can be separated into 3 major subsystems: Human Motion Collection, Control, and Joint Actuation. The Human Motion Collection subsystem includes two body-mounted Inertial Measurement Units (IMU). The Control subsystem includes a National Instruments (NI) myRIO embedded microcontroller, and code written in LabVIEW workbench software. The Joint Actuation subsystem includes a stepper motor, motor controller, power converter, and feedback motor encoder. Finally, to ensure the data obtained from the IMUs and the encoder are accurate, a 3-Dimensional (3-D), six degrees of freedom (DoF), motion capture camera system is used to validate the data.

A high-level approach of the system is modeled in Figure 3.1. Body-mounted IMUs offer input data to the system while an embedded microcontroller executes math, produces logic, outputs data to control the joint motor, and makes adjustments via feedback input data from an encoder mounted to the joint motor.

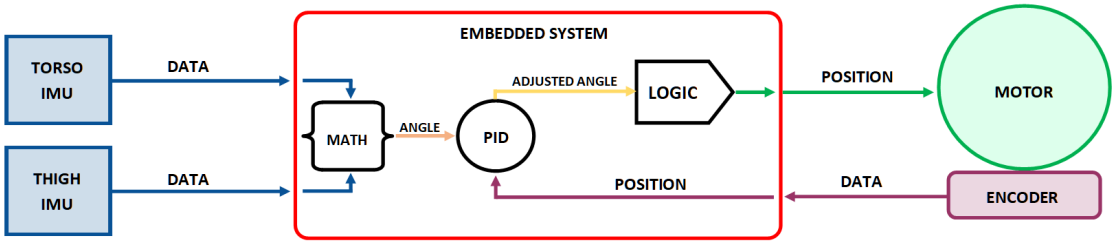


Figure 3.1: System-level system data flow.

3.1 Human Motion Collection

When determining the angle of a physical object, its pose compared to the gravity acceleration vector can be utilized. For this thesis, two GY-521 modules are used as the means to measure thigh and torso orientation. One of these modules can be viewed in Figure 3.2. Each module is a breakout board containing an MPU-6050 six-axis microelectromechanical system (MEMS) device that tracks acceleration and angular velocity in each of three axes, creating 6 degrees of freedom. Each GY-521

module is powered using +3.3 V from the embedded controller. The MPU-6050 is programmable for sensing acceleration in the full-scale range from ±2 g to ±16 g and gyroscope angular velocity in the full-scale range from ±250 degrees/sec to ±2000 degrees/sec.

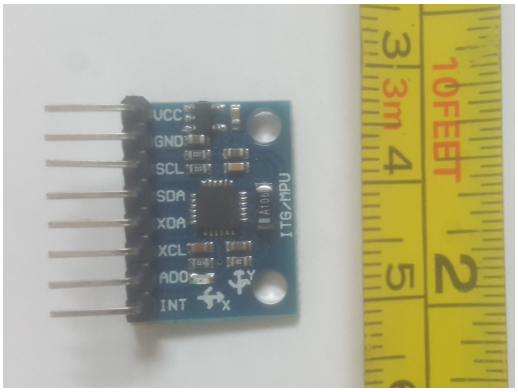


Figure 3.2: GY-521 breakout board containing MPU-6050 IMU.

The MPU-6050 communicates over an Inter-Integrated Circuit (I^2C) data bus with a default slave address of 0x68. Two MPU-6050 devices are used; each one communicates on its myRIO I^2C channel to allow for simultaneous data capture. The IMU designated as the "thigh IMU" communicates over Channel A, whereas the "torso IMU" uses Channel B. Specific information on the MPU-6050 and its communication over the I^2C data bus with the myRIO is explained in further detail in the Methodology chapter of this document. The GY521 breakout board is a compact device, measuring 2 cm x 1.5 cm, which provides uninterrupted data without hindering the full range of motion of the body during testing.

When transitioning from standing to sitting and vise-versa, the thigh and pelvis both move about the transverse axis. A hip prosthesis mounted to the torso of a transpelvic amputee would need to move with the lower torso nearest the ilium, as well as beyond the hip joint at the artificial thigh. To assist in producing symmetrical tracked movement, the non-amputated thigh and torso angles must be measured in real-time. For measuring thigh angle, an IMU is mounted on the ventral side of the

thigh, midway up. Refer to Figure 3.3.

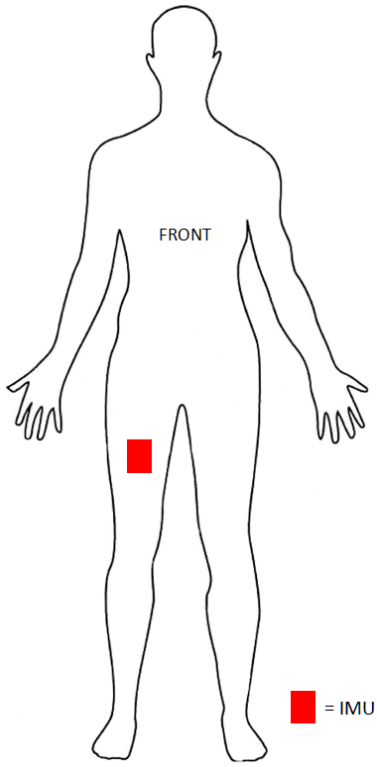


Figure 3.3: Location of thigh mounted IMU.

The lateral and medial parts of the thigh were not chosen due to distortion of the body mounting surface when sitting and transitioning. The dorsal side of the thigh is also not used due to substantial changes in surface angle when the hamstring muscles are exercised. The ventral side of the thigh offers the most consistent results and provides a relatively flat angle when sitting in a chair perpendicular to the ground. The IMU is attached to a 3-D printed platform that is then attached to the right thigh via a flexible hook-and-latch band such that the y-axis travels in-line with the thigh, the x-axis travels laterally across the thigh, and the z-axis is mounted along the anteroposterior axis. The platform also contains 3-D motion markers that are used for data validation and motion tracking, which are explained in more detail later in this document.

The IMU mounted for torso measurements was placed just above the iliac crest

on the right section of the lower back, generating torso measurements as close to the transverse plane as possible. The IMU is attached to a 3-D printed platform that keeps the IMU straight against the body. This platform is then attached to a 2-inch wide hook-and-latch waist-band that is fit snug to the body. The location of the IMU is relatively flat when standing and sitting and is not distorted by abdominal muscle movement. Body shape, weight, and especially breathing would create inaccurate torso angle measurements if the IMU were to be placed on the dorsal side of the torso. A robust and rigid location near the lower spine is ideal for torso angle measurement. It is essential to locate the IMU just above the iliac crest to ensure upper body movements beyond the lumbar region aren't affecting hip joint measurements. The actual angles being measured are the thigh and the lumbar section of the torso about the transverse axis. Mounting the IMU far above the lumbar region would capture torso angle measurements above where a prosthesis would be mounted, thus generating a net calculated angle that would be overcompensated during the transitional phase of sitting and standing and cause the artificial limb to rotate much closer to the body than necessary. Figure 3.4 shows the relative location of the torso IMU.

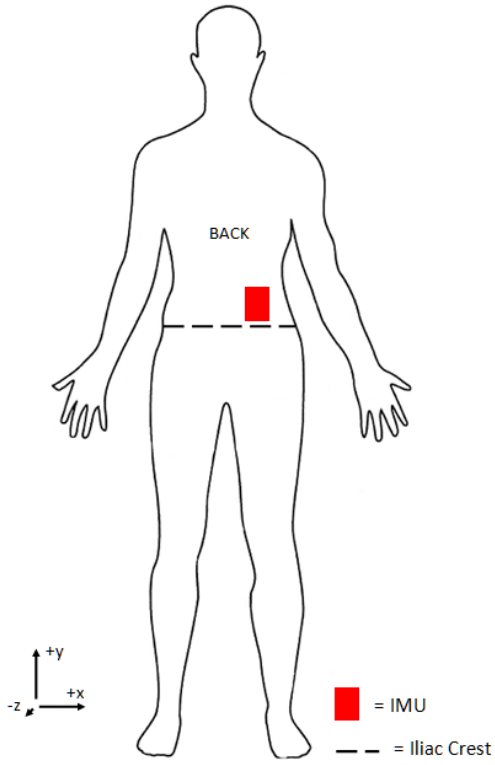


Figure 3.4: Location of torso mounted IMU.

Figure 3.5, for explanatory purposes only, displays possible thigh and torso angles during body movements in a relative angular position to help understand the relationship between the two. The measured angle is relative across the sagittal plane with the zero-degree position measuring a line vertical towards the head and positive degrees rotating down the dorsal side of the body.

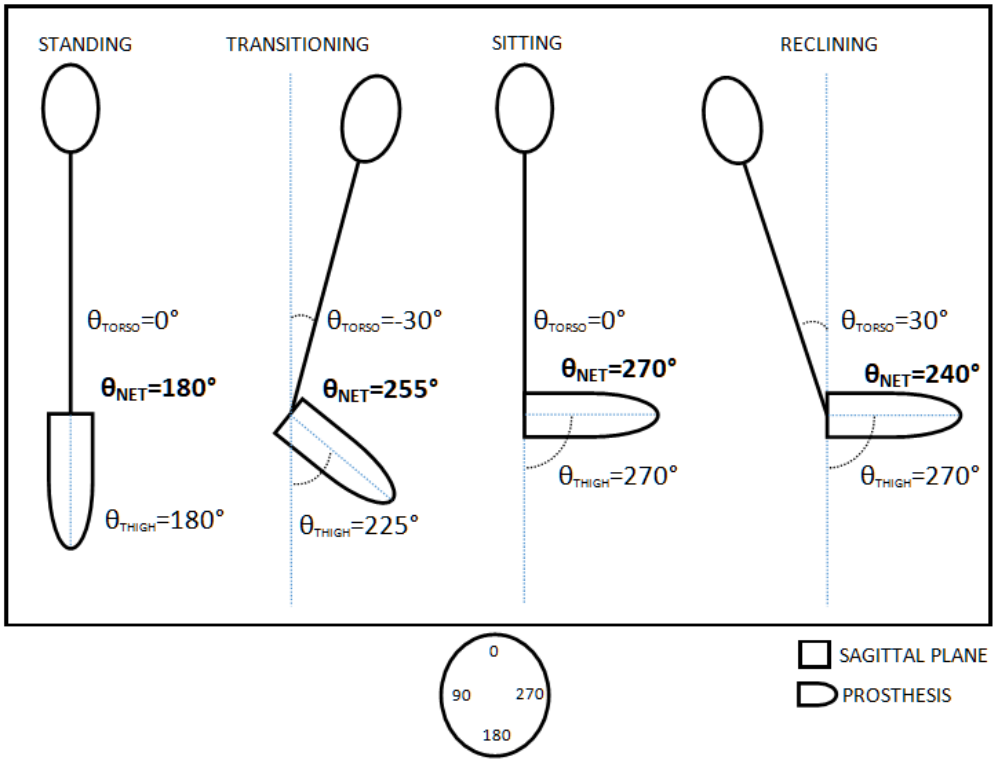


Figure 3.5: Body positions and their possible measured angles.

To calculate the net angle between the torso and thigh, Equation 3.1 is used.

$$\theta_{NET} = \theta_{THIGH} - \theta_{TORSO} \tag{3.1}$$

As an example, when standing perfectly straight, θ_{NET} would theoretically equal 180° while sitting straight could theoretically yield θ_{NET} equal to 270°. These angles are relative to a counterclockwise rotation with the zero-degree mark positioned at twelve o'clock.

3.2 Joint Actuation

A dual shaft stepper motor was selected for simulated single-axis hip-joint operation. A stepper motor allows for precise bidirectional control of the artificial limb as well as persistent torque. A high driving voltage of 36 V is used to provide fast ramp-up between steps, creating a smoother overall step. This motor maintains a

standard NEMA 24 frame size (60 mm x 60 mm) with 8 mm diameter front and rear shafts. The rear shaft of the motor allows for an external motor encoder. This two-phase motor is wired with each phase in parallel, consuming 4.24 A_{RMS} per phase. It has been chosen to control the motor at 1,000 steps per revolution, with a 1 kHz pulse frequency and an operating voltage of 36 V. This produces smooth, continuous motion at 60 RPM. Equation 3.2 was used to determine the appropriate operating frequency to produce a motor rotation speed of 60 RPM.

$$f = \omega/(\gamma * 360 * 60) \tag{3.2}$$

Where

f = frequency (Hz)

ω = angular velocity (revs/min)

γ = step angle (deg/step)

According to the manufacture's load chart, seen in Figure 3.6, while operating at 60 RPMs and microstepped at 2,000 steps/rev, this stepper motor produces approximately 2.75 Nm of pull-out torque [34]. Operating at smaller microstepping produces high torque but less smooth step transitions. The blue line in Figure 3.6 denotes operation at 36 V and 4 A_{RMS}

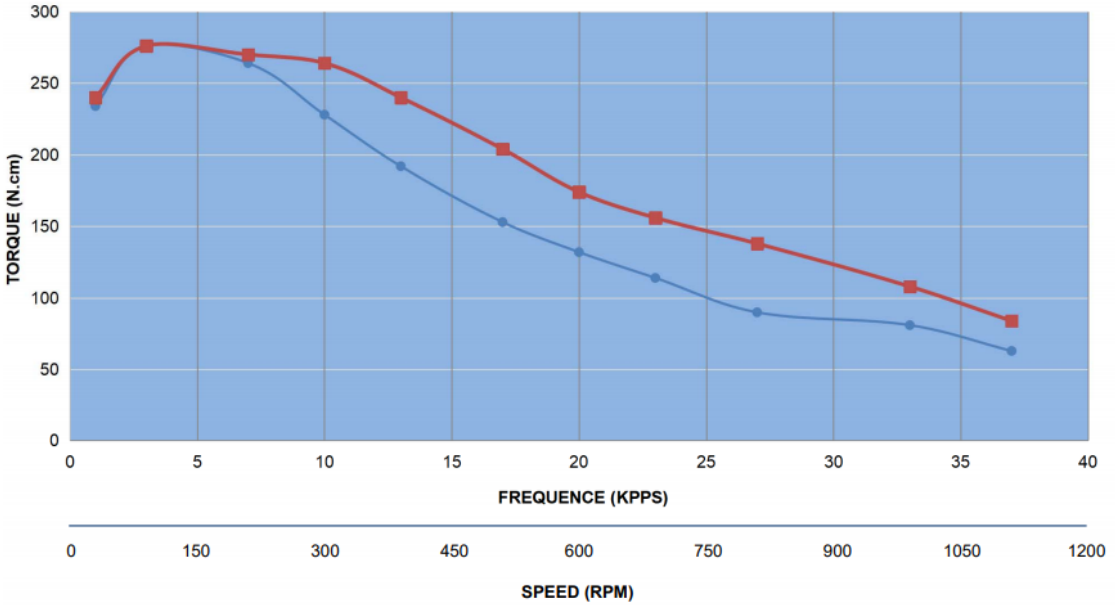


Figure 3.6: 24HS39-4204D pull out torque specification chart.

The stepper motor's rated holding torque of 4 Nm can be achieved with both phases energized and a supply current of 8.48 A; however, this is not safely achievable with the RTELLIGENT R60 stepper motor driver used for this design. The holding torque of the motor is not tested in this project as the project's primary focus is on system control and position data analysis. Motor torque and current characteristics should be more closely examined when the system is scaled to a larger prosthesis prototype.

For driving the stepper motor, an RTELLIGENT R60 stepper motor driver is used. This device allows 2-phase driving with an input voltage range of 18-50 VDC. Its maximum rated drive current is 5.6 A peak at 50% duty cycle. For experimentation and data collection, the current has been limited to 4.3 A peak. The driver is supplied with 36 VDC from the NOYITO AC-DC power module. Microstepping is supported from 200 steps/rev to 25,000 steps/rev. As mentioned previously, 1,000 steps/rev was selected to ensure smooth and precise motor control. This value was chosen to ensure motor precision is more coarse than encoder precision to ensure the motor does not move in such small increments that the encoder is unable to measure a change in motion. Motor enable (ENA), pulse (PUL), and direction (DIR) are all

sinking outputs, requiring a ground from the microcontroller to operate. This device measures 112 mm long, 75.5 mm wide, and 33 mm tall.

The NOYITO AC-DC switching power module was chosen to supply power to the stepper motor. This power module receives 120 VAC 60 Hz and outputs 36 VDC with a max continuous current of 5 A. This device provides maximum rated current (4.24 A peak) to a single phase of the stepper motor in a relatively compact package.

An AMT203-V 12-bit absolute encoder was selected as an affordable yet precise means of measuring artificial limb rotational position. This encoder utilizes the Serial Peripheral Interface (SPI) communication protocol and provides system feedback at a maximum update rate of 48 μs. The encoder provided 12 bits of resolution at 4096 positions per revolution and a frame length of 8 bits, which requires two frame concatenation for full precision. In cases of artificial limb movement while the controller is not powered, this encoder provides immediate position upon system power-up. The device also provides developer position zeroing, discussed later in this thesis, which was leveraged for calibrating a null motor position for data logging. A detailed datasheet of this encoder can be accessed from CUI Devices [35].

3.3 System Control

All system inputs, control, computations, and communications are fed through a National Instruments (NI) myRio-1900 embedded device. The myRIO contains 40 digital input/output (DIO) connections, 10 analog inputs, and 6 analog outputs, an onboard accelerometer, a Xilinx FPGA, and Cortex-A9 ARM processor. A 6-16 VDC power supply is required to power the myRIO.

The myRIO contains many communication protocols distributed across its FPGA while processing and computations take place through its Real-Time processor, both of which interface with NI's LabVIEW development software either via USB or WiFi. A visual of this can be seen in Figure 3.7 from NI's myRIO-1900 User Guide and Specifications document [36].

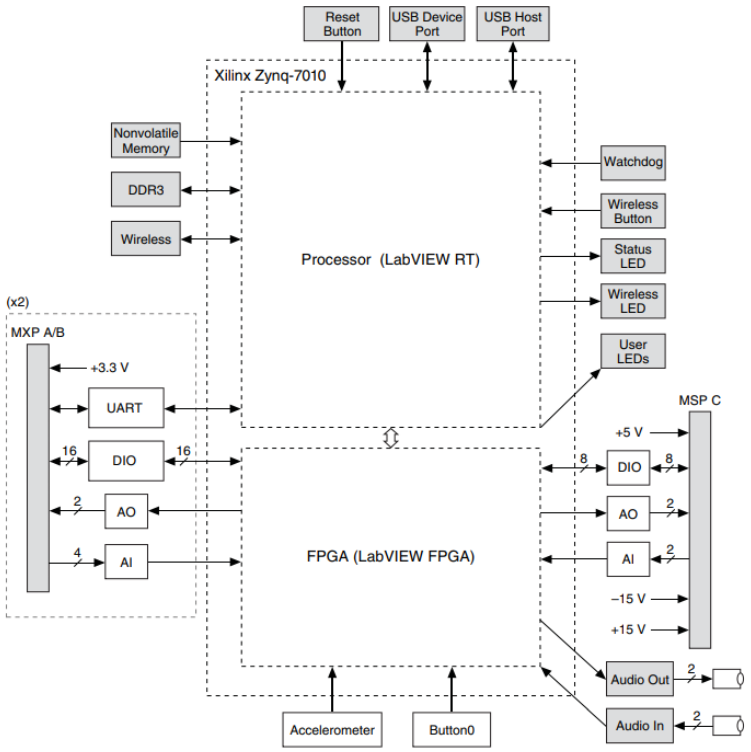


Figure 3.7: Hardware Overview of myRIO-1900.

The Xilinx Z-7010 processor contains two cores and operates at 667 MHz. It contains 512 MB of memory and 256 MB of DDR3 RAM operating at a clock frequency of 533 MHz. Digital outputs, including pulse-width modulation (PWM) outputs, operate high between 2.4 V and 3.465 V and low between 0 V and 0.4 V. PWM is restricted to 100 kHz maximum operating frequency, SPI is restricted to 4 MHz, and I^2C can operate in both Standard (100 kHz) and Fast (400 kHz) modes. For this project, I^2C is operated in Standard mode.

There are two (Channel A and B) myRIO Expansion Ports (MXP); each provides access to 16 DIOs, 3 PWM outputs, 4 analog inputs, 2 analog outputs, I^2C, UART, and SPI communications pins. +5 V and +3.3 V can also be accessed through this connector. These connectors are limited to a maximum current of 100 mA at +5 V and 150 mA at +3.3 V. For this reason, the stepper motor is powered through a separate driver and power supply. Digital and analog grounds are present on all three

connectors. The myRIO physical dimensions are 136.6 mm long, 86 mm wide, and 24.7 mm tall.

CHAPTER 4: METHODOLOGY

4.1 IMU Configuration

To communicate with the myRIO, the MPU-6050 communicates over the I^2C data communication bus. The thigh IMU is connected to the myRIO via Channel A MXP connector I^2C pins. The torso IMU is connected via the Channel B MXP connector. Because two channels are used, both IMUs can share the same slave address. The address selector pin on the GY-521 breakout board is tied to ground to ensure the default I^2C slave address is of 0x68 is used. Tying this pin high would change the slave address to 0x69, which is useful when communicating on a single channel I^2C data bus. The GY-521 contains eight pins for configuration and control of the MPU-6050. VCC is tied to +3.3 V, GND, and AD0 to myRIO MXP connector digital ground, SCL to I2C.SCL, and SDA to I2C.SDA. The remaining three pins are not used. A wiring diagram of this can be viewed in Figure 4.1.

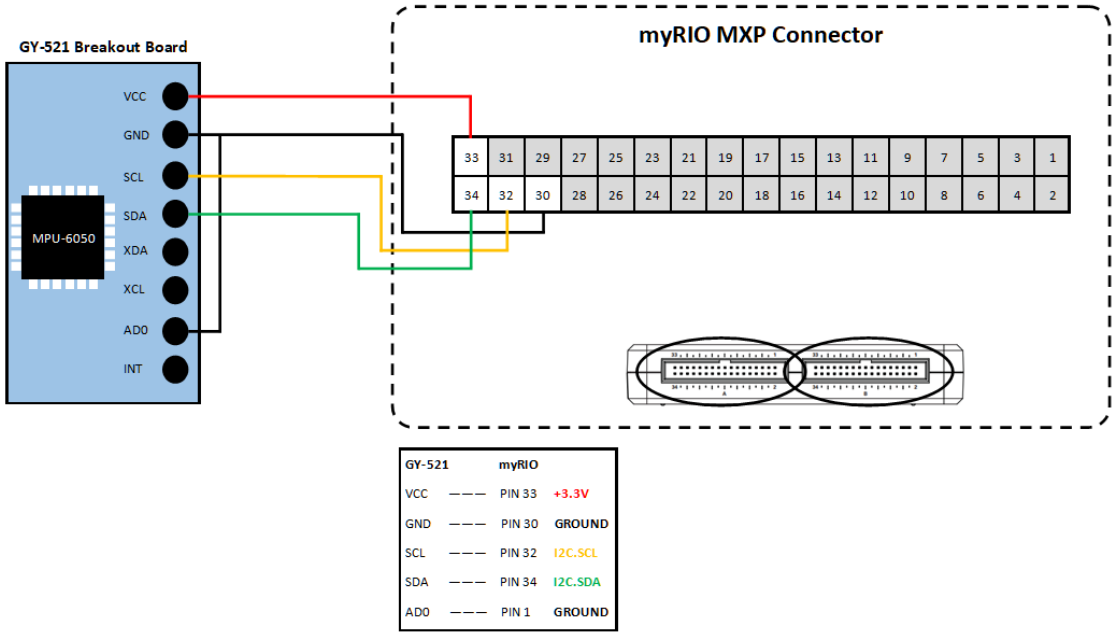


Figure 4.1: IMU to myRIO MXP wiring diagram.

The myRIO I^2C Channels (A and B) are configured in Standard Mode (100kbps).

Specific configuration settings such as the Digital Lowpass Filter (DLPF) and Digital Highpass Filter (DHPF) viewed in Figure 4.1, are written to the MPU-6050 over I^2C to provide ideal operating parameters for this project's application.

Table 4.1: MPU-6050 startup register configuration.

Register #	Name	Value	Description
0x19	$SMPLRT\_DIV$	0x00	8 kHz Sample Rate
0x1A	CONFIG	0x00	DLPF Setting
0x1C	$ACCEL\_CONFIG$	0x00	DHPF Setting
0x38	INT_ENABLE	0x01	Enable data ready
0x6B	PWR_MGMT_1	0x00	Use 8MHz oscillator

The three Least Significant Bits (LSB) of register 0x1A are used to configure the DLPF, which controls the bandwidth and consequently delay for accelerometer and gyroscope data. The configuration setting of 0x00 disables this filter and creates a 260 Hz passband with an added output delay of 0ms. It has been concluded through extensive testing that producing output data with no delay far outweighs the advantage of a tight low-pass filter. Further configuration settings for this register can be viewed in Figure 4.2 from the MPU-6000 Register Map [37].

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Figure 4.2: MPU-6050 register 0x1A DLPF configuration.

Register 0x19 is configured with the value 0x00, which ensures the Digital Motion Processor sampling rate is 8 kHz. This step is not entirely necessary, but added as redundancy, as the DLPF filter has been configured with 0x00, which disables the filter. When the DLPF filter has been disabled, the sampling rate defaults to 8 kHz. Maintaining this 8kHz sampling rate by configuring register 0x19 as such, allows for fast adjustment of the DLPF during testing without having to reconfigure the sampling rate.

Register 0x1C contains four settings for the MPU-6050's DHPF, which ranges from ± 2 g to ± 16 g. Human motions performed in this project are not capable of producing data at high frequencies, thus a tight range of data is selected at ± 2 g by configuring the register as 0x00. Further, DHPF configuration settings can be seen in Figure 4.3 from the MPU-6000 Register Map [37].

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

Figure 4.3: MPU-6050 register 0x1C DLHF configuration.

Register 0x38 must be set to 0x01 for the MPU-6050 to know configuration has been set, and it should prepare to send data. Register 0x6B contains multiple settings for the internal clock and power mode of the MPU-6050. A setting of 0x00 selects the 8 MHz oscillator and ensures the device is not in low power sleep mode. Both MPU-6050s are programmed identically. LabVIEW code for setting the configuration data of the thigh MPU-6050 can be viewed in Appendix A.1.

For data processing, the myRIO is configured to read 14 bytes of information every 100 μs that contains the accelerometer, gyroscope, and thermometer information. For this project, the first 6 bytes of information are used to determine the IMU angle.

In total, accelerometer data is read as 16 bits across two registers for each axis (x, y, and z) according to Figure 4.4 from the MPU-6000 Register Map [37].

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

Figure 4.4: MPU-6050 accelerometer registers for x, y, and z axes.

The bits of each axis are then digitally joined and converted into 16-bit integers via LabVIEW. A moving average of the previous 20 values is generated, and a 64-bit floating-point number is produced that is used for angle computation. This process occurs for each axis on both IMUs. A visual of the code written in LabVIEW for this process can be viewed in Appendix A.2. To calculate the pitch angle of each IMU in degrees, that is, the angle created across the sagittal plane when standing and sitting by the thigh, Equations 4.1 and 4.2 were used to determine thigh pitch angle (θ_{THIGH}) and Equations 4.3 and 4.4 were used to determine torso pitch angle (θ_{TORSO}) using raw accelerometer values (X_{acc}, Y_{acc}, and Z_{acc}). When the thigh is in extension (motion in front of the body), Equation 4.1 was used as the raw Z_{acc} value is positive. When the thigh is in extension, Equation 4.2 was used. When the torso tilted forward, Equation 4.3 is used as the orientation of the IMU in the torso IMU holder generates a negative raw Z_{acc} value. Finally, Equation 4.4 is used when the torso is reclined.

$$\theta_{THIGH} = \arccos\left(\frac{Y_{acc}}{\sqrt{X_{acc}^2 + Y_{acc}^2 + Z_{acc}^2}}\right) * \frac{180}{\pi} \tag{4.1}$$

$$\theta_{THIGH} = -\arccos\left(\frac{Y_{acc}}{\sqrt{X_{acc}^2 + Y_{acc}^2 + Z_{acc}^2}}\right) * \frac{180}{\pi} \tag{4.2}$$

$$\theta_{TORSO} = -[180 - \arccos\left(\frac{Y_{acc}}{\sqrt{X_{acc}^2 + Y_{acc}^2 + Z_{acc}^2}}\right)] * \frac{180}{\pi} \tag{4.3}$$

$$\theta_{TORSO} = 180 - \arccos\left(\frac{Y_{acc}}{\sqrt{X_{acc}^2 + Y_{acc}^2 + Z_{acc}^2}}\right) * \frac{180}{\pi} \tag{4.4}$$

The 180-degree translation for the torso pitch calculations is necessary as the IMU holder places the torso IMU in-line with the thigh IMU; however, when standing upright, the thigh is oriented 180 degrees away from the torso. This condition is illustrated in the "Standing" section of Figure 3.5.

4.2 Encoder Feedback Configuration

The AMT203-V absolute encoder by CUI Devices communicates via SPI with the myRIO. Six pins on the encoder ISDF-07-D connector are used and wired to the myRIO MXP Connector B, as viewed in Figure 4.5.

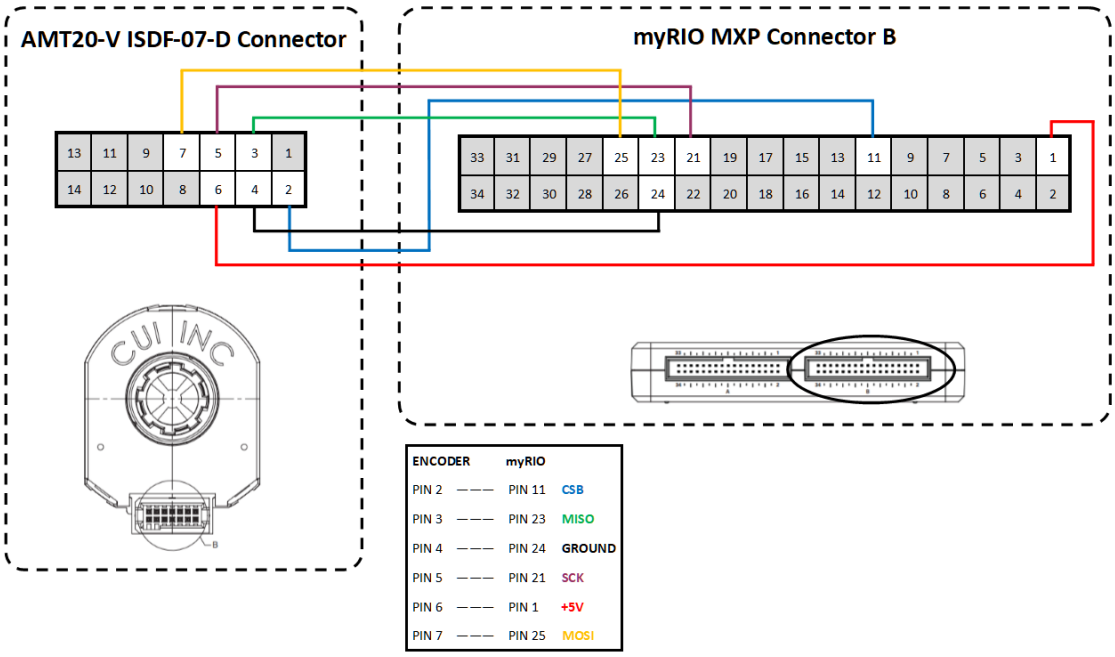


Figure 4.5: AMT20-V encoder to myRIO wiring diagram.

The CSB pin is tied to the myRIO's Channel B Digital Input/Output "0" (DIO0) pin and operated in an enable/disable sequence each cycle to allow the encoder to

receive data and then transmit response data. CSB is set low and allows data transmission from the slave (myRIO). On the falling edge of this clock, the myRIO sends the read position command as decimal "16" to the encoder via the MOSI line. The encoder receives this signal and replies via the MISO line with an acknowledgment command of "16". The encoder then sends its current position to the myRIO in two separate cycles; 4 MSB followed by the 8 LSB bits of its position. This two-cycle data transfer occurs because the encoder communicates via SPI in 8-bit frame lengths but contains 12 bits of position data. CSB is then set high to end transmission of the encoder position. This 12-bit position must then be converted into an angle from 0 to 360 degrees to be used for accurate artificial limb positional feedback. This value is updated every 50 μs, as it is recommended to request the encoder's position no faster than 47 μs at a time [35].

To allow interpretation of this data in LabVIEW, a shift register and case structure were created. SPI VI settings should be set as follows to communicate with the encoder properly:

Clock Frequency = "1 MHz"
Frame Length = "8 bits"
Clock Phase = "Trailing"
Clock Polarity = "Low"
Data Direction = "Most Significant bit first"

The read position command (16) is written over SPI, and three frames of information are received. "16" is received first, followed by 4 bits, and then 8 bits. As these frames are read, they are moved into a three-position shift register. When the 4 bits are received, they are placed in the first shift register position, and the "16" response bits are moved into the second shift register position. When the last 8 bits are received, they are put into the first shift register position, the 4 bits are moved into the second position, and the "16" response bits are moved into the third position. The

case structure is designed to read the frame of bits in the third shift register position. When this position contains the decimal value "16", the statement is true, and the first two shift register positions are read and converted into two 2-bit hexadecimal numbers. These two numbers are then concatenated into a single 4-bit hexadecimal number and then converted into an integer that ranges from 0 to 4095. This positional value is then divided by 11.378 (4096/360) to produce an encoder angle value between 0 and 359.912 degrees with a resolution of 0.088 degrees. This process can be visualized through the snippet of LabVIEW code in Appendix A.3.

To zero the encoder, that is to generate a new "0" degrees position, the following steps were performed. The artificial limb was positioned vertically straight about the positive y-axis as possible. This position represents the 0 degrees position for net artificial limb motion. 180 degrees represents the limb pointing perpendicular towards the ground. The "zero set" command (decimal 112) was sent to the encoder from the myRIO via the SPI MOSI line. The encoder sends back to the myRIO, via MISO, the zero set successful response (decimal 128). The encoder is then powered off and back on to save the zero position. Appendix A.4 shows the LabVIEW code for this operation.

4.3 Motor Control and Feedback

Three parameters determine control of the motor: the commanded angle determined from IMU calculations, the current motor position determined by the encoder, and how quickly the motor is approaching the commanded angle. When the myRIO calculates the required limb angle, the current angle of the motor is then compared to the required limb angle. This first determines if the motor should be commanded to move, and then determines the direction of travel of the motor. The control system has a 0.5-degree maximum error programmed such that if the current position of the motor is within 0.5 degrees of the commanded position, the motor does not move. This deadband helps stabilize the system since the encoder has a rated accuracy of

0.2 degrees and produces a range of uncertainty 0.4 degrees. It should be noted that without additional variables which would be obtained from a physical prosthesis, this 0.5 degrees of error may be far too small to produce ideal control of the system. The need for an investigation into this matter is discussed in Future Work. For this control system, if the difference in angle is greater than 0.5 degrees, then the motor is commanded to move at 60 RPM to its new position. A proportional-derivative (PD) controller has been included in the control of the motor in LabVIEW. This PD controller uses the calculated commanded limb angle as the set-point, the encoder position as the input, and produces an output that controls the speed of the motor.

Experimental testing has produced the best results when the proportional constant (k_p) is set to 1.5, and the derivative constant (k_d) is set to 0.75. To obtain these values, a standing-to-sitting motion was performed multiple times with just proportional gain. After locating a proportional constant that created excellent rise time, the derivative gain was adjusted to reduce system overshoot and oscillation. Increasing derivative gain too much significantly slowed the rise time of the system. A combination of PD that produced fast rise time and low overshoot was selected. An integral gain was not needed as the system corrects for 0.5-degree steady-state error by commanding the motor to move when the measured error in angle is greater than 0.5 degrees. The PD controller is called in every loop the system is run, obtaining a new encoder value as a process variable (input) and using it as feedback to the system for variable control of the motor speed. A diagram of this process can be viewed in 4.6.

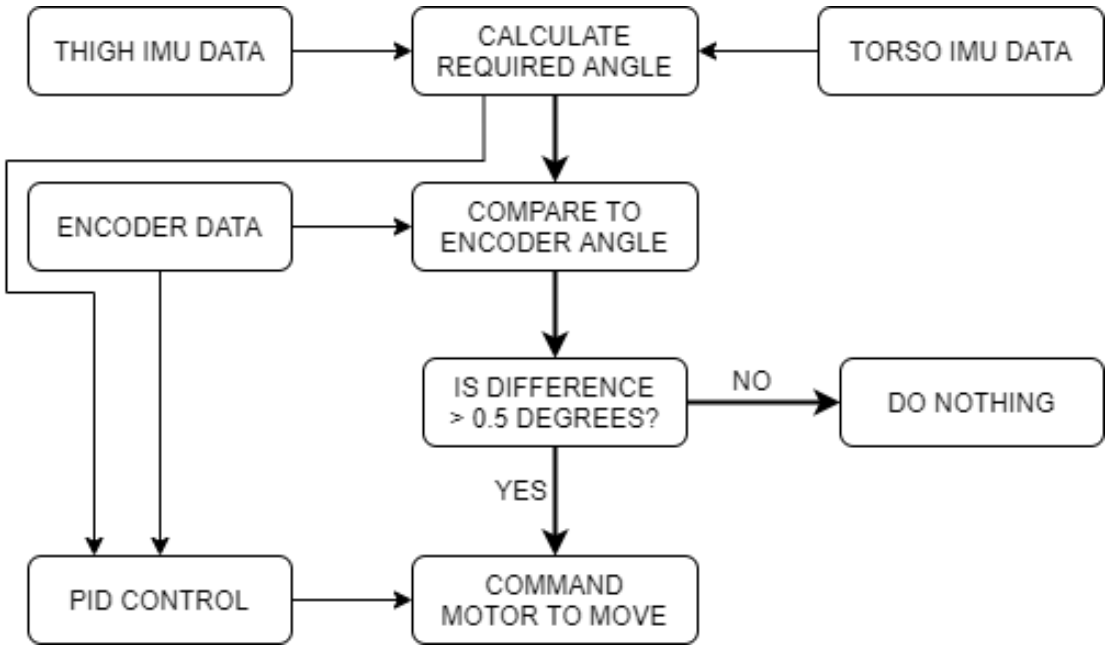


Figure 4.6: Flow of data for motor control and decision making.

To control the speed of the motor, a 1 kHz pulse is sent to the "pulse" input of the stepper motor controller when the motor is commanded to move. The PD controller makes real-time adjustments to this control frequency, reducing it as the motor nears its commanded position (set-point). To control the direction of the motor, the commanded position is compared to the motor's current position. If the motor's current position is less than the commanded position, the motor controller's "direction" input is sent a digital "1" from the myRIO to change the direction of movement counterclockwise. The opposite is done when the motor position is greater than the commanded position. The LabVIEW code created to control the motor can be viewed in Appendix A.6.

4.4 Data Validation

To ensure the digital data obtained from both IMUs and the motor encoder is accurate and representative of the physical motion of the system, a high-speed 3-D camera system from Optitrack is used to validate both stationary and motion data. For this project, the camera system operated using 8 cameras strategically placed

around the test area to obtain real-time motion data of a flange connected to the shaft of the motor, and two IMU holders attached to the human body. Each camera has a resolution of 1.3 megapixels (1280 x 1024), streams data at 240 FPS (frames per second), and maintains horizontal and vertical FOVs (field of view) of 56 and 46 degrees respectively. It is known that each camera runs with a latency of 4.2 ms.

A flange, seen in Figure 4.7, was 3-D printed and attached to the shaft of the motor. The flange contains three markers positioned in such a way to produce an asymmetrical rigid body allowing reliable motion tracking of the physical angular position of the motor. CAD drawings of this flange can be Appendix B.1.

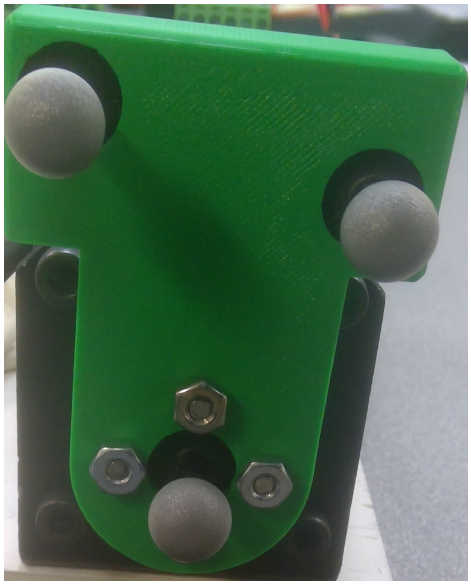
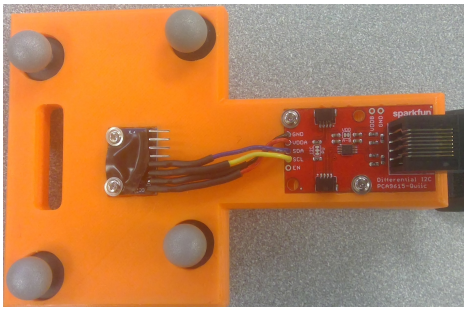


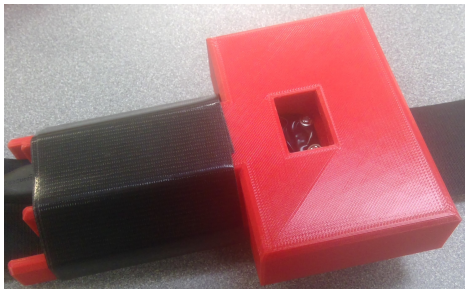
Figure 4.7: Data validation motor flange.

The thigh and torso IMUs, seen in Figure 4.8a and Figure 4.8b respectively, are attached to custom holders that are affixed to the body and also contains four markers positioned in unique orientations. The orientation of the IMU holders on the body can be seen in Figures 4.9a and 4.9b. Both IMU holders produce a rigid body with a plane approximately parallel to the IMUs such that the 3-D motion system can identify and validate the 3-dimensional position of the IMUs with minimal angle offsets. Computer Aided Design (CAD) drawings of these IMU holders can be viewed

in Appendix B.2 and Appendix B.3.



(a) Thigh IMU holder.



(b) Torso IMU holder.

Figure 4.8: IMU and PCA9615 holders.

(a) Thigh IMU holder.

(b) Torso IMU holder.

Figure 4.9: Location of IMU holders on the body during data collection.

All motion data is streamed to a central computer running Optitrack's Motive software version 1.10.3. This software provides a real-time visualization of the markers and bodies being tracked. Motive streams raw x, y, and z data at 240 FPS to its local application where. A separate computer that is connected to the myRIO and running LabVIEW can fetch the camera data while also logging LabVIEW data. Optitrack's NatNet Software Development Kit (SDK) version 2.10 is utilized inside of MATLAB

such that a direct connection is made between Motive and MATLAB. All streamed data is logged in real-time from the camera system and made available by the local computer via multicast. Using LabVIEW's built-in Open Platform Communication (OPC) server, all IMU and encoder data is streamed directly to MATLAB, where it is analyzed alongside data retrieved from the camera system. The MATLAB scripts created to log system validation data can be Appendices C.3, C.4, C.5, C.6. Figure 4.10 displays a visual of the communication backend, allowing seamless data logging into MATLAB.

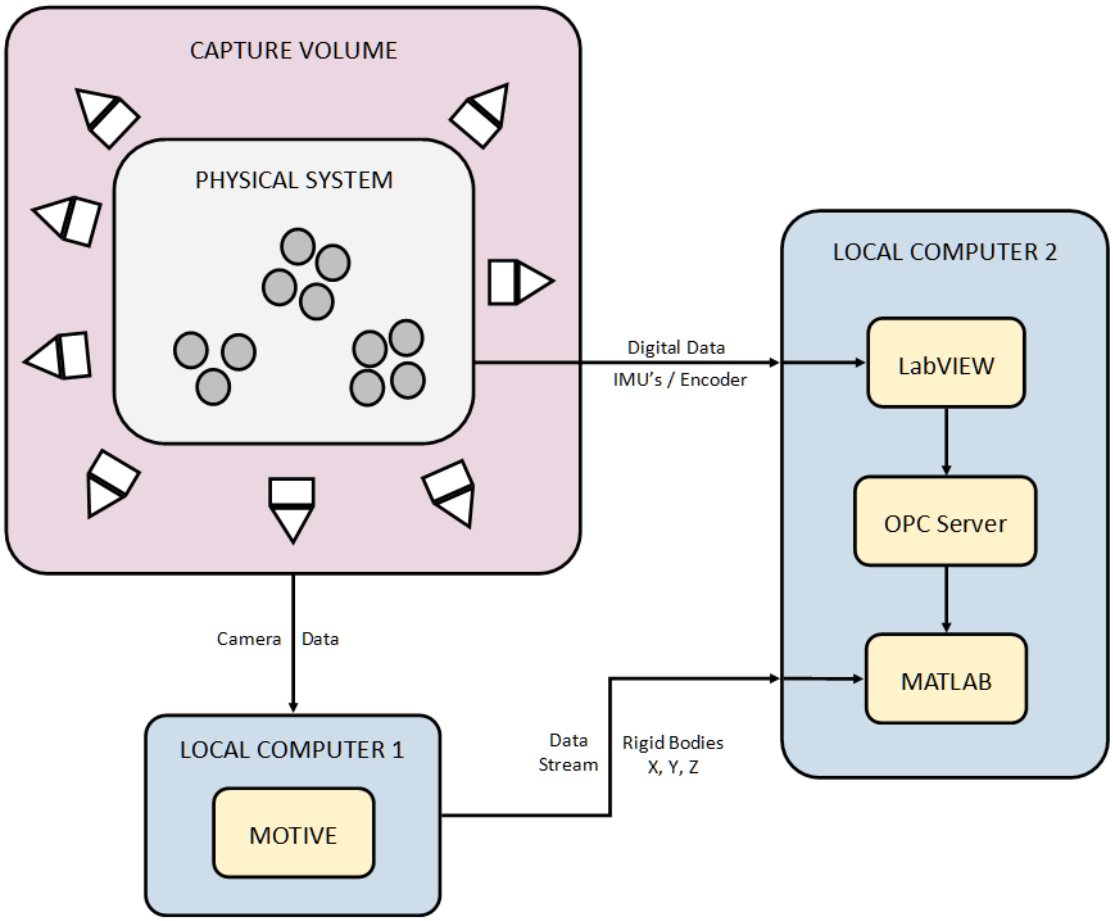


Figure 4.10: Data validation flow into MATLAB.

For data validation, the system and its data are logged and analyzed in stages to ensure all components are producing reliable and accurate data. All system components that were not attached to the human body were mounted to a board, as

seen in Figure 4.11, that allowed the motor and attached flange to rotate freely while maintaining a rigid platform for data capture.

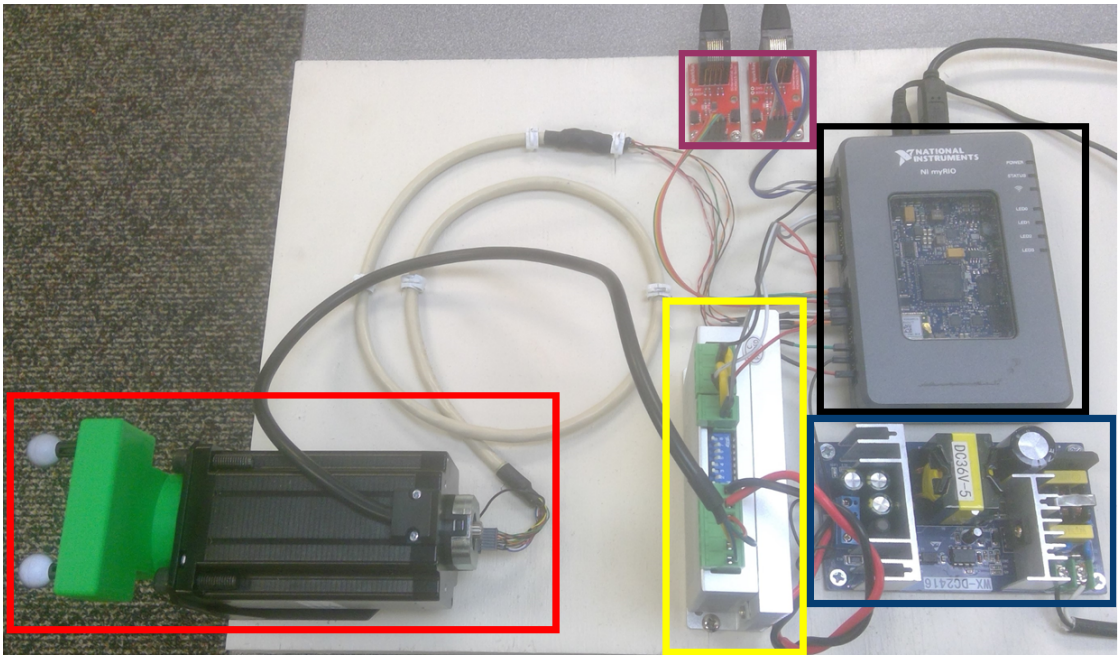


Figure 4.11: System components test board.

The red box contains the stepper motor, motor encoder, and motor flange while the yellow box contains the stepper motor driver, and the blue box contains the stepper motor power supply. The black box contains the myRIO embedded system. The purple box contains two PCA9615 differential I2C breakout boards which are discussed in more detail in "Section 5.2 Data Logging". The placement of the test board was as central to the camera system as possible to ensure all motion was contained to the capture volume. Two cameras were placed at lower verticals to produce diversity in capture angle and to ensure all markers are visible by a maximum number of cameras.

To validate the motor, a set of 10 in-motion datasets were observed and analyzed as well as 28 stationary datasets. For the in-motion datasets, the motor was initially positioned at approximately 45 degrees and rotated to 315 degrees via a LabVIEW VI. The motor was then rotated back to approximately 45 degrees. This motion was

repeated 5 times for each rotational direction producing a total of 10 datasets. The rotational position of the encoder was logged alongside the 3-D physical shaft position from the camera system.

To validate the thigh and torso IMUs, one stationary and one in-motion test were performed. To validate in-motion data, the IMUs were attached to the appropriate locations on the body, and a motion was performed 10 times that involved starting with the right leg straight towards the floor, moving the leg from that position up towards the abdomen, and then fluidly motioning the leg back to the starting position. Calculated pitch angles from both IMUs were logged as well as the calculated pitch angle of the IMU holders from the camera system. This test was performed 10 times. The stationary test involved laying the IMU holders on a table at specific orientations and logging their pitch angles for 5 consecutive seconds. The Optitrack cameras were also capturing the physical position of the IMU holders during the same period. For the thigh IMU, orientation was tested with the IMU placed flat on the table, producing an angle of approximately 90 degrees and then tilted slightly on end to produce an angle of approximately 5 degrees. For the torso IMU, orientation was tested resting flat on the table, producing an angle of approximately -90 degrees and then upright producing an angle of approximately 0 degrees.

The MATLAB script used to obtain all necessary data in real-time can be viewed in Appendix C.7; the IP address have been removed for security purposes. The script first sets up file constraints and file paths and connects to Motive's multicast data stream. The script then connects to the local LabVIEW OPC server; next, data log constraints are created and data logging commences. Two scripts, "motiveData" and "labviewData", are called every 25 ms to obtain 3-D camera data and LabVIEW digital data respectively. The "motiveData" script, presented in Appendix C.9, reads the most recent frame of data for the three rigid bodies (motor flange, thigh IMU holder, and torso IMU holder), and performs math from the quaternion numbers

obtained to generate a pitch, roll, and yaw angle of the rigid bodies. Angle translation occurs for the motor roll angle as quaternions produce -180-degree to +180-degree angles, whereas the motor encoder and system operation require an angle between 0 and 360 degrees. The rigid body data is then stored in matrices for the main MATLAB script to manipulate. The "labviewData" script, available in Appendix C.8, generates a group and items from the shared variables logged across the OPC server. The item is read, and its values (encoder and IMUs) are logged into matrices for the main MATLAB script to manipulate. After repeating the data log loop for a predetermined amount of time, the pitch, roll, and yaw of the IMUs and roll of the motor flange are extracted from the matrices and saved to a local ".CSV" file. The multicast and OPC clients are then disconnected. This script is run each time a new dataset is required.

4.5 Load Testing

To replicate the effects of the mass and thus the moment of inertia of a prosthetic thigh, a 30.50 cm x 5.08 cm x 0.476 cm hot-rolled steel flat bar is attached to the motor's shaft. In addition to this steel bar, three square pieces of steel denoted as the "load," have been attached to the steel bar 28 cm, (L_{LOAD}), from the motor attachment point. An image of the steel bar with the load attached to the motor shaft is visible in Figure 4.12.

Figure 4.12: Simulated prosthetic thigh load.

The mass of the steel bar is calculated via Equation 4.5 using the dimensions described above and a density of 7.87 g/cm^3.

$$M_{BAR} = L * W * T * \rho \tag{4.5}$$

Where

L = length of bar = 30.5 cm

W = width of bar = 5.08 cm

T = thickness of bar = 0.476 cm

ρ = density of hot-rolled steel = 7.87 g/cm^3

The mass, M_{BAR}, calculates to 580.42 g.

The load includes the bolt, washer, and wing nut securing the pieces together. The load was weighed using a scale, and the resulting mass is 307 g. A total moment of inertia, I, was calculated using Equation 4.6 to include both the steel bar and the load. The moment of inertia for the steel bar is calculated via Equation 4.7.

$$I_{TOTAL} = I_{BAR} + I_{LOAD} \tag{4.6}$$

$$I_{BAR} = \frac{M_{BAR}}{L} \int_0^L r^2 dr \tag{4.7}$$

Where

L = total length of bar = 30.50 cm

r = incremental length of bar

Equation 4.7 can be solved and simplified to create Equation 4.8 where the resulting moment of inertia, I_{BAR}, of 180,000 $g \cdot cm^2$ is found.

$$I_{BAR} = \frac{M_{BAR}}{3} L^2 \tag{4.8}$$

Equation 4.9 is used to calculate the moment of inertia of the "load," I_{LOAD}, at 28 cm with a result of 241,000 $g \cdot cm^2$.

$$I_{LOAD} = M_{LOAD} * L_{LOAD}^2 \tag{4.9}$$

Using Equation 4.6, the total moment of inertia, I_{TOTAL}, is calculated as 421,000 $g \cdot cm^2$. Prosthetics vary significantly in size, shape, mass, and weight distribution. A set of assumptions have been made to produce an equivalent model representative of the moment of inertia expected in a manufactured prosthetic thigh. The thigh prosthesis should have a non-uniform distribution of mass due to heavier components, such as the motor and battery, being located nearest to the axis of rotation. The 1700 g stepper motor and a 10 Ah rechargeable Lithium Nickel Manganese Cobalt Oxide battery have masses of 1700 g and 1800 g, respectively, for a total of 3.500 kg. If the motor and battery masses are uniformly distributed about the first 15 cm of prosthesis length, it can be assumed and thus calculated that a moment of inertia of 262,500 $g \cdot cm^2$ exists. Lighter components such as the embedded controller, motor controller, and power supply should be placed farther away from the axis of rotation

but should not be placed farther than 30.5 cm (12 in). A uniformly distributed mass of the system's lighter components beyond the first 15 cm can be approximated as containing a moment of inertia no greater than 150,000 $g \cdot cm^2$. It is assumed the total moment of inertia generated by the reasonable placement of components, as well as light-weight but rigid prosthesis material such as polyethylene, will not exceed 415,000 $g \cdot cm^2$.

Full system testing and analysis were performed with this load applied to the motor. The results of this testing are examined later in this thesis; they indicate whether the current control system can produce satisfactory results when moving a load with moment-of-inertia characteristics representative of the previously described thigh prosthesis.

A set of assumptions has been collected on the system before analyzing the results of testing. It has been assumed that the load applied to the motor is representative of a thigh prosthesis, which includes the mass and distribution of all system components, and that standing to sitting motion can be accomplished by moving the hip-joint about a single axis. It is also assumed that the prosthesis user would complete the standing to sitting motion once beginning the motion, and do so in 4 to 5 seconds. Further, in the standing position, the leg and torso are relatively vertical but not necessarily at 0 degrees; in the sitting position, the user would be in a seat at a height such that the knee is bent at approximately 90 degrees. The final assumption made before testing is that there are no physical objects inside the model prosthetic's range of motion that can hinder its ability to move as commanded.

CHAPTER 5: RESULTS

5.1 Data Calibration and Collection

Before any data is retrieved from the IMUs and 3-D motion camera system, both must be calibrated. To calibrate an IMU, a VI, seen in Appendix A.5, was created in LabVIEW. Each IMU is independently calibrated while placed inside of its IMU holder. The IMU holder is placed horizontally on a table-top, which is approximately flat and level. With the accelerometer level, upright and calibrated, the x and y axes should theoretically read a value of "0" while the z-axis reads a value of "+16,384". With the IMU horizontal and upside down, the z-axis should read a value of "-16,384". If the IMU were to be placed vertically on the x-axis, the y and z axes would read "0," and the x-axis would read either "+16,834" or "-16,834" depending on orientation. To obtain calibration constants, the VI is run which performs a loop in 1ms iterations to obtain a total of 1000 accelerometer values in the x, y, and z axes. The values for the x and y axes are then averaged, and offset values are produced. The z-axis is also averaged and subtracted from the theoretical value of "16,384". The calibration constants provide linear mid-point offsets for each axis, ensuring the minimum and maximum achievable values are "-16,834" and "+16,834," respectively. These offsets are then added to the main system VI before orientation angles are calculated. Just before collecting validation data, the IMU offsets were as listed in Table 5.1 and applied to the LabVIEW VI.

Table 5.1: IMU offsets from initial calibration.

Axis	Thigh Offset	Torso Offset
X	220	577
Y	488	-719
Z	1347	-680

To calibrate the Optitrack 3-D camera system, the CW-500 calibration wand from Optitrack is used. This wand contains three 12.7 mm markers placed in specific attachment holes for 250 mm wand calibration. The wand can also be used in the 500 mm position; however, the smaller calibration settings should yield better calibration results with the trade-off of longer calibration time. The wand is moved across the entire capture volume in "Figure-8" motions until a sufficient number of samples are collected from each camera. Motive software then takes these samples, performs calculations, and creates a 3-D capture volume in the software depicting the physical camera system. A list of calibration results is displayed that includes the following criteria: overall reprojection error (Mean 3-D and 2-D), worst camera reprojection error, triangulation error, overall wands error, and suggested maximum ray length. The overall 3-D reprojection error is the estimated actual error between the marker's measured location and its physical location averaged across all cameras and is measured in mm. The overall 2-D reprojection error relates to the averaged 2-D error in the measured marker position compared to its real physical position and is displayed in camera pixels. The 2-D reprojection error accumulates for each camera and provides a range of error depending on the particular cameras used to project the image of the marker. The worst camera error displays the highest 3-D and 2-D reprojection error calculated during calibration. Triangulation error refers to the residual offset value and is the most critical characteristic of the calibration results. The residual offset directly relates to the precision of the system and is the offset distance, in mm, between the camera converging rays of a marker during reconstruction. Optitrack suggests "a well-tracked marker has a sub-millimeter average residual value" [38]. The overall wand error is the overall measurement error of the length of the calibration wand during calibration. Lastly, the ray length if the suggested maximum distance a marker should be placed from a camera. The final step in calibrating the capture volume is setting the ground plane. This step generates the global x, y, and

z axes and orientates the calibrated cameras to a flat plane- the floor of the capture volume. Optitrack's CS-200 Calibration Square was placed on the floor in the capture volume and leveled using the built-in adjustment screws. Once the ground plane is highlighted and set in Motive, calibration is complete, and data collection can begin.

The camera system is calibrated for each day data is to be collected as cameras may vibrate, move slightly, and drift throughout the day. The "continuous calibration" process was not introduced until Motive version 2.2, postdating the version of Motive used for this project. Data were collected across multiple days, thus requiring multiple calibrations. Figure 5.1 displays the calibration results used when obtaining encoder data validation data.

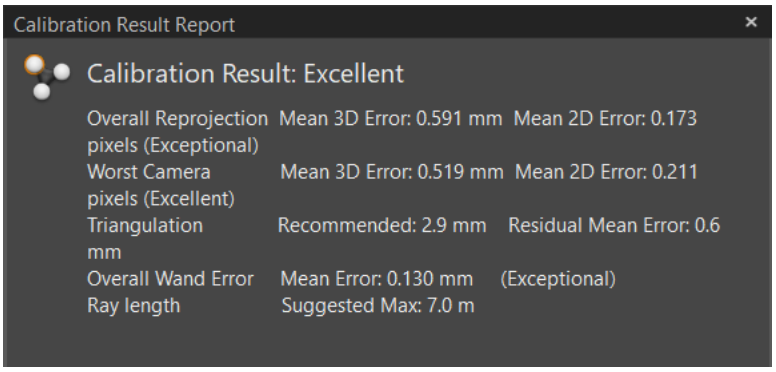


Figure 5.1: Motive calibration results for encoder data validation.

The encoder calibration results were subjectified as "excellent" by Motive. A residual mean triangulation error of 0.6 mm is excellent and provides high confidence in the accuracy of a marker position. The worst camera in this calibration provides a mean 3-D error of 0.519 mm.

The Motive calibration results used for obtaining IMU validation data, IMU motion and stationary data, and full system operation data can be viewed in Figure 5.2.

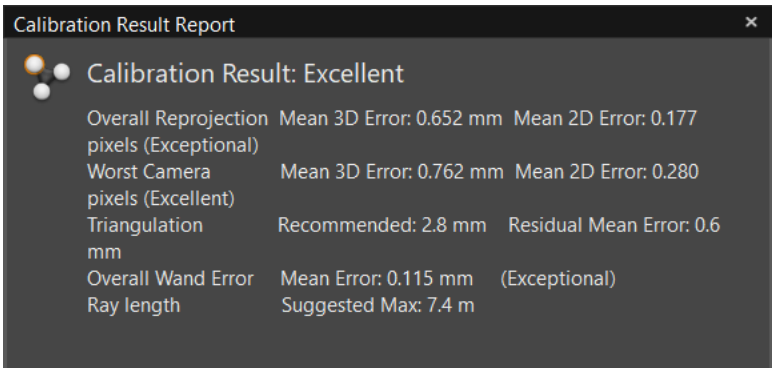


Figure 5.2: Motive calibration results for the remaining data capture.

For testing purposes, a PCA9615 differential I^2C breakout board was needed. Complete I^2C signal degradation can occur within 1 meter of the slave device. During testing, the system, including the myRIO, was placed a distance away from the test area to ensure it would not inadvertently hinder body motion, thus producing inaccurate data. The PCA9615 converts the I^2C data signals into four differential signals that are then sent across a 4.3 m long RJ45 CAT-6 cable to another PCA9615 that is wired into the myRIO. The SDA and SCL signals each split into two differential signals. At the receiving PCA9615, each set of differential signals is then subtracted, producing either a digital "high" with greater magnitude than the originating signal or a digital "low" with no magnitude. This scheme allows long-distance transmission of serial data with a much cleaner resulting signal for the embedded system. The differential converter measures 4.5 cm x 2.5 cm and is wired 4 cm away from each IMU and about the same distance away from the myRIO. This module was taped to the body alongside the IMU to allow the least signal degradation as possible. Figure 5.3 shows the complete test module.

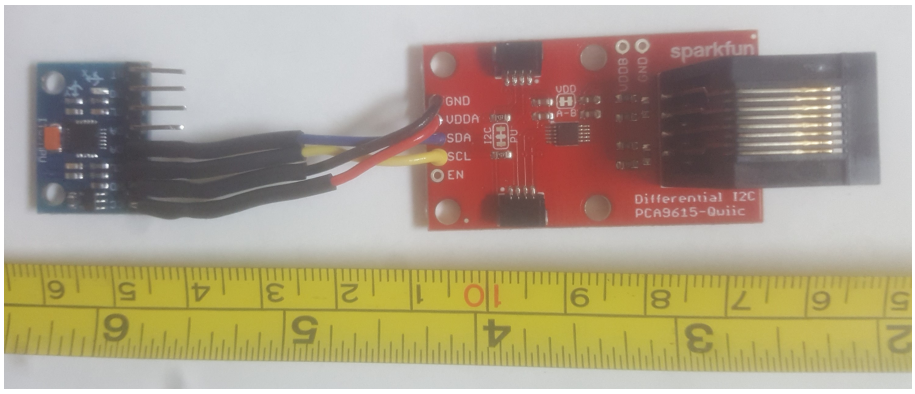


Figure 5.3: PCA9615 connection to GY-521

During all tests, motor position and thigh and torso IMU data are saved to various .csv files for analysis. Motor position in degrees is logged and saved from the encoder and camera system. Thigh and torso IMU pitch positions are logged and saved from LabVIEW calculations and the camera system. For full system testing, the net angle between the thigh and torso IMUs is calculated in LabVIEW and saved to a .csv alongside camera and encoder data.

5.2 Data Validation

Before input and output data of the control system can be analyzed, the digital data obtained by both IMUs and the encoder must be proven accurate. To validate digital data from the motor encoder, 10 in-motion and 28 stationary datasets were taken. The results of the encoder in-motion data validation will be analyzed first. Figures 5.4 and 5.5 display datasets 1 and 2, respectively, from the collection of 10 in-motion datasets. Figure 5.4 displays the encoder position vs. motor shaft position seen from the encoder and camera system with motion starting at 45 degrees and moving to 315 degrees. Figure 5.5 displays data from the motor while it was moving from 315 degrees to 45 degrees.

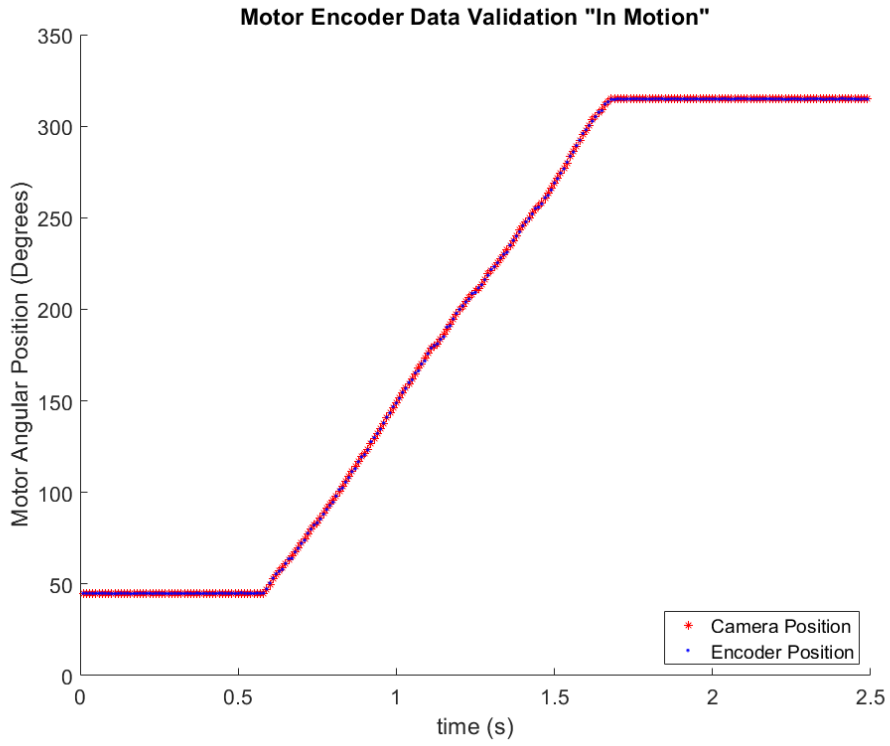


Figure 5.4: Motor position when rotating from 45 to 315 degrees.

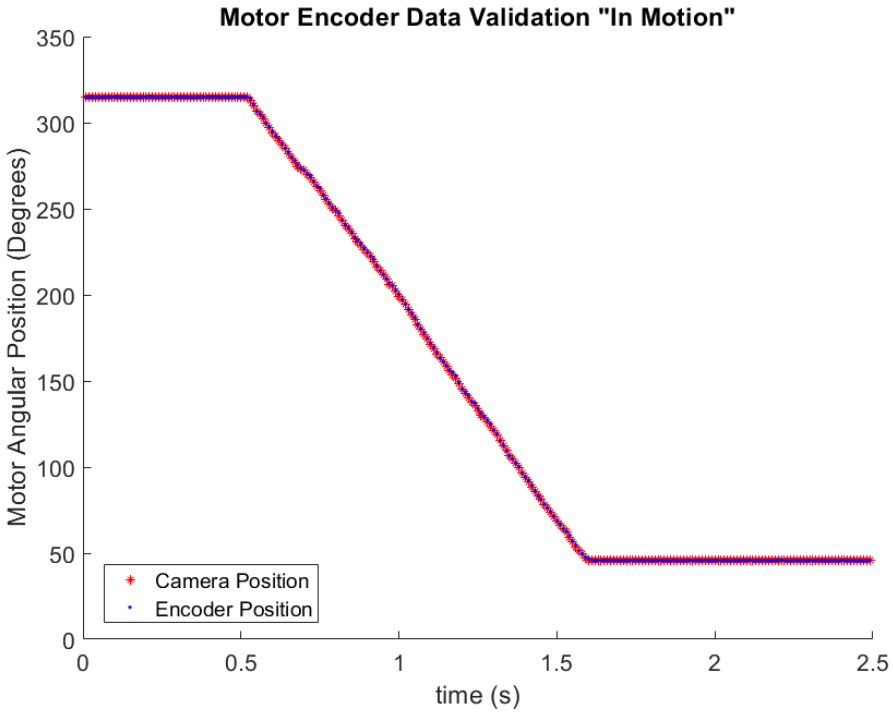


Figure 5.5: Motor position when rotating from 315 to 45 degrees.

In both figures, the encoder position overlaps the camera position at nearly every data point. From this perspective, it appears that the encoder is measuring the physical position of the motor shaft exceptionally well. To identify point-to-point differences in the dataset, Figure 5.6 was generated, which displays the difference in measurement between the encoder and camera system at every degree point measured by the encoder during the in-motion data validation test.

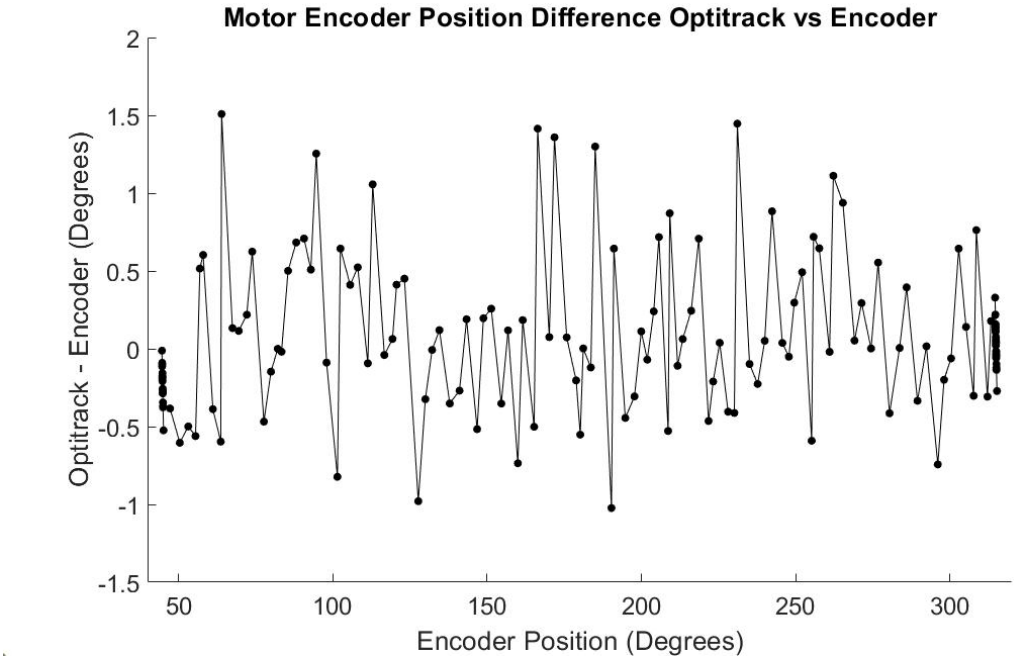


Figure 5.6: Difference in motor position between encoder and cameras.

The majority of the encoder data is within 0.25 degrees of the camera data, with the mean difference of dataset equal to 0.0092 degrees. There seems to be no apparent oscillation in the distribution of the data, and the difference in data appears to be noise, injected by both the encoder and camera system. Table 5.2 displays the mean difference between the encoder and camera system for all 10 in-motion datasets. Half of the datasets produced a mean difference of less than 0.2 degrees, the rated accuracy of the encoder [35]. The remaining datasets produced mean differences between 0.2158 degrees and -0.3117 degrees. As explained previously, the control system is designed to not correct if the commanded position of the motor is within 0.5 degrees of the

measured position. Given this control specification, the listed differences in position above 0.2 degrees are extremely small for this thesis' application and were deemed negligible.

Table 5.2: Motor in-motion difference across all datasets.

Dataset	Minimum Difference	Maximum Difference	Mean Difference
1	0.000263	1.5085	0.0092
2	0.000685	2.0223	-0.3117
3	0.000701	1.9500	0.0953
4	0.001500	2.4292	-0.2158
5	0.000049	1.7676	0.0732
6	0.000456	2.2682	-0.2618
7	0.000258	1.4916	0.0837
8	0.001500	2.5019	-0.2343
9	0.000557	1.8552	0.1111
10	0.000008	2.1038	-0.2773

A linear regression analysis was performed on each of the 10 datasets. The resulting graphs yielded a very similar line of best fits with a tight linear distribution concluding the data is normally distributed. A t-test was performed on the data with resulting p-values of 0 and R-Squared values of 1. The resulting regression graph of dataset 1, plotting encoder vs. camera position, can be viewed in Figure 5.7.

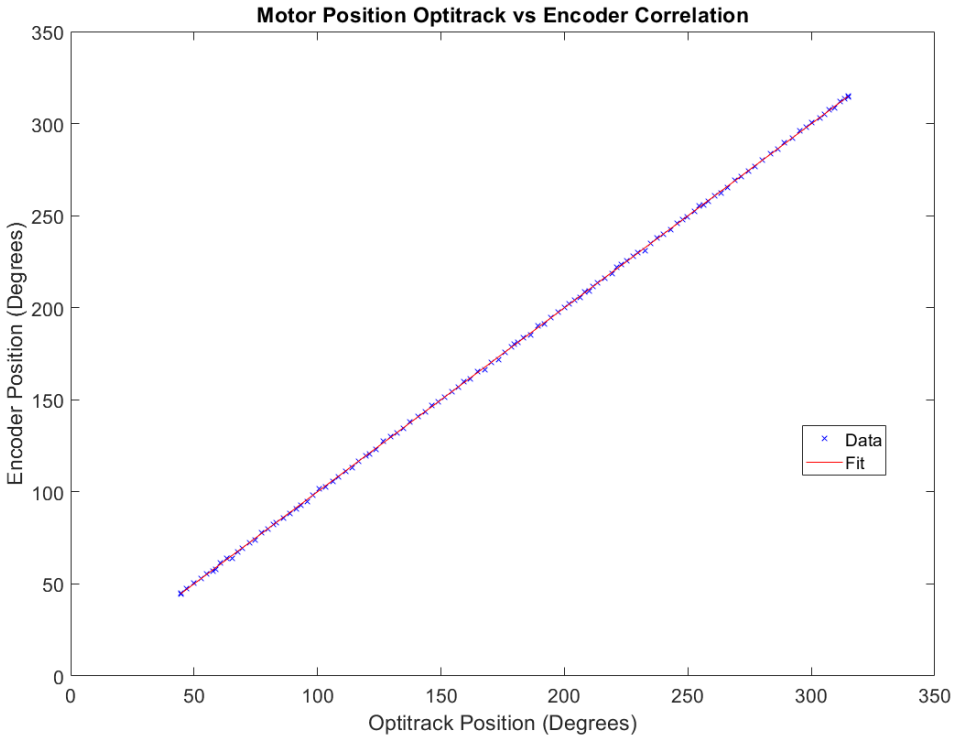


Figure 5.7: Regression curve of motor position measurements.

The width of the confidence bounds (provided at a 95% confidence interval) is ±0.5 degrees, practically overlapping the best fit line in Figure 5.7. This result indicates that, on average, the encoder's measured position is within 0.5 degrees of the shaft's position. The mostly straight line visually displays the line of best fit with a slope very close to "1". Performed in MATLAB using the *fitlm* regression tool, an analysis of all 10 in-motion datasets was performed with the resulting data viewed in Table 5.3.

Table 5.3: Motor in-motion data validation regression results.

Dataset	Start Position (Degrees)	Stop Position (Degrees)	Regression Fit $y = \beta_0 + \beta_1 x$	RMSE (degrees)
1	45	315	y = 0.1268 + 0.9993x	0.384
2	315	45	y = 0.2758 + 1.0002x	0.453
3	45	315	y = -0.0825 + 0.9999x	0.432
4	315	45	y = 0.1786 + 1.0002x	0.473
5	45	315	y = 0.0918 + 0.9991x	0.391
6	315	45	y = 0.2990 + 0.9998x	0.510
7	45	315	y = -0.0125 + 0.9996x	0.413
8	315	45	y = 0.2624 + 0.9998x	0.514
9	45	315	y = -0.0511 + 0.9997x	0.408
10	315	45	y = 0.2752 + 1.0000x	0.470

For each dataset, the calculated r-squared values were automatically rounded to "1" by MATLAB. This outcome indicates that the regression-fit equations represents all data from the dataset. Also, all datasets display an extremely low p-value, rounded to "0" in MATLAB as their magnitudes were extremely small (less than 10^{-30}), which corresponds to a very high t-value for the coefficient (x) and explains the statistical significance of "x". The null hypothesis of the coefficient (x) being statistically insignificant (0) is rejected, thus providing evidence that the data y (encoder position) can be explained by the data x (camera position) to a high degree. With the slopes of the regression line being nearly "1", it can be concluded that encoder position data is equal to position data measured by the camera system. This result, along with the graphical analysis of Figure 5.6, proves that a strong relationship between the encoder and the physical motor's shaft position measured by the camera system exists. This data successfully validates the in-motion digital data of the encoder. All 10 data samples displayed similar results. The full set of data was individually plotted to show the difference in the measurement ability of the encoder vs. the camera system; it can be viewed in Appendix D.1.

To validate stationary data of the encoder, the motor was positioned at approxi-

mately 45 degrees. The digital and physical position of the motor's shaft was logged for 2.5 seconds, and its data was analyzed. This process was repeated in increments of approximately 10 degrees for a total of 28 datasets. The mean encoder position for each dataset, and its variance, were compared to that of the mean camera system's measured position and variance. Figure 5.8 displays the resulting plot of dataset 1, where the motor was set to approximately 45 degrees.

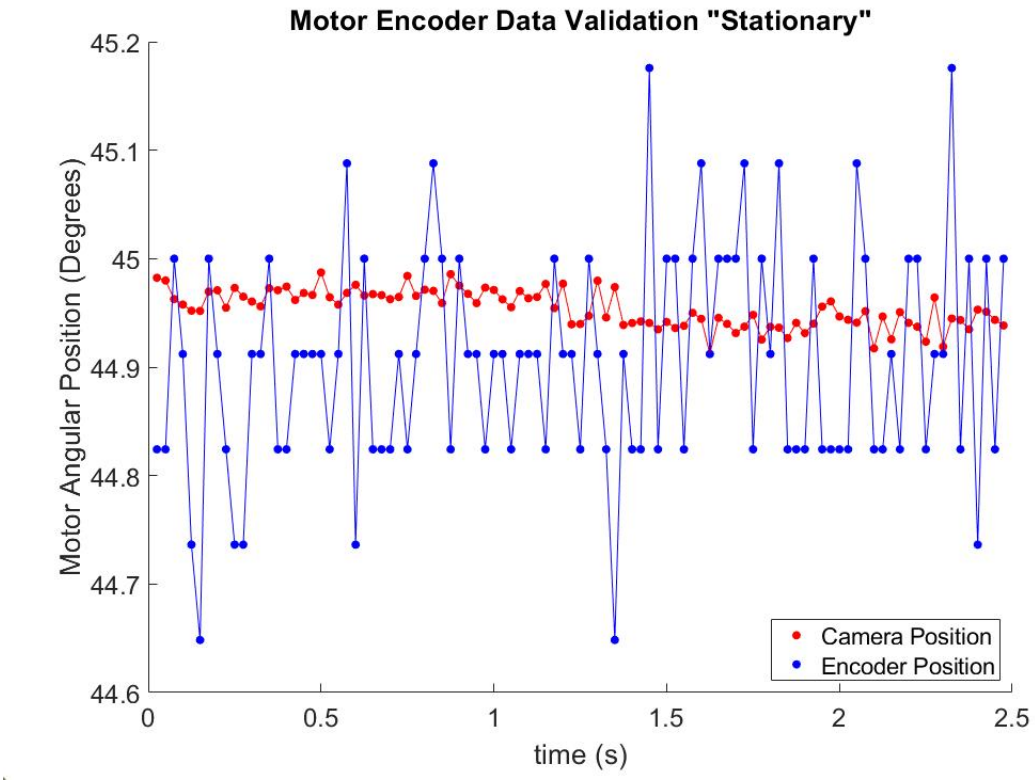


Figure 5.8: Stationary position of motor seen by encoder and cameras.

The encoder mostly fluctuated between 44.8 and 45 degrees, with some outliers reaching 44.65 and 45.2 degrees. A mean angle was observed at 44.904 degrees, with a variance of 0.011 degrees. The camera data contained much less variance and averaged 44.954 degrees with a variance of 0.006 degrees. To validate the encoder data. the camera data must contain less variance. This dataset produced a mean difference in encoder vs. camera position value of 0.0499 degrees, which is a percent difference of 0.11%. This analysis was performed for each of the 28 stationary motor

datasets; a summary of the data is displayed in Table 5.4 as well as the difference in means for each mean angle of the encoder, in Figure 5.9. The results in their entirety can be viewed in Appendix D.2.

Table 5.4: Summary of collective stationary motor validation data.

Maximum Camera Variance:	0.005943 $degrees^2$
Maximum Encoder Variance:	0.016803 $degrees^2$
Average Mean Difference:	0.130173 degrees
Maximum Mean Difference:	0.527877 degrees
Maximum % Difference:	0.495424 percent

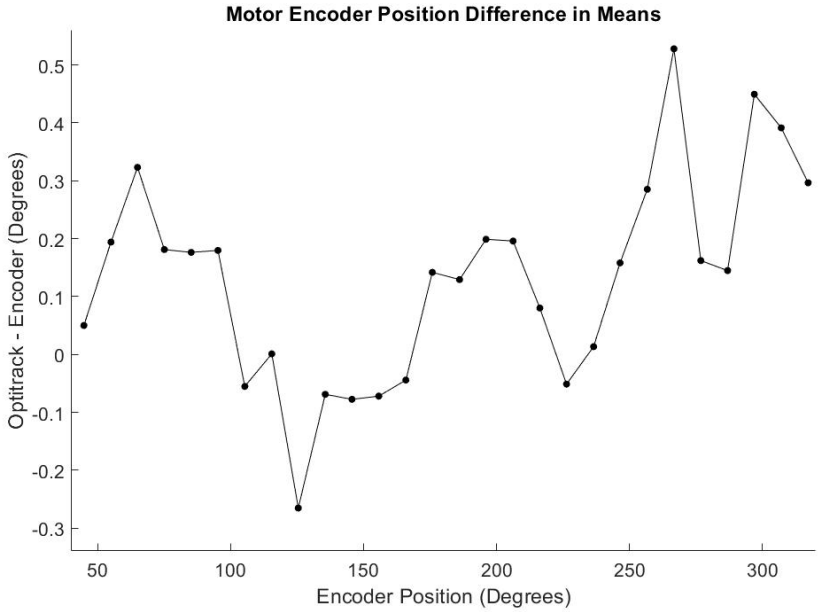


Figure 5.9: Difference in means of all stationary motor datasets

It is important to note that the variance of the camera system for each sample is much smaller than that of the encoder, which is necessary for the validation of measurement data. The maximum camera and encoder variances were calculated as approximately 0.006 $degrees^2$ and 0.017 $degrees^2$, respectively. The maximum difference in means was 0.528 degrees, which occurred at approximately 267 degrees (dataset 23). The graph of this dataset can be viewed in Figure 5.10.

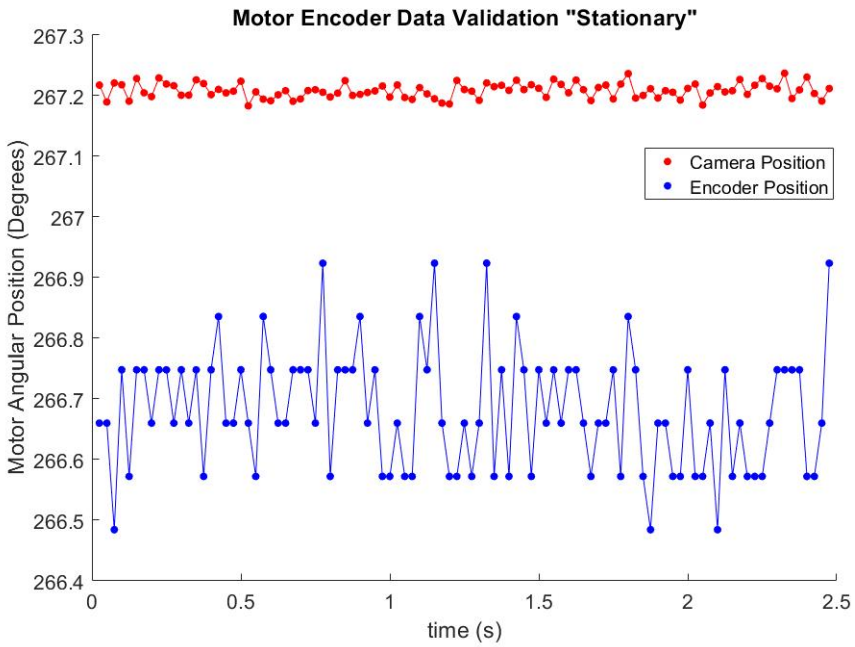


Figure 5.10: Motor stationary data from dataset 23.

A slight but noticeable difference in data can be seen in the above figure. This difference can likely be explained by a slight imbalance in the rotation of the motor's shaft and dc noise in the system. This more considerable difference is within the range of observed differences in the position seen during the in-motion tests viewed in Figure 5.6. It can be implied that the offset which occurred at approximately 267 degrees of rotation did not have a noticeable effect on the in-motion validation of data and the final full system results explained later in this document. The average mean difference across all stationary datasets was 0.130 degrees, which is less than the AMT203-V encoder's rated accuracy of 0.2 degrees [35].

Next, the analysis of in-motion and stationary IMU validation data was performed. As with the motor, the in-motion analysis was performed first, beginning with Figure 5.11, displaying the results of dataset 1 of the thigh position, and Figure 5.12, displaying the results of the torso position. The motion for this test involved standing upright, and moving the thigh up from the floor towards the abdomen, and then back down to the floor.

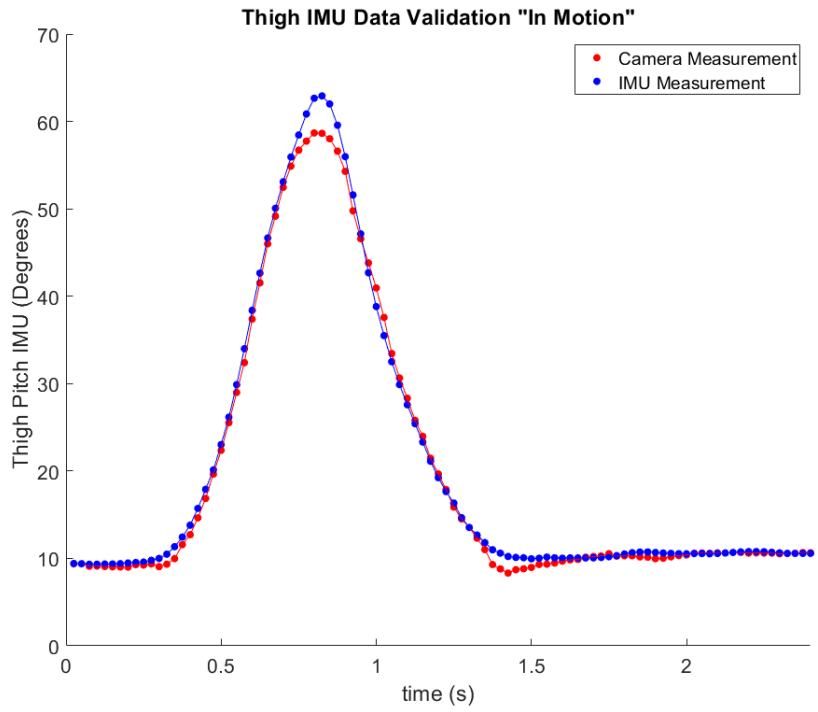


Figure 5.11: In-motion data of thigh position from dataset 1.

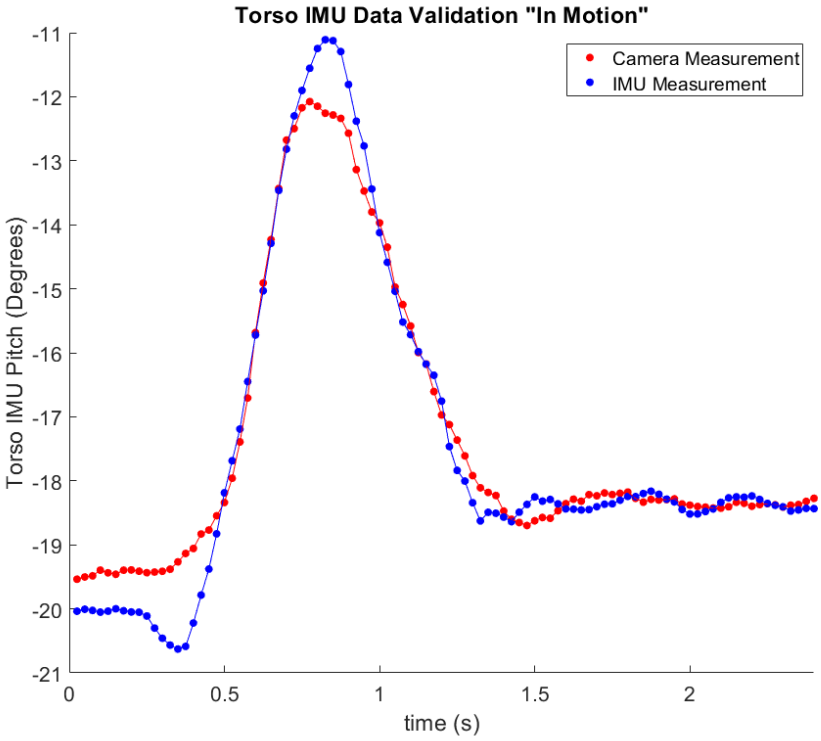


Figure 5.12: In-motion data of torso position from dataset 1.

Both thigh and torso IMUs perform similar traces to that of the camera system except for the peak when the thigh was at its highest point and the torso was at its most vertical position. Though data from the torso looks more extreme than from the thigh, its scale is much smaller; the IMU data differs from the camera system data a maximum of 1.5 degrees. The measured position of the thigh IMU is very close to the camera system's when the thigh is steady, lifted, and lowered. The peak position of the thigh differs by nearly 4 degrees. Stationary data, obtained later in this section, contradicts this error margin. The departure occurs because when the thigh is at it's highest point, two markers placed on the upper part of the IMU holder become slightly hidden in by the body, reducing the accuracy of the visible markers and overall position of the rigid body. This significant difference is explained by poor marker tracking by the camera system. This error is consistent across all 10 IMU in-motion data validation datasets but does not appear in any of the 10 full system motion trials. The full system motion differs slightly from this motion when the thigh is at its highest point until the torso begins to recline, providing a better view of the thigh IMU holder markers.

A regression analysis was performed on both IMUs. The analysis results from dataset 1 of the thigh and torso can be viewed in Figure 5.13 and 5.14, respectively.

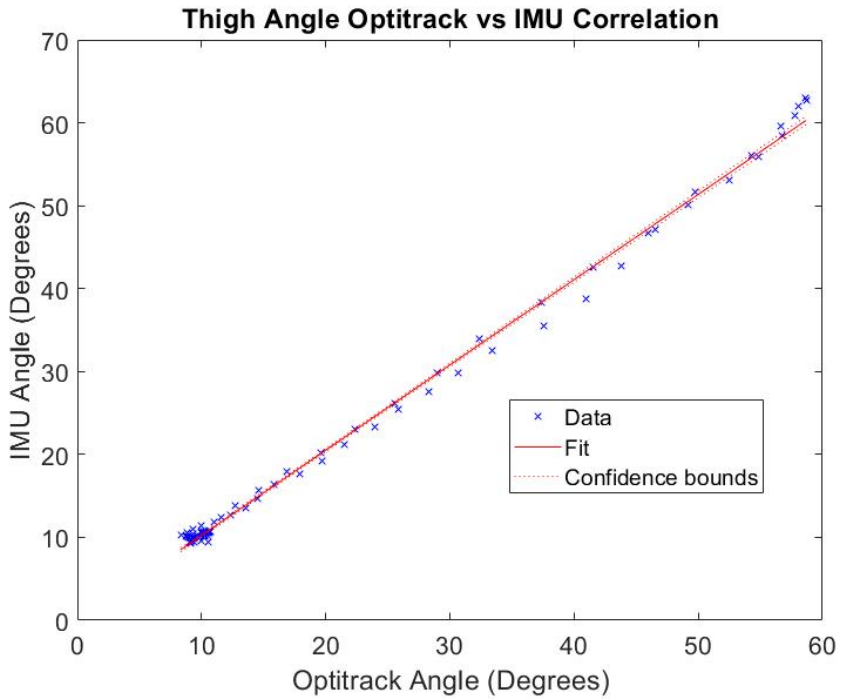


Figure 5.13: In-motion regression fit of thigh position from dataset 1.

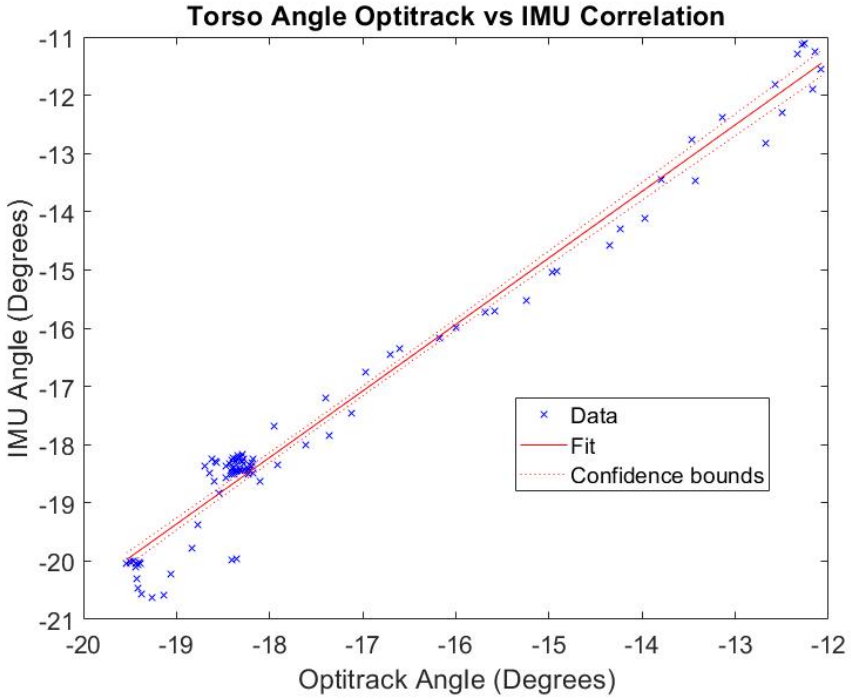


Figure 5.14: In-motion regression fit of torso position from dataset 1.

Thigh IMU regression results display a line of best fit with a slope in the range

of 0.9995 - 1.0452. The torso IMU regression results display a line of best fit with a slope in the range of 1.0217 - 1.2331. Visual analysis of Figures 5.13 and 5.14 allows the assumption of data standardization to be made based on line-of-best-fit slopes, as indicated previously; utilizing MATLAB's *lmfit* tool provided r-squared values for the thigh IMU in the range of 0.991-0.997 and the torso IMU in the range of 0.926-0.990. All thigh IMU r-squared values indicate at least 99% of angles measured by the camera system can be explained by angles measured by the IMUs when iterated through the generated regression- Equation 5.1.

$$y = \beta_0 + \beta_1 x \tag{5.1}$$

Where

y = limb orientation from the camera

β_0 = regression line y-intercept

β_1 = the slope of the regression line

x = the limb orientation from the IMU

Almost all torso IMU r-squared values surpassed 95% indicating with high confidence (above 95%) that torso IMU measurements explain measurements by the camera system. P-values for all datasets were rounded down to "0" by MATLAB's *fitlm* tool as all values were smaller than 10^{-30}. It can be concluded that with such small p-values, a correlation exists between IMU data and camera data for these 10 trials. Because β_1 values are close to "1", it can be concluded that IMU measurements are equal to camera measurements. It can be further concluded that net in-motion IMU data, measuring human hip-joint angle, is valid and equivalent to the physical net angle of the hip-joint measured by a 3-D camera system.

To provide further validation of IMU data, stationary data from the IMU is generated in two datasets and compared to camera system data. Each of the two datasets

was created with both IMUs placed in a different stationary position for 2.5 seconds. The first dataset involved placing the thigh IMU at approximately 90 degrees and the torso IMU at approximately -90 degrees. Their physical position was tracked by the camera system and digital position tracked by LabVIEW. Figure 5.15 shows the position of the thigh IMU, and Figure 5.16 shows the position of the torso IMU during this test.

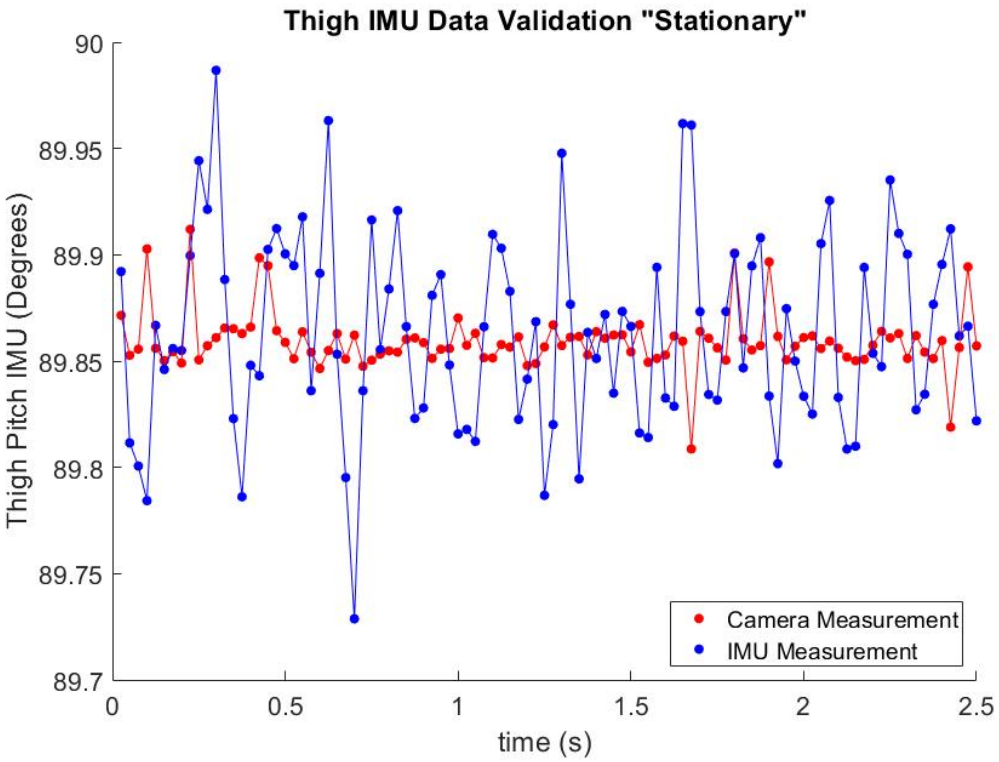


Figure 5.15: Test 1 of stationary position of thigh IMU.

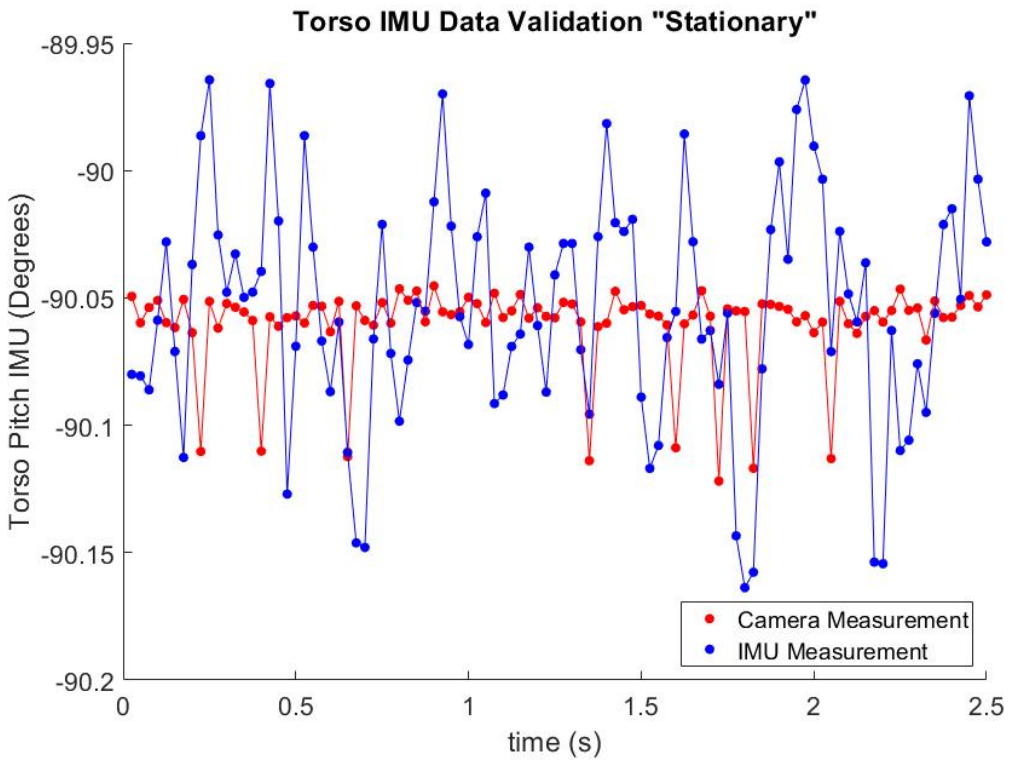


Figure 5.16: Test 1 of stationary position of torso IMU.

Both thigh and torso IMUs display a tight mean in position around the same position mean measured by the camera system. In areas where the IMU has measured a higher value at nearly 0.1 degrees away from the average position, the camera system does not. The same can be said when the IMU measures lower values. It can be concluded that oscillation in IMU values is most likely not sensed motion but simply error in the IMUs accuracy or that the IMUs provide much higher sensitivity than what the camera systems can measure and that the IMUs are vibrating slightly on the table. IMU oscillation around the mean position appears random with an entire range of approximately 0.15 degrees without including outlier measurements. With 5 outlier measurements included, the entire range is approximately 0.25 degrees. Both IMUs were calibrated at the 90 and -90 degree position, thus the results of this first stationary dataset are within expectations.

The second stationary data validation test was performed with both the thigh and

torso IMUs placed at approximately 0 degrees.

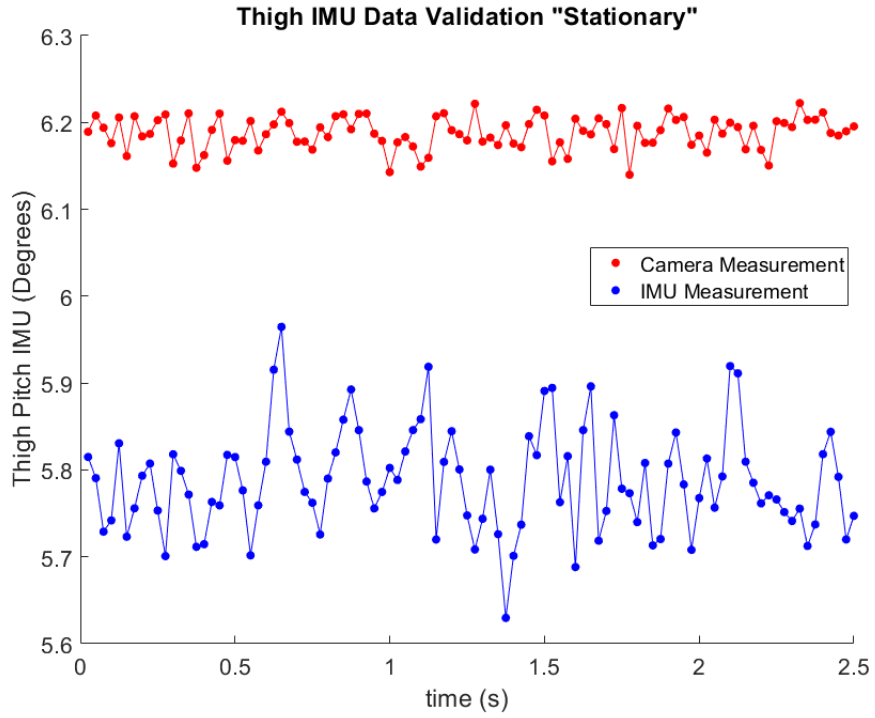


Figure 5.17: Test 2 of stationary position of thigh IMU.

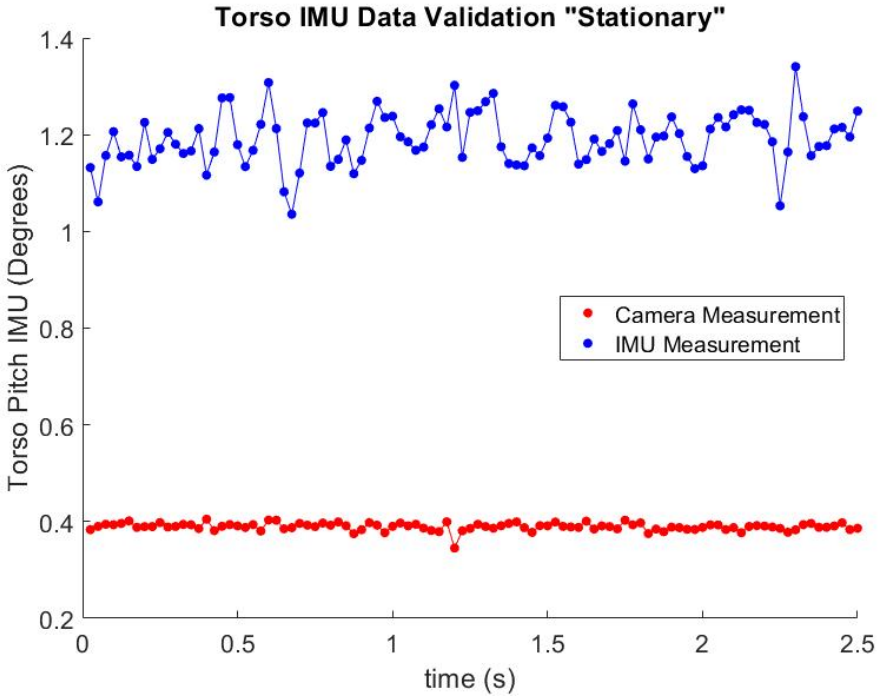


Figure 5.18: Test 2 of stationary position of torso IMU.

The results of the second test show an obvious steady error in measurement between the IMUs and the camera system. Both IMUs display noise in an equivalent range to the first test; however, the mean difference in the position of the thigh IMU is approximately 0.4 degrees, and the mean difference in the position of the torso IMU is approximately -0.8 degrees. When calculating the net IMU angle for system operation, an error of 1.2 degrees could be observed. Though this value is quite significant, this amount of error was not apparent in full system motion testing, which is displayed further in this document. It is important to note that IMU measurements are consistent in their location, and with a proper offset setting of +0.6 degrees, an overall constant net IMU error of +0.6 degrees could be observed across both stationary datasets. An error of 0.6 degrees is close to the forced maximum error of the control system (0.5 degrees) and should be sufficiently small for proper system operation. For numerical analysis, a summary of both stationary datasets has been provided in Table 5.5.

Table 5.5: Summary of stationary IMU validation data.

	Thigh Data					**Torso Data**				
	Camera Data		**IMU Data**		**Comparison**	**Camera Data**		**IMU Data**		**Comparison**
Dataset	Mean Angle	Variance	Mean Angle	Variance	Difference in Means	Mean Angle	Variance	Mean Angle	Variance	Difference in Means
1	89.859	0.00019	89.863	0.00212	-0.0039	-90.06	0.00027	-90.05	0.00213	-0.00622
2	6.188	0.00036	5.787	0.00358	0.4007	0.39	0.00006	1.19	0.00298	-0.80253

5.3 Full System Data Analysis

The purpose of this section is to clearly outline the reliability and accuracy of the encoder and IMUs. Their positions and motion have been validated previously by the 3-D motion camera system. This section presents the final motion of encoder vs. IMU and compares the calculated net angle of the IMUs with those measured by the camera system. To perform full system operational tests, net IMU pitch and encoder rotational position data were logged in conjunction with camera data and

plotted to display position over time. The body began in an upright standing position, transitioned to a sitting position, and then a slight but fluid reclination occurred. The entire motion was logged during a 5-second interval and performed a total of 10 times with each trial generating a unique set of data. The same MATLAB script used to obtain data validation input was used to obtain full system data with the addition of a few lines of code to ensure all data was logged to appropriate files. A full system analysis script was also written in MATLAB to analyze all logged data and generate visuals for data comparison, which can be viewed in Appendix C.1.

The IMU position is the net measured position of the right hip angle determined by Equation 3.1 and is designated as the "commanded angle." The encoder angle is the measured output position of the system and is designated as the "measured angle." The commanded vs. measured angles of Trial 1 can be viewed in Figure 5.19.

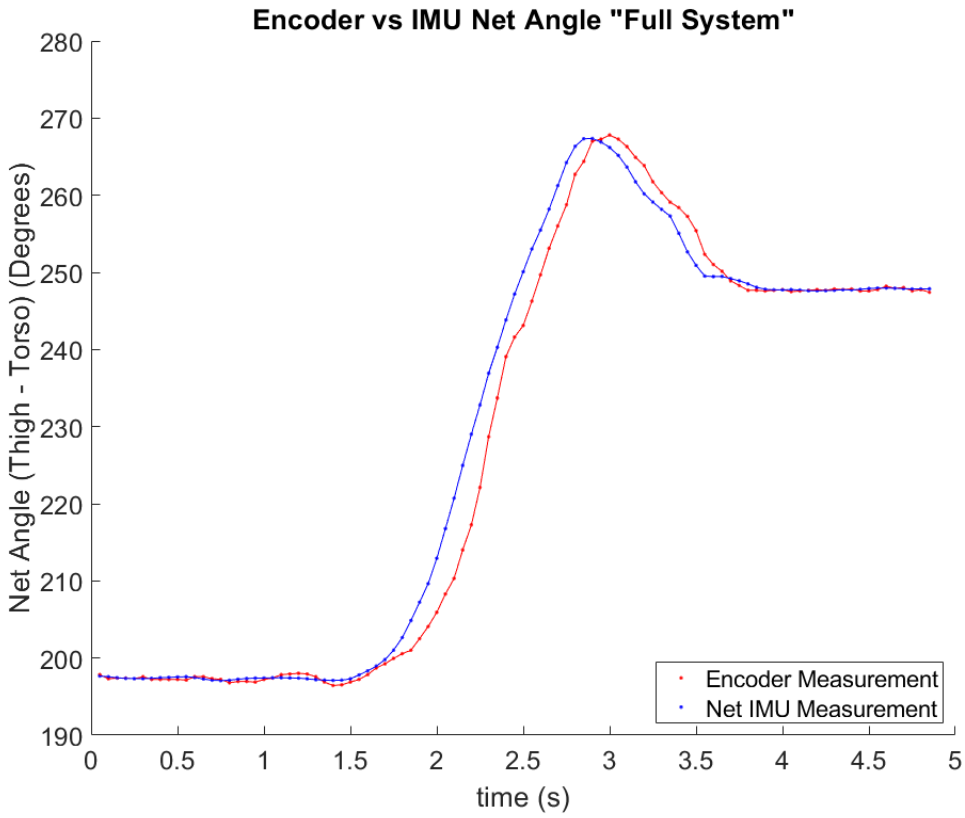


Figure 5.19: Trial 1: Full system operation - IMU vs. encoder position.

The measured angle follows the commanded angle well with apparent delay. Some amount of delay is expected as the control system performs various processes and calculations, which all require small amounts of time to complete. The measured delay during this trial is approximately 90 ms. The peak commanded angle during full system motion was logged at 267.4 degrees. The peak measured angle was measured at 267.8 degrees, a difference of 0.4 degrees. After the motion was performed, the steady-state commanded angle determined by the IMUs oscillated between 247.6 and 248 degrees while the measured angle of the motor fluctuated similarly with a maximum difference of approximately 0.4 degrees as viewed in Figure 5.20.

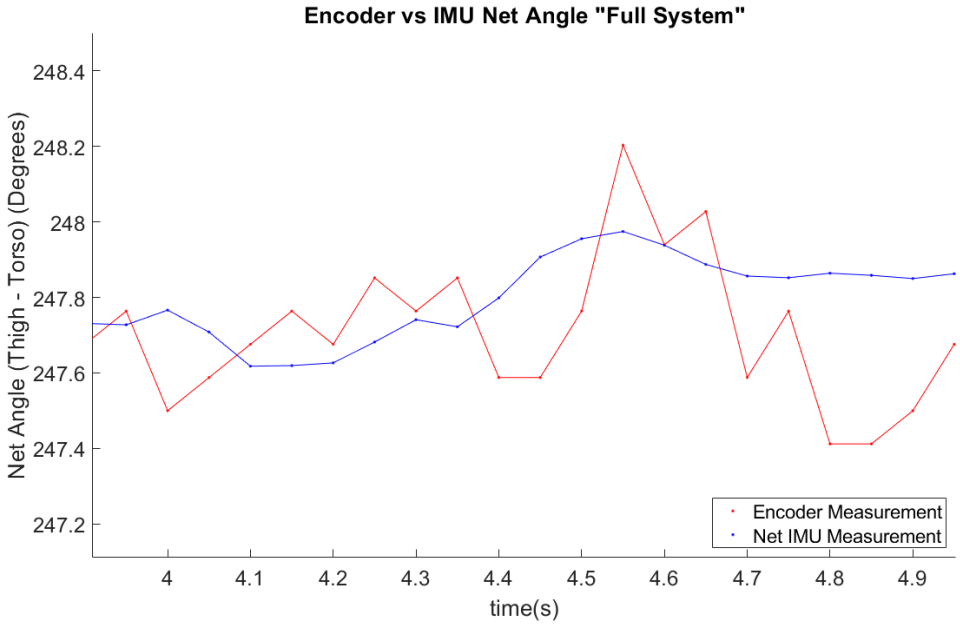


Figure 5.20: Trial 1: Full system operation Trial 1 end position.

As determined during the data validation tests, at approximately 246.6 degrees, the encoder's measured position of the motor shaft was, on average, 0.158 degrees less than the physical position of the shaft. Also, during data validation tests, the net IMU position was determined to be as much as 1.2 degrees less than what the camera system determined to be the actual angle. With these tolerances included, the commanded and measured angles observed during full system motion are within

the expected range of difference. Figure 5.21 displays the net IMU pitch angle vs. the net pitch angle measured by the camera system. Disregarding the delay, the net IMU angle appears to match the camera angle nearly identically with a constant offset error of approximately 0.5 degrees. This observation further validates the digital data from the IMUs.

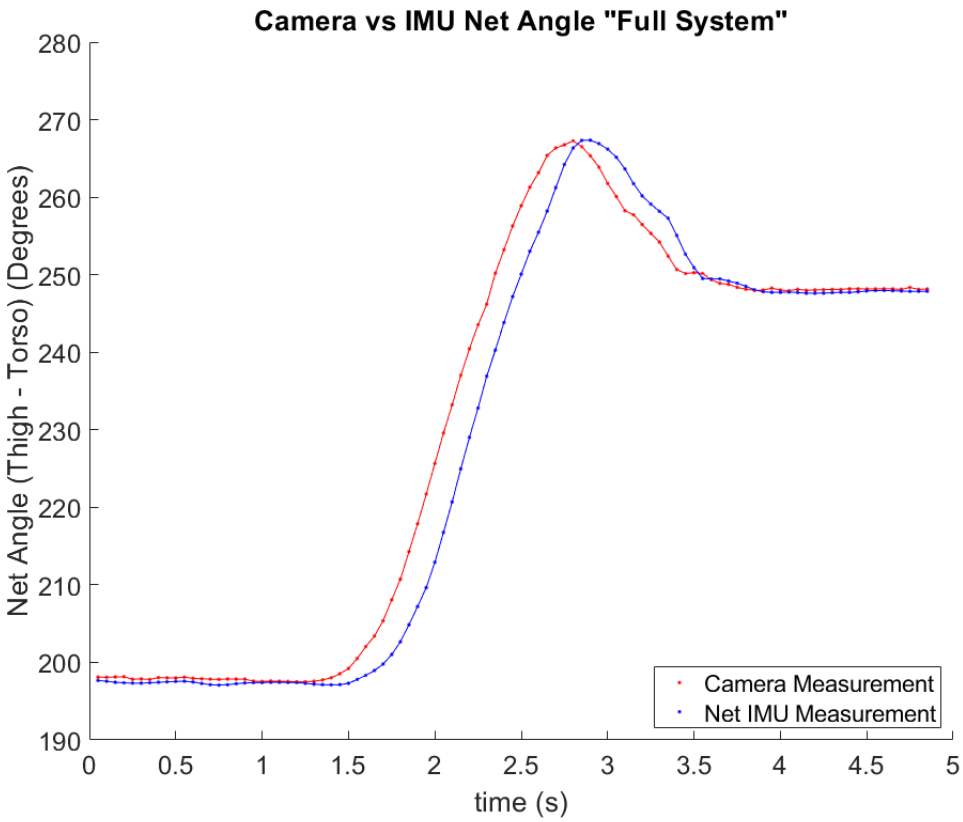


Figure 5.21: Trial 1: Full system operation - IMU vs. camera position.

This analysis was performed for all 10 full system motion datasets with remarkably similar results from each trial. For visual comparison only, the result of Trials 2 has been included below in Figure 5.22 with the results of all trials included in Appendix E.

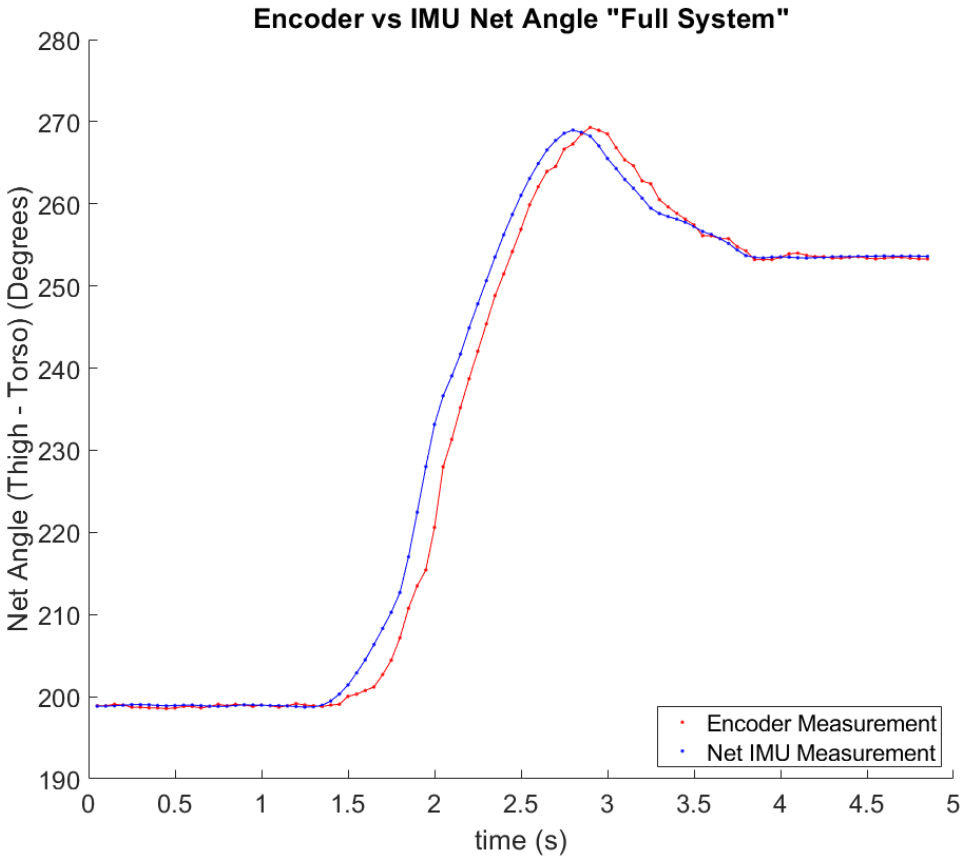


Figure 5.22: Trial 2: Full system operation - IMU vs. encoder position.

If system delay is ignored and the output is shifted to align with the input, both traces (commanded and measured angles) compare nearly identically. Figure 5.23 displays this result where the output has been shifted to align with the input in Trial 1 of full system motion.

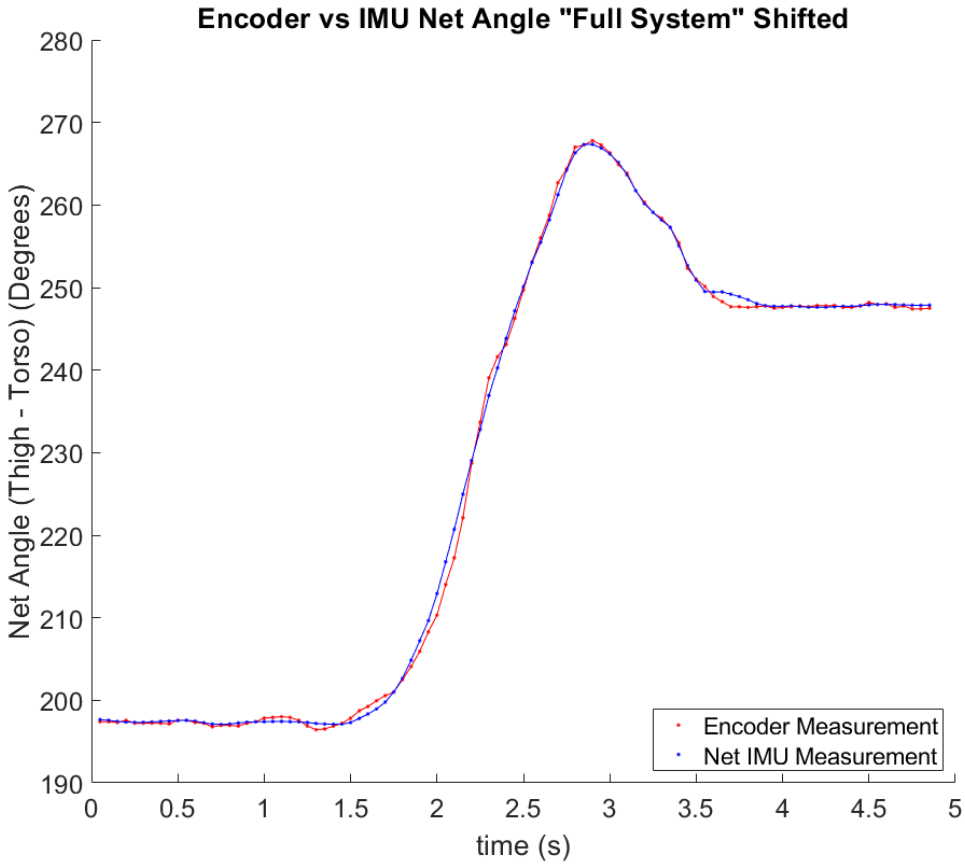


Figure 5.23: Trial 1: Full system operation with delay shifted.

5.4 No-Load Transfer Function Modeling

Before the system could be modeled, the data needed to be set up in a fashion that allows MATLAB to perform computations. Within MATLAB, an *iddata* object was created from the logged full system operation data tests. An *iddata* object is a matrix collection of input and output data alongside a time interval. This data object can then be used as a package for MATLAB algorithms. The *iddata* object used net angle data from the IMUs as the input (u), the motor's shaft position determined by the encoder as the output (y), and a sampling time of 15 ms as determined by the period between data points in the raw data log. The raw data log, used for determining the transfer function and system stability, is different from the data log used to generate motion characteristics and position validation from the previous sections of

this document. The raw data log was taken directly from LabVIEW and does not contain any camera data. Data was able to be logged in 15 ms intervals, quicker than the logged data used earlier for validation. As the digital encoder and IMU data has been previously validated, using this raw data log allows for the data analysis of significantly more precise data. The data used was obtained by performing an additional full system motion with net IMU pitch and encoder position data logged. Figure 5.24 displays the net IMU pitch (commanded) angle vs. encoder (measured) angle during the precise full system data log.

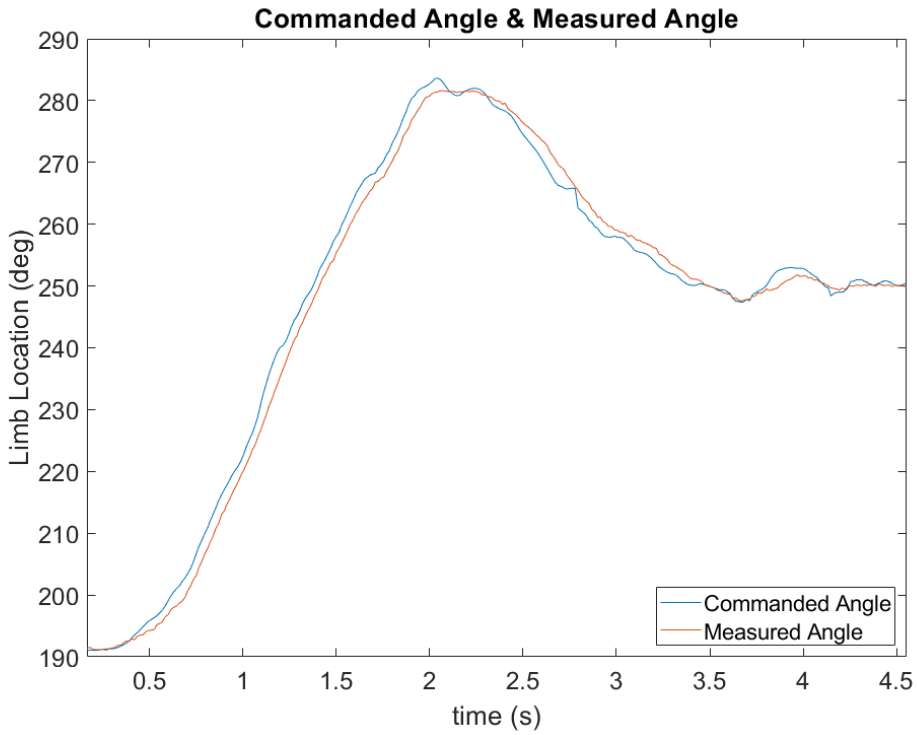


Figure 5.24: Full System Motion Data for System Modeling

The flow of data in Figure 5.24 has a similar shape to full system motion figures displayed previously as the same motion was performed during this data log but with much higher precision, as discussed previously in this section. More significant oscillations can be seen beyond 3.5 seconds on the graph. These oscillations occurred just after the torso was reclined; it was difficult to hold the torso in the same position

without back support. Though not ideal, this motion should be tracked and followed by the motor as the torso is moving, thus creating a change in the hip angle. This motion is similar to what could be observed if a subject were to sit in a typical reclining chair affixed with a spring reclining mechanism. It can also be observed that there are minor changes in the commanded angle that are not reflected in the measured angle, such as near the peak of the motion and during reclining at t=2600 ms. The system appears to filter out higher frequencies and smooth the output motion of the system. This observation is further analyzed later in this section.

To model the system, a discrete-time state-space algorithm in MATLAB, called *n4sid*, was used. The *n4sid* algorithm uses input and output data from the *iddata* object with a sample time of 15 ms. When performed on the no-load data, the following Hankel singular values are displayed with the optimal system order colored in red, as viewed in Figure 5.25.

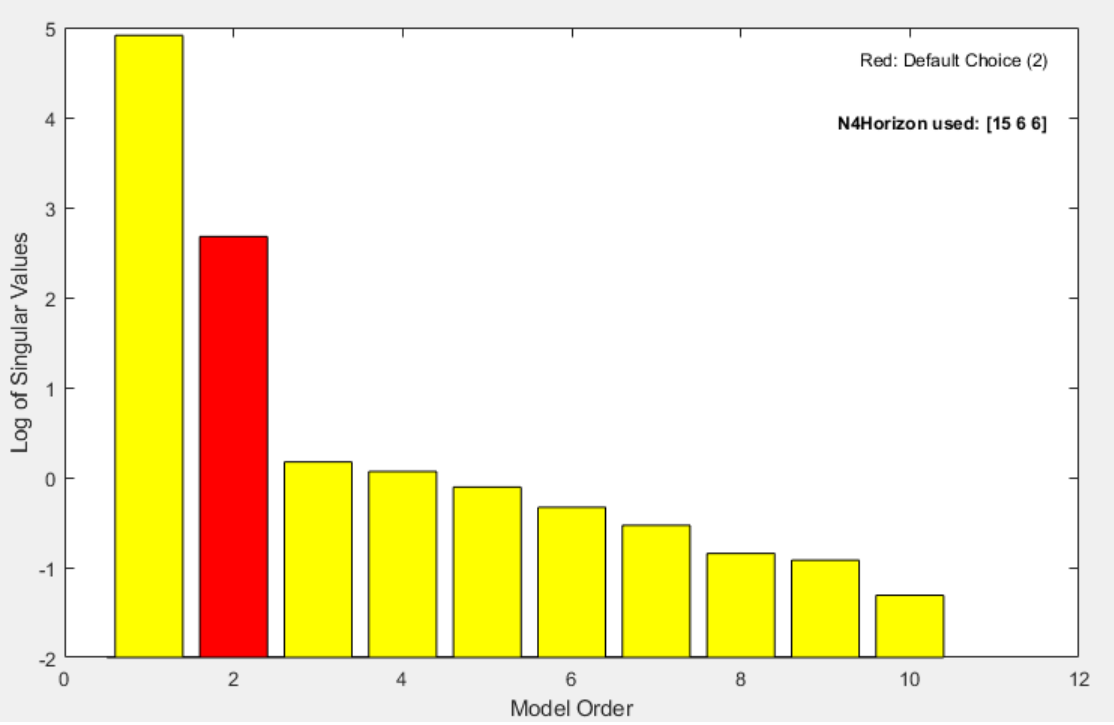


Figure 5.25: MATLAB results used to determine optimal system order.

After applying the chosen value of order "2", MATLAB performs the *n4sid* algo-

rithm to solve for a discrete-time state-space model with the results listed as follows:

$$x(t+Ts) = Ax(t) + Bu(t) + Ke(t)$$

$$y(t) = Cx(t) + Du(t) + e(t)$$

$$A = \begin{bmatrix} 0.9241 & 0.05386 \\ -0.2181 & 0.7896 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.001437 \\ -0.01378 \end{bmatrix}$$

$$C = \begin{bmatrix} -61.56 & -0.4586 \end{bmatrix}$$

$$D = 0$$

$$K = \begin{bmatrix} -0.009232 \\ -0.04711 \end{bmatrix}$$

This state-space model completed with a fit to estimation data (prediction focus) of 99.22%, a final prediction error (FPE) of 0.04365, and a mean root square (MSE) of 0.04144 degrees. These model results display a high estimated fit. Model output values are compared to system output values and can be viewed below in Figure 5.26.

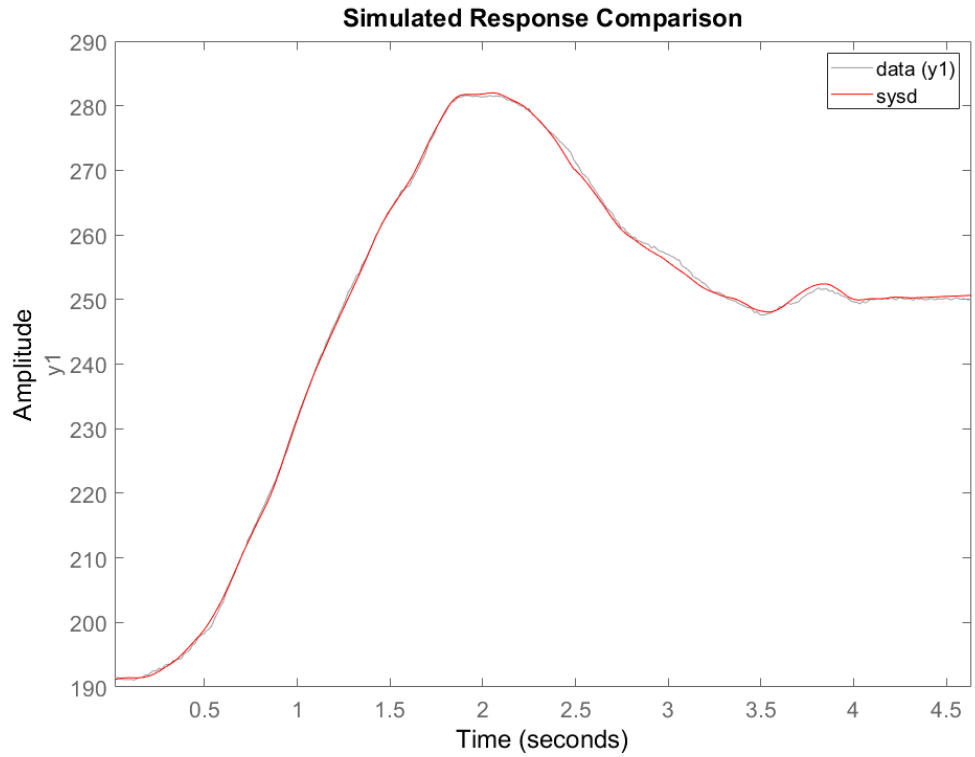


Figure 5.26: No-load state-space model compared to no-load system.

Figure 5.26 shows that model output (sysd) closely follows the system output (data (y1)), with the most significant point of error being 0.8 degrees. There is high confidence that the step response and characteristics of this state-space model compare closely to the physical system.

Another modeling technique was used and compared to that of the state-space model. In a MATLAB script, a loop was run that performed second-order transfer function estimation calculations of the system via the *tfest* command as the Hankel singular values determined this to be of optimal order. Four algorithms, Instrumental Variable (*iv*), State Variable Filters (*svf*), Generalized Poisson Moment Functions (*gpmf*), and Subspace state-space estimation (*n4sid*), were used to obtain a continuous-time transfer function of the control system. All algorithms were run against the system *iddata* object through all second-order pole-zero possibilities. The algorithm that produced the lowest MSE value was chosen as the MSE value is

considered one of the best determining factors of error between data and estimated regression and has a direct relationship to model fit percentage. The estimated model fit percentage can be calculated from MSE by Equation 5.2.

$$f = 100 * (1 - MSE) \tag{5.2}$$

Where f = fit percentage as a percent.

The corresponding number of poles and zeros utilized in the best algorithm's calculation was then extracted and used to generate the system transfer function. Table 5.6 displays the best results for each algorithm, its corresponding number of poles and zeros, and the FPE of the model in degrees. The n4sid-c data in this table is from the *n4sid* algorithm that obtains a state-space model and converts it into a continuous-time model. The n4sid-d data is from the discrete-time state-space model described earlier in this section and is listed for comparison purposes.

Table 5.6: MATLAB "No-Load" transfer function estimation results.

Algorithm	Fit %	MSE	# of Poles	# of Zeros	FPE
iv	97.84	0.316	2	0	0.326
svf	86.47	12.347	2	2	12.92
gpmf	97.84	0.316	2	0	0.326
n4sid-c	98.02	0.264	2	1	0.274
n4sid-d	99.28	0.041	2	1	0.043

The n4sid-d data obtained from the *n4sid* discrete-time state-space algorithm performed significantly better than any of the continuous-time models with the lowest MSE and FPE and utilized 2 poles and 1 zero. The minimal final prediction error of 0.04328 degrees alongside the 99.28% estimated fit percentage provides excellent support that the *n4sid* algorithm produced a transfer function very close to the physical

system. The *n4sid* algorithm then generated a discrete-time transfer function with a sampling time of 15 ms as follows:

$$T(z) = \frac{-0.08212z^{-1} + 0.1098z^{-2}}{1 - 1.714z^{-1} + 0.7414z^{-2}}$$

A set of complex conjugate poles exist at 0.8568 ±0.085i and a zero at 1.3375. The resulting pole-zero plot can be viewed in Figure 5.27.

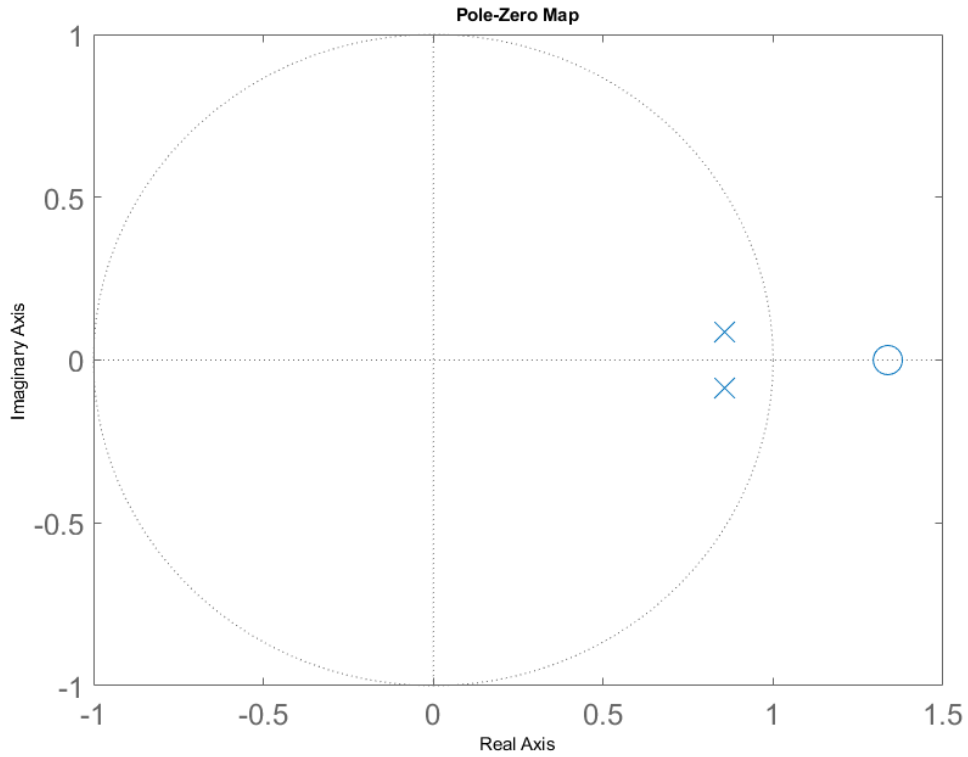


Figure 5.27: No-load pole-zero plot of discrete-time system, T(z).

Pole location inside the unit circle indicates a stable system; however, the existence of a zero outside of the unit circle indicates non-minimal phase. This phenomenon produced undershoot in the system step response and created phase instability at higher frequencies. This finding will be analyzed later in the continuous model bode plot. The step response of this model is shown in Figure 5.28.

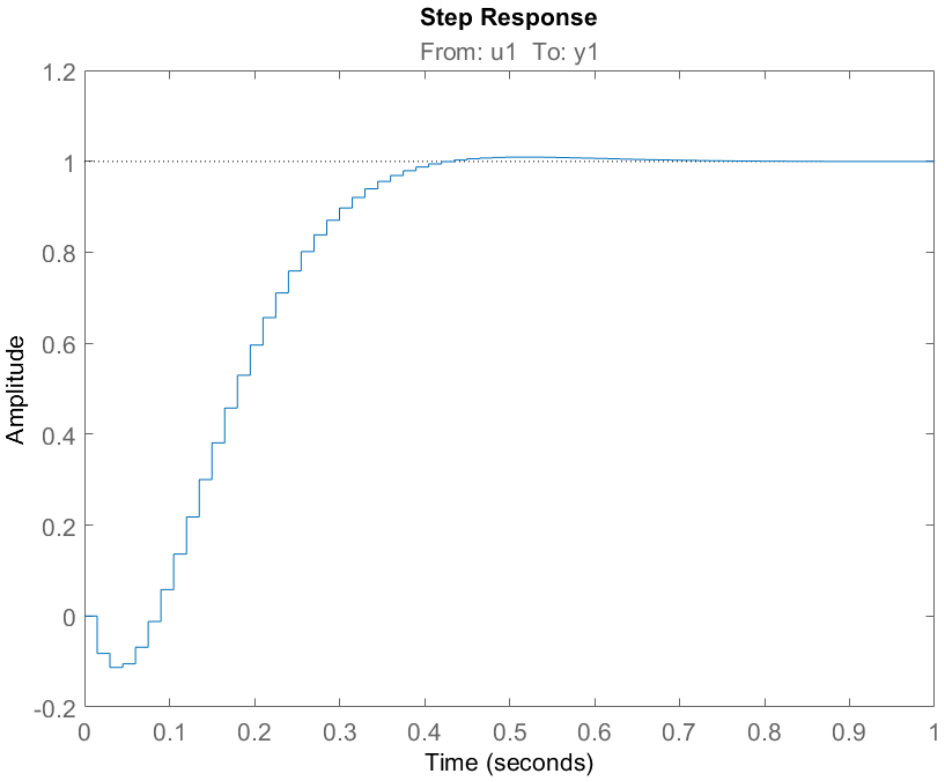


Figure 5.28: No-load state-space model step response.

The step response generated the characteristics listed in Table 5.7.

Table 5.7: Step response curve characteristics of T(z).

Characteristic	Value
Rise Time (s)	0.2100
Settling Time (s)	0.3750
Settling Min	0.9204
Settling Max	1.0092
Overshoot (%)	0.9606
Undershoot (%)	11.3061
Peak	1.0092
Peak Time (s)	0.5100

The response indicates excellent rise time, settling time, minimal oscillation, and overshoot. Overshoot peaks by a maximum of 0.96% which is acceptable for this application. The response also has a significant undershoot of 11.3%. For this control system, undershoot below 10% may be tolerated and could be beneficial. If the patient wearing the prosthesis is transitioning to a sitting position, the prosthesis should naturally move up towards the torso, moving the limb out of the way of the chair. A small amount of undershoot would move the limb down slightly, providing contact with the chair and stabilizing the patient as they begin to take a seat. This system has produced an unacceptable amount of undershoot, but loaded system undershoot is examined later in this chapter.

The physical system operates with continuous motion. To better visualize the response of the system with continuous motion, the discrete-time state-space model was transformed into a continuous-time model using MATLAB's *d2c* command. The following transfer function was generated with the same number of poles and zeros as the discrete-time model.

$$T(s) = \frac{-7.497s + 142.9}{s^2 + 19.95s + 142.9}$$

For this control system, T(s) is the closed-loop transfer function, Y(s)/U(s), with the system utilizing negative unity feedback from the motor position, Y(s), read by the encoder. The system output is subtracted from the commanded position of the control system and fed directly into the PD controller. This design can be more readily understood from the block diagram in Figure 5.29, which was generated with PD constant (k_p and k_d); these constants were determined by methods described in *4.3 Motor Control and Feedback* and unity negative feedback to form the MATLAB generated transfer function T(s). In reference to Figure 5.29, system stability and step response characteristics are discussed later in this section and focus on the output of the control system, T(s). The requirement that the plant subsystem transfer function,

G(s), be known and determined experimentally is irrelevant as conclusions are made on data generated from the collective sum of the plant subsystem and PD controller.

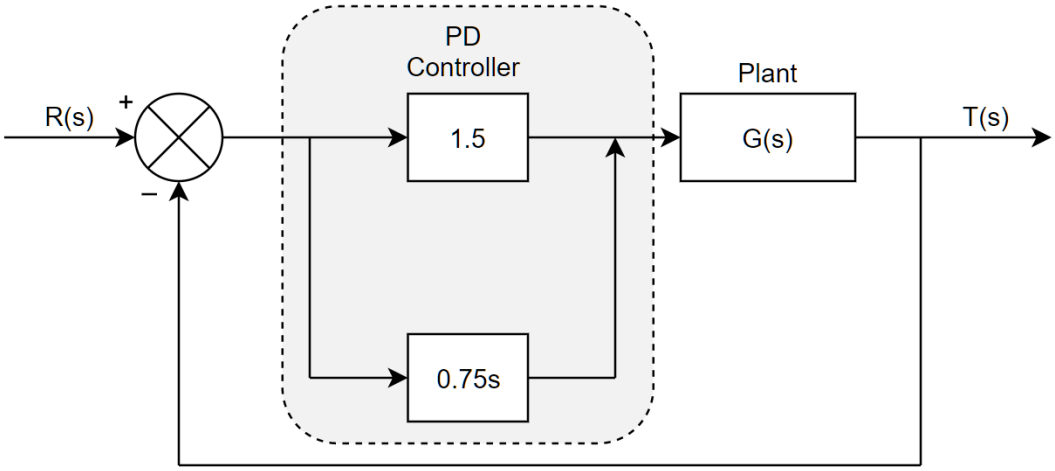


Figure 5.29: Initial block diagram of the control system

T(s) contains a set of complex conjugate poles at -9.9749 ± 6.5924i and a zero at 19.0611. A pole-zero plot of this function was generated and can be viewed in Figure 5.30.

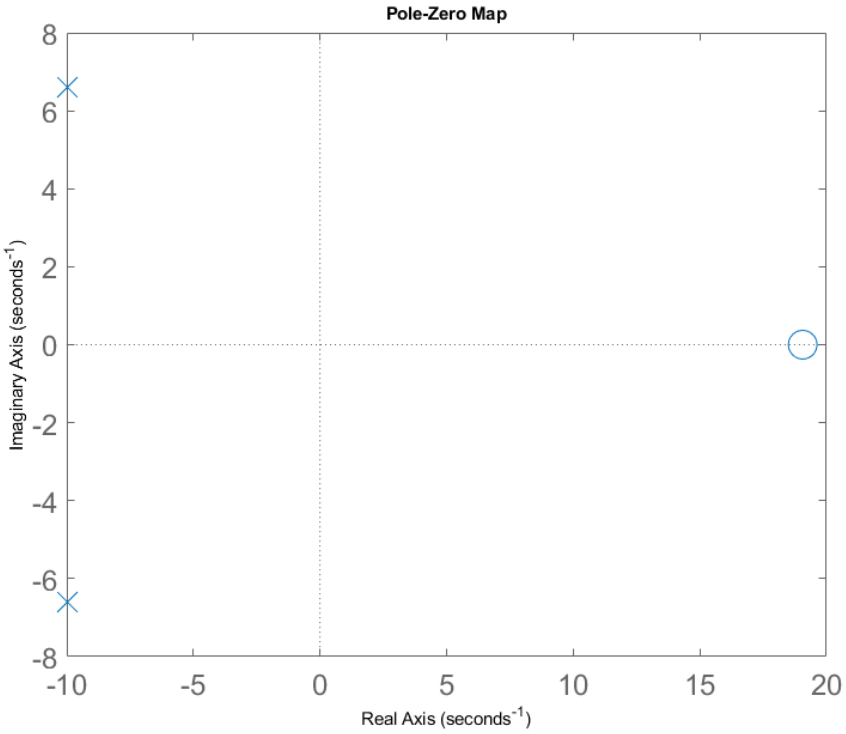


Figure 5.30: Pole-zero plot of continuous-time model, T(s).

The presence of all poles in the left-half plane indicates a stable control system. A zero is located in the right-half plane which causes phase instability at higher frequencies. A Bode plot, Figure 5.31, displays the frequency response of the system in both magnitude and phase.

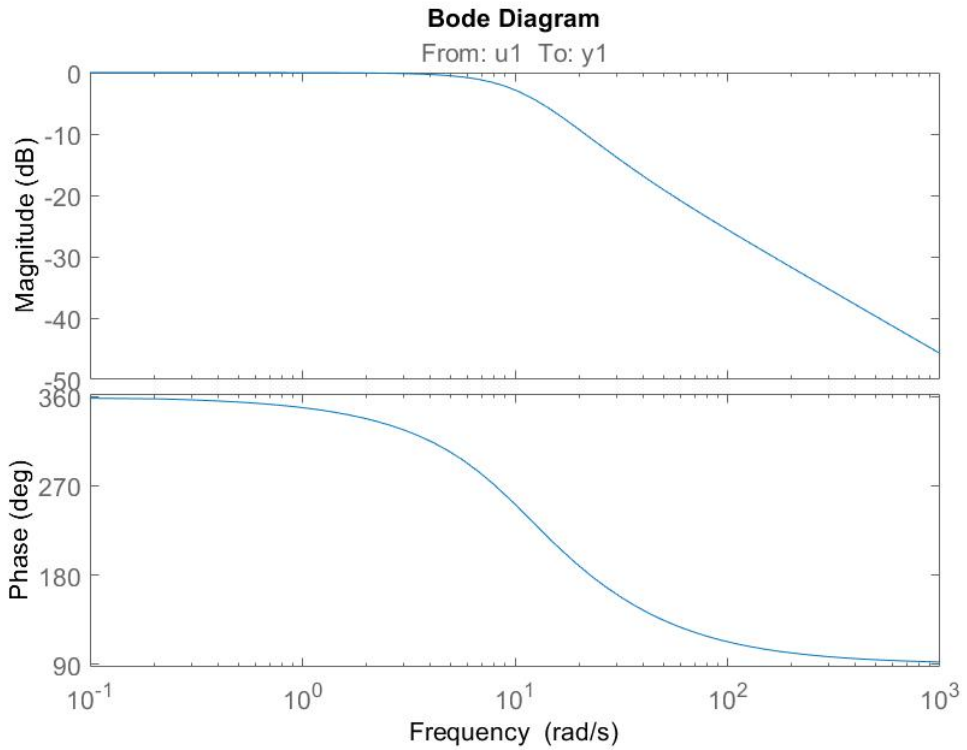


Figure 5.31: Bode plot of T(s).

The right half plane zero creates a -90 degree phase shift (just as a left half pole does) and creates undershoot in the step response, as seen in Figure 5.32. The system's frequency response in terms of magnitude mirrors that of a 2nd order lowpass filter with higher frequencies being eliminated. This response helps negate the effects of the right-half plane zero as system outputs are not of significant magnitude at higher frequencies where phase instability could be an issue. By examining the bode plot, a cutoff frequency,ω_c, of 10.29 rad/s was determined, which equates to 98.28 RPM. It is not expected that a human generate sitting and standing motions higher than 43 RPM [28]. The control system attenuates frequencies at and above 10.29 rad/s. Frequencies inside the range of expected human motion (below 43 RPM) have minimal magnitude loss and experience a change in phase of at most -51 degrees.

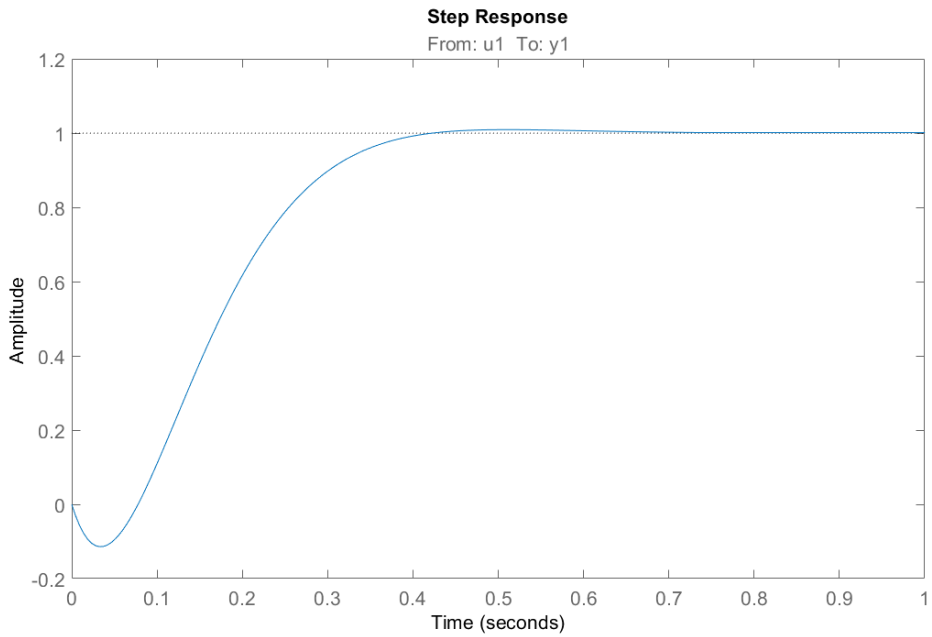


Figure 5.32: No-load step response of T(s).

The location of the zero creates undershoot in the step response of the system, visible in Figure 5.32. As mentioned previously, the undershoot viewed in this step response is significant and more examination was performed on the loaded system step response. Because this time-model was converted from a discrete-time model, the step response parameters nearly mirror that of the discrete model step response.

5.5 Loaded System Transfer Function Modeling

The same data collection and analysis methods were used for the loaded system modeling. Figure 5.33 displays the resulting curve when performing the standing to sitting motion with the system under load.

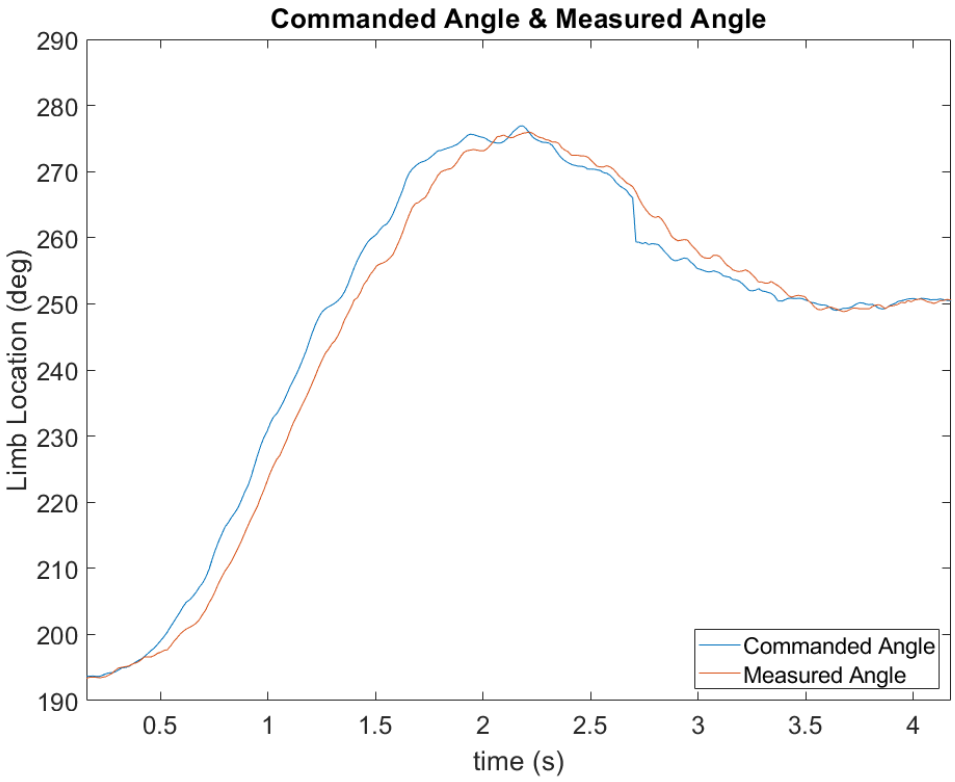


Figure 5.33: Full Load System Motion Data for System Modeling

This motion contains oscillations throughout the control system, in part due to the commanded angle containing some oscillatory motion after 2.5 seconds. A considerable reduction in commanded angle is seen just after 2.5 seconds as the torso axes aligned at 0 degrees and created a moment of discontinuity in the Euler equations. Because of the systems frequency response, discussed later in this section, fast motions such as this one are filtered out of the system. For the loaded system, the *n4sid* command was again utilized to determine the optimal system order and to generate a discrete-time state-space model. The resulting Hankel singular values are shown in Figure 5.34.

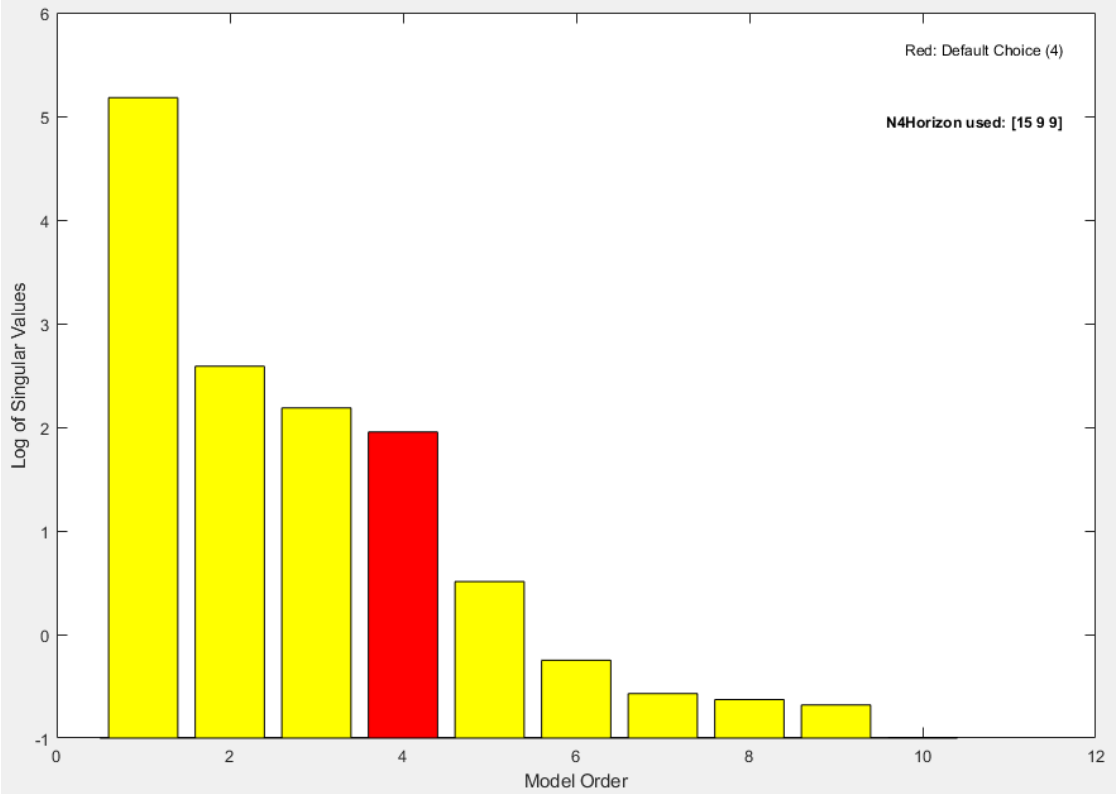


Figure 5.34: Loaded System *n4sid* results.

It was identified that a 4th-order system would be optimal based on the calculations performed during state-space model estimation. The estimated discrete-time state-space model with a sampling time of 15 ms is as follows:

$$x(t+Ts) = Ax(t) + Bu(t) + Ke(t)$$

$$y(t) = Cx(t) + Du(t) + e(t)$$

$$A = \begin{bmatrix} 0.936 & -0.009377 & 0.0536 & -0.02098 \\ 0.03404 & 0.8248 & 0.462 & 0.2957 \\ -0.08554 & -0.5112 & 0.7243 & 0.4073 \\ 0.1789 & -0.1712 & -0.2727 & 0.6872 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.00119 \\ 0.003438 \\ 0.01842 \\ -0.01504 \end{bmatrix}$$

$$C = \begin{bmatrix} 72.55 & -0.09067 & 1.711 & 0.02028 \end{bmatrix}$$

$$D = 0$$

$$K = \begin{bmatrix} 0.01423 \\ 0.04103 \\ 0.04693 \\ -0.02213 \end{bmatrix}$$

This state-space model was estimated using the *n4sid* discrete model state-space algorithm and obtained a fit percentage of 99.37% with an FPE of 0.03231 degrees and MSE of 0.0288 degrees. Figure 5.35 shows the estimated output data compared to the actual system output data.

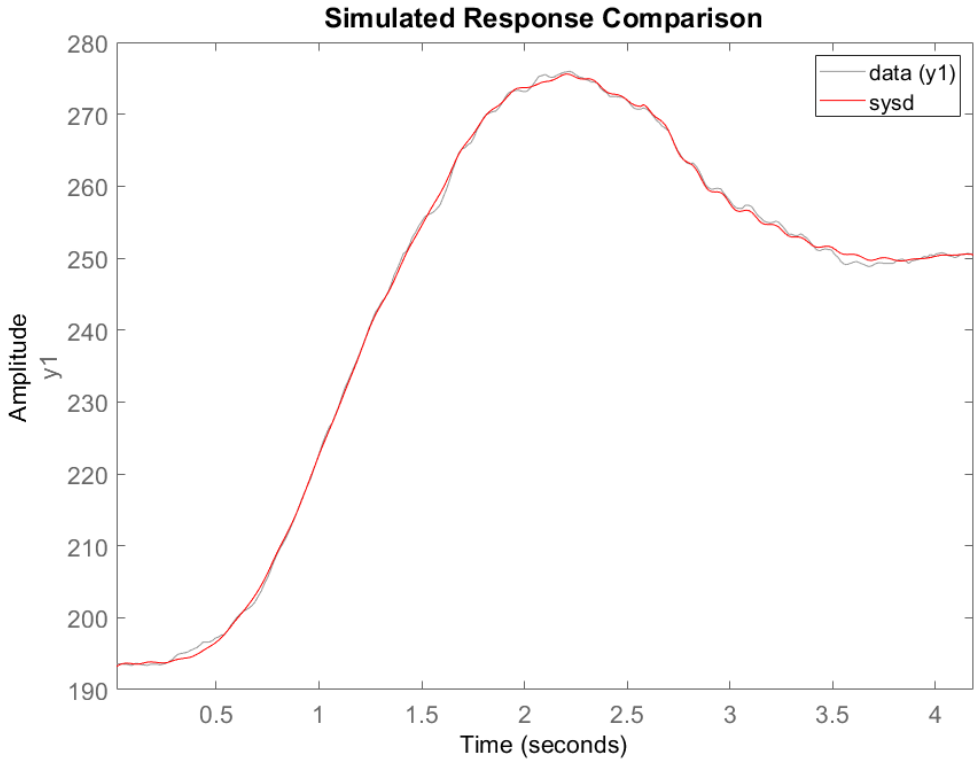


Figure 5.35: Loaded System state-space model comparison.

As with the no-load system tests, tfest was also used to find a continuous-time model to compare against the *n4sid* discrete-time model. Table 5.8 displays the results from the 4th-order system estimations.

Table 5.8: MATLAB "Load" transfer function estimation results.

Algorithm	Fit %	MSE	# of Poles	# of Zeros	FPE
iv	97.46	0.462	4	3	0.504
svf	86.52	13.00	4	2	14.07
gpmf	98.02	0.282	4	2	0.305
n4sid-c	98.16	0.243	4	3	0.264
n4sid-d	99.37	0.029	4	3	0.032

Again, the *n4sid* discrete-time state-space model performed significantly better

than any of the continuous-time model algorithms. Using this model, a discrete-time transfer function, T(z)' was generated as follows:

$$T(z)' = \frac{-0.05547z^{-1} + 0.1959z^{-2} - 0.2528z^{-3} + 0.1232z^{-4}}{1 - 3.172z^{-1} + 4.162z^{-2} - 2.654z^{-3} + 0.6751z^{-4}}$$

T(z) contains two sets of complex conjugate poles at 0.8481 ±0.0138i and 0.7380 ±0.6275i and three zeros at 1.4293 and 1.0508 ±0.6701i. The pole-zero plot is displayed below in Figure 5.36.

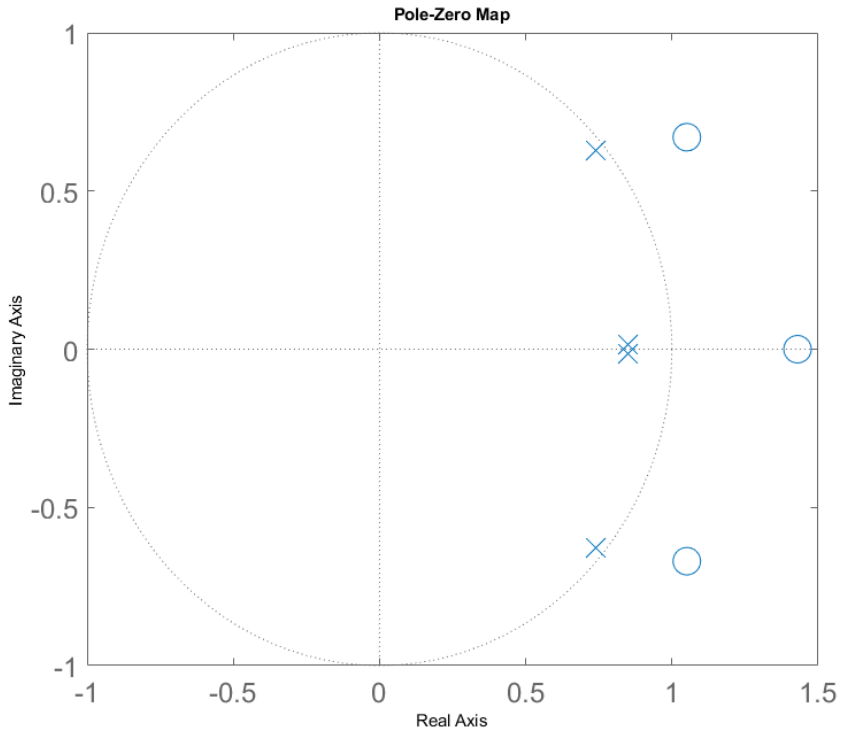


Figure 5.36: Loaded System pole-zero plot of discrete model, T(z).

As with the no-load analysis, poles inside the unit circle and zeros outside of the unit circle indicate a stable system with phase instability in higher frequencies and possible undershoot in the step response. The discrete-time step response is shown in Figure 5.37, with its characteristics displayed in Table 5.9.

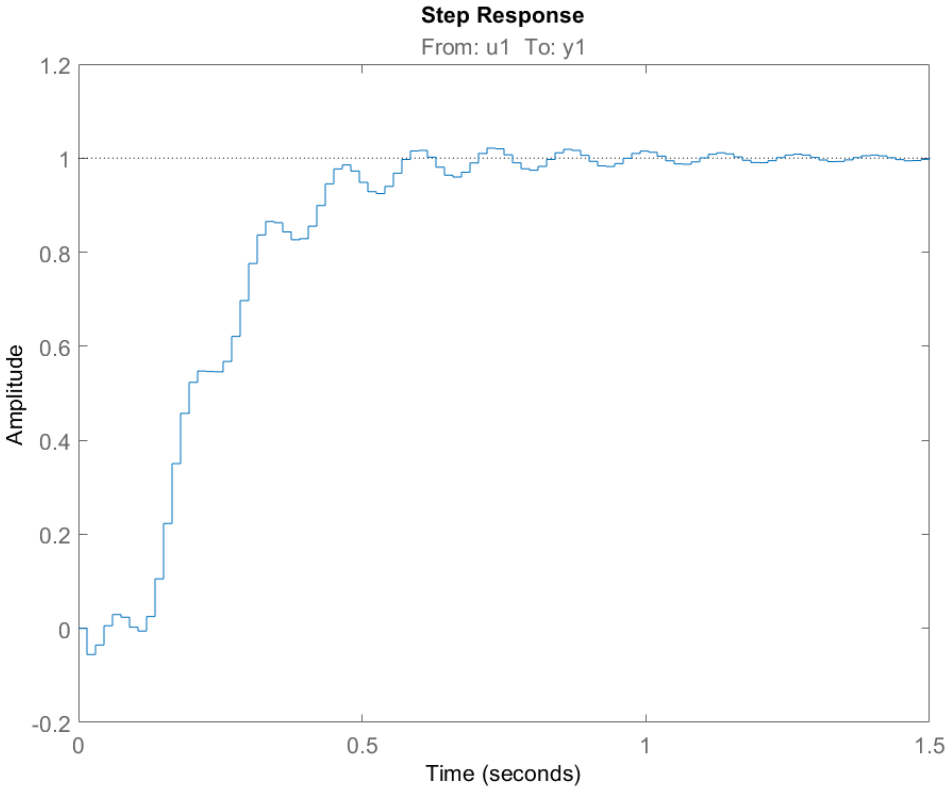


Figure 5.37: Loaded system step response of discrete model, T(z)'.

Table 5.9: Loaded step response characteristics of T(z)'.

Characteristic	Value
Rise Time (s)	0.3000
Settling Time (s)	0.8100
Settling Min	0.9249
Settling Max	1.0214
Overshoot (%)	2.1334
Undershoot (%)	5.5469
Peak	1.0214
Peak Time (s)	0.7200

These characteristics are similar in number to those observed in the no-load system

step response from Table 5.7, except there is a slight increase in overshoot and a substantial reduction in undershoot. The overshoot of this model, T(z)', is 2.13% with peak oscillations between 1.0214 and 0.9249. The curve has distinct visual differences compared to the no-load step response. There are steady oscillations that occur throughout the entire response. This fact was noted earlier when observing the system output in Figure 5.33.

For a cleaner representation of the system, a continuous-time model was generated using the "d2c" MATLAB command. The transfer function, T(s)', is displayed below:

$$T(s)' = \frac{-7.432s^3 + 376.9s^2 - 16520s + 268900}{s^4 + 26.19s^3 + 2426s^2 + 49050s + 268900}$$

The same PD parameters were used during load testing such that the block diagram created by the system remains the same, as viewed in Figure 5.29. T(s)' contains two sets of complex conjugate poles at -10.9741 ±1.087i and -2.1202 ±46.9749i, as well as three zeros at 13.9576 ±37.3095i and 22.8033. The pole-zero plot of T(s)' is shown below in Figure 5.38.

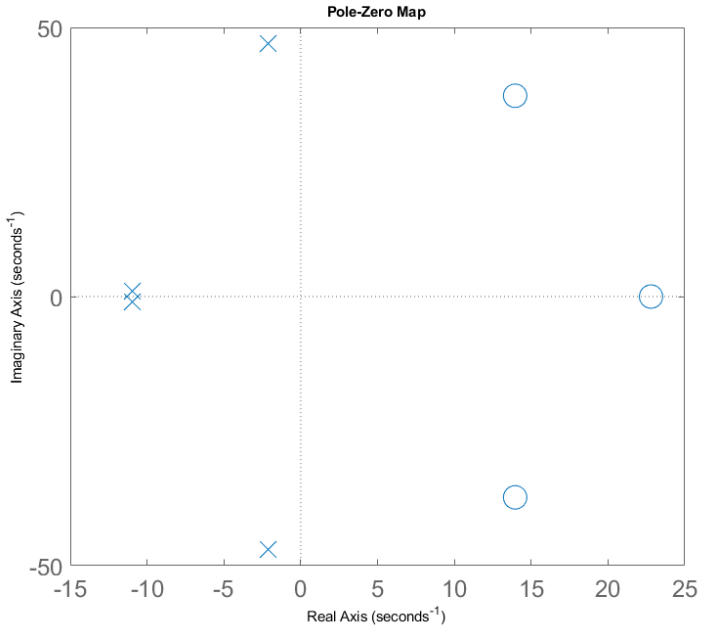


Figure 5.38: Pole-zero plot of T(s)'.

As with the discrete-time model, pole locations indicate a stable system, while zero location indicates a minimum phase system. The frequency response in magnitude and phase can be viewed in Figure 5.39.

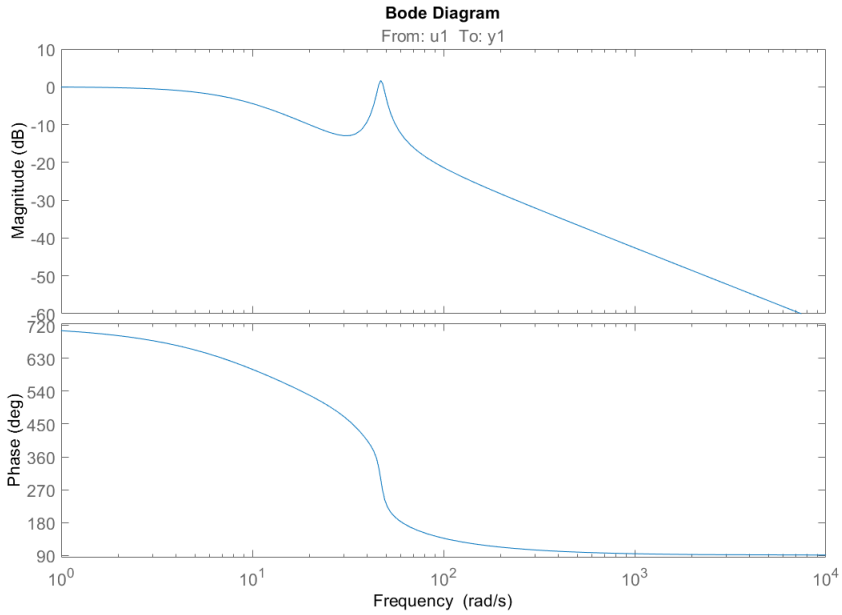


Figure 5.39: Bode plot of T(s)'.

The Bode plot displays some interesting characteristics. The right-half plane zeros create an unwanted peak of gain in higher frequencies between 44 and 50 rad/s. Though the system remains stable, frequencies could be introduced to the system and create unwanted side effects such as a slight increase in gain (1.68db) with a high negative phase shift (-376 degrees). The three zeros have created a very sharp phase shift just after the cutoff frequency of 7.72 rad/s. The location of the higher frequency peak occurs at a phase shift around -360 degrees, ultimately producing a phase shift of 0 degrees. This nullified phase shift is useful in the case frequencies between 44 and 50 rad/s are introduced into the system. The frequency response of this loaded system is far from ideal; however, it is stable and minimizes phase shift in frequencies that are passed by the system.

The continuous-time step response is now analyzed and viewed in Figure 5.40.

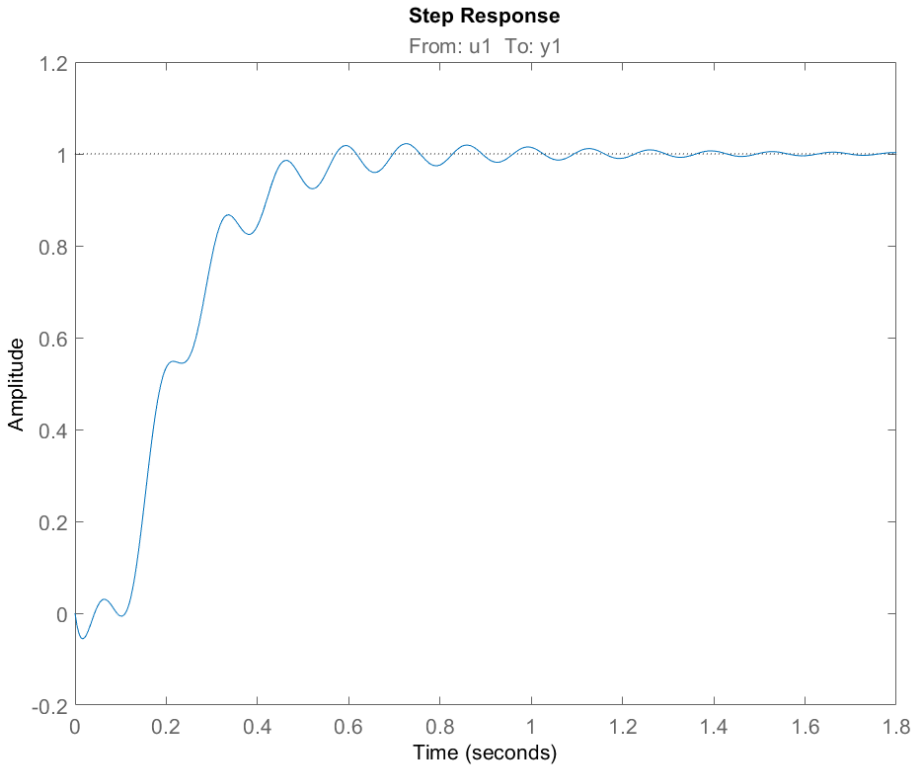


Figure 5.40: Step Response of T(s)'.

Just as with the no-load system, the step response characteristics are very similar to those shown in the discrete-time model as MATLAB has performed a direct transformation. This response more clearly displays the overall motion of the loaded system with visible oscillations existing throughout the response. These oscillations are most likely caused by the load's inertial on the motor. Because the motor is rotating at 60 RPM, faster than the human body is moving, the motor must slow itself between steps. As the motor advances a step, the inertia of the load continues to rotate the shaft, and the motor then compensates by stepping the motor back to the commanded position. This response occurs rapidly (many times per second) and is not be visible to the naked eye. The step response shows an undershoot of 5.6%, far better than the no-load 11.3%. 5.6% seems relatively reasonable, considering how quickly it is eliminated. Undershoot correction occurs within 42 ms with a step rise time of 300 ms (286 ms in the continuous-time model). If these oscillations produce unwanted motion, an increase in the derivative constant, k_d, could eliminate them; however, the rise time would be affected, and the system may operate far too slowly.

CHAPTER 6: CONCLUSIONS

6.1 Summary

As more patients survive transpelvic amputations, the need for autonomous hip-joint prosthetics continues to grow. The shift towards increasing the quality of life for transpelvic amputation survivors must be considered. This research designed and analyzed a proof-of-concept model for autonomously controlling a single-axis hip-joint prosthesis. A 3-D camera system was successfully used to validate digital data obtained from two body-mounted IMUs and a motor encoder. An embedded controller performed calculations that successfully determined thigh and torso orientation in real-time, while also sending precise motor positional commands that were used to direct single-axis motion of a simulated hip-joint prosthesis. All system data was tracking simultaneously in real-time data across three separate software platforms. Analysis of this prosthesis analogue showed stable, and precise human-mapped single-axis motion that could be feasible and effective in further research toward the development of an autonomous prosthetic device for patients that have undergone transpelvic amputations. Review of no-load and loaded full system data analysis concluded the motor and control system, with an acceptable amount of delay, generated adequate single-axis motion determined by calculations performed on data obtained from both the thigh and torso IMUs. The IMUs were able to produce an angle consistent with the physical orientation of the body and report that information quickly to the control system. The motor was able to perform bi-directional single-axis motion at speed consistent with human motion and within the range of hip-joint flexion and extension. The study, as a whole, successfully presented a control system which measures data accurately from body-mounted IMUs, and controls a closed-feedback, single-axis, rotational device, representative of a plausible electromechanical thigh-joint prosthesis.

6.2 Significant Findings

The Optitrack 3-D camera system was successful in validating digital data from the system encoder and IMUs. After performing multiple in-motion and stationary tests on all measurement devices, the camera system was able to successfully validate the accuracy of the encoder and both IMUs. Analysis of the system's no-load step response indicates minimal overshoot and an unacceptable amount of undershoot that, if reduced, could be beneficial to the amputee. Though logical assumptions have been made into the possible benefits of undershoot, a complete conclusion on the matter cannot be made until a simulated physical model is tested. The no-load system model, generated from the *n4sid* algorithm in MATLAB, showed system stability in magnitude and phase at lower frequencies. At higher frequencies, there could be phase instability; however, the magnitude of these higher frequencies are significantly attenuated to the point where phase instability is not of concern.

Analysis of the system under a load representative of a possible thigh prosthesis resulted in control characteristics better, in part, than when under no-load. The system response was slightly slower; however, undershoot was decreased significantly with the side effect of slightly increased overshoot. Pole locations indicate a stable system, but zero locations create an unusual phenomenon at certain frequencies where there is an increase in gain. Fortunately, at these high frequencies, the overall angle phase shift equates to a net phase shift of effectively zero. Under consideration of all no-load and load data, it is the opinion of the author that this system is plausible in its performance and future implementation into an electromechanical prosthesis, suitable for transpelvic amputees, with the addition of necessary future work as described below.

6.3 Future Work

To improve the control system and overall operation of the thigh-prosthesis analogue, recommendations for future research is described. Some topics might be considered as near-term future work, whereas others could be considered long-term future work. Near-term objectives should include research into prosthesis materials such that the physical properties of the prosthesis can be determined, thus eliminating many of the assumptions of this thesis. Prosthesis physical properties include optimizing and sizing of all system components as the mass and location of these components would alter the moment of inertia of the prosthesis. Due to the lack of available thigh prosthetics, identification and testing of more advanced motors to include larger servo motors and disc style dc motors could offer better system characteristics such as higher torque, lower current draw, and improved component compactedness if a prosthesis with a higher moment of inertia were to be necessary. Component size and mass are both crucial when optimizing the physical system which includes the motor, battery, embedded controller, and power supply. Their position and weight distributions are even more critical in optimizing the performance of the motor. The artificial limb should be small and lightweight to allow an amputee flexibility and agility. Unlike this proof-of-concept model, an actual prosthesis would need to be powered remotely within the limb itself and cannot be plugged into an external power source. Ensuring components are of the smallest possible size allows room for a larger battery pack and thus a longer prosthesis operational cycle period. There is some future work that can be accomplished with the current project assumptions but would benefit more with known physical limb properties. These topics include controls analysis of the system under many different types of inputs. Some unanswered questions are:

- How does the system respond with very slow input motion?
- How does the system respond if the input motion is greatly increased?
- Will producing an unexpected motion such as canceling standing to sitting

motion mid-test, cause system instability?

- What is the system response when injected with an impulse or a multi-step scenario?

Identifying the system's stability margin and robustness is an important design aspect that should be investigated. It is known that the system contains right-half plane zeros and thus could be unstable in higher frequencies. At what point will the system begin to fail? Although these higher frequencies cannot be generated by human motion, the effects of introducing higher frequency noise should still be investigated. It can be inferred that the system's stability margin will decrease as frequency increases due to the system's known "unstable" zeros. A possible solution to improving system robustness and stability could be in optimizing system code to reduce time delay [39].

Some long-term features that could be made to the current design include improvements to safety, ease of use, and quality of life for the user. The prosthesis could contain operational modes to help a user in fine-tuning its motion and position. An adjustment feature could be included that allows the user to jog the limb in either direction manually. This feature would be especially useful when the user is sitting, and the limb is not perfectly lined up with the other limb, causing a slight tilt in the body. The user would be able to jog the limb in either direction to a comfortable position. A separate mode could also be implemented that allows the user to quickly turn off position commands from the IMUs, putting the artificial limb in a holding torque mode, fundamentally locking the limb in place. A final additional feature that could be implemented is a manual angle offset, which would be useful when the limb is positioning too high or too low consistently through all motions.

REFERENCES

[1] N. C. Institute, "Anatomical terminology." Available at https://training.seer.cancer.gov/anatomy/body/terminology.html (10/09/2019).

[2] Prosthetic & Orthotic Care, "Hemipelvectomy and hip disarticulation." Available at http://www.pandocare.com/hemipelvectomy-hip-disarticulation/ (10/09/2019).

[3] M. Aragon, G. Orozco, and A. Altamirano, "Bionic hip prosthesis based on polycentric mechanisms," *2013 Pan American Health Care Exchanges (PAHCE)*, 2019. Last Retrieved on 10/13/2019 from IEEE Xplore database.

[4] S. Pandit *et al.*, "An affordable insole-sensor-based trans-femoral prosthesis for normal gait," *Sensors*, vol. 18, no. 3, 2018. Last Retrieved on 10/13/2019 from MDPI database.

[5] D. Smith, "Higher challenges: The hip disarticulation and transpelvic amputation levels," *InMotion*, vol. 15, no. 1, 2005. Last Retrieved on 09/28/2019 from http://www.prs-research.org/Texts/InMotion/15-1-document.pdf.

[6] C. Wedemeyer and M. Kauther, "Hemipelvectomy- only a savage therapy?," *Orthopedic Reviews*, vol. 3, no. 1, 2011. Last Retrieved on 10/11/2019 from PubMed Central database.

[7] J.P. Apffelstaedt, *et al.*, "Complications and outcome of external hemipelvectomy in the management of pelvic tumors.," *Annals of Surgical Oncology*, vol. 3, no. 304-9, 1996. Last Retrieved on 10/11/2019 from PubMed Central database.

[8] R. Capanna, *et al.*, "Complications of pelvis resections.," *Archives of Orthopaedic and Trauma Surgery*, vol. 106, no. 71-7, 1987. Last Retrieved on 10/11/2019 from PubMed Central database.

[9] A. Couto, *et al.*, "Survival rate and perioperative data of patients who have undergone hemipelvectomy: a retrospective case series.," *World Journal of Surgical Oncology*, no. 255, 2016. Last Retrieved on 10/11/2019 from BioMed Central database.

[10] S. Feng, E. Whitman, X. Xinjilefu, and C. Atkeson, "Optimization based full body control for the atlas robot." Available at http://www.cs.cmu.edu/~sfeng/sf_hum14.pdf (05/28/2020).

[11] T. Chin, R. Kuroda, T. Akisue, T. Iguchi, and M. Kurosaka, "Energy consumption during prosthetic walking and physical fitness in older hip disarticulation amputees," *Journal of Rehabilitation Research and Development*, vol. 49, no. 8, 2012. Last Retrieved on 04/01/2020 from PubMed Central database.

[12] M. T. Karimi, M. Kamali, H. Omar, and J. Mostmand, "Evaluation of gait performance of a hemipelvectomy amputation walking with a canadian prosthesis," *Case Reports in Orthopedics*, no. 962980, 2014. Last Retrieved on 03/29/2020 from PubMed Central database.

[13] M. T. Redfield, J. C. Cagle, B. J. Hafner, and J. E. Sanders, "Classifying prosthetic use via accelerometry in persons with transtibial amputations," *Journal of Rehabilitation Research and Development*, vol. 50, no. 9, 2013. Last Retrieved on 10/13/2019 from PubMed Central database.

[14] J. Lee and T. Jeon, "Imu-based but magnetometer-free joint angle estimation of constrained links," *IEEE Sensors*, 2018. Last Retrieved on 11/05/2019 from IEEE Xplore database.

[15] G. Thomas and D. Simon, "Inertial thigh angle sensing for a semi-active knee prosthesis," *Imaging and Signal Processing In Healthcare and Technology, IASTED International Symposia*, 2012. Last Retrieved on 10/12/2019 from Semantic Scholar database.

[16] C. Duraffourg, X. Bonnet, B. Dauriac, and H. Pillet, "Real time estimation of the pose of a lower limb prosthesis from a single shank mounted imu," *Sensors*, vol. 19, no. 13, 2019. Last Retrieved on 10/09/2019 from MDPI database.

[17] H. Benzerrouk and A. Nebylov, "Robust imu/uwb integration for indoor pedestrian navigation," *25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, 2018. Last Retrieved on 11/05/2019 from IEEE Xplore database.

[18] G. Li and T. Kuiken, "Modeling of prosthetic limb rotation control by sensing rotation of residual arm bone," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 9, 2008. Last Retrieved on 11/05/2019 from IEEE Xplore database.

[19] L. Cempini, M. Hargrove and T. Lenzi, "Design, development, and bench-top testing of a powered polycentric ankle prosthesis," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019. Last Retrieved on 10/13/2019 from IEEE Xplore database.

[20] M. Ryder and F. Sup, "Leveraging gait dynamics to improve efficiency and performance of powered hip exoskeletons," *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, 2013. Last Retrieved on 11/22/2019 from IEEE Xplore database.

[21] D. Blana, T. Kyriacou, J. M. Lambrecht, and E. K. Chadwick, "Feasibility of using combined emg and kinematic signals for prosthesis control: A simulation study using a virtual reality environment," *Journal of electromyography and kinesiology : official journal of the International Society of Electrophysiological Kinesiology*, vol. 29, 2016. Last Retrieved on 10/09/2019 from PubMed Central database.

[22] D. Nuemann, "Kinesiology of the hip: A focus on muscular actions," *Journal of Orthopedics & Sports Therapy*, vol. 40, no. 2, 2010. Last Retrieved on 03/29/2020 from JOSPT database.

[23] F. Pelisse, F. Marie, A. Bouzidi, and D. Geiger, "Microcontroller use for the assistance of a prosthesis for above-knee amputees," *14th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1992. Last Retrieved on 10/13/2019 from IEEE Xplore database.

[24] F. Aragon *et al.*, "Comparison between classic control systems techniques against adaptive and nonlinear control techniques in a lower limb prostheses," *4th International Conference on Control and Robotics Engineering (ICCRE)*, 2019. Last Retrieved on 10/13/2019 from IEEE Xplore database.

[25] M. Grimmer *et al.*, "A powered prosthetic ankle joint for walking and running," *BioMedical Engineering OnLine*, vol. 15, no. 141, 2016. Last Retrieved on 11/05/2019 from BioMed Central database.

[26] M. Saracoglu, D. Sahin, T. Orhanli, and A. Yilmaz, "Application of wireless, embedded microcontroller circuit for a semi-active above-knee prosthesis with pneumatic cylinder," *23nd Signal Processing and Communications Applications Conference (SIU)*, 2015. Last Retrieved on 11/05/2019 from IEEE Xplore database.

[27] D. Childress, "Hip disarticulation and transpelvic amputation: Prosthetic management," *Atlas of Limb Prosthetics*, vol. 2, no. 6D, 1992. Last Retrieved on 11/05/2019 from O&P Library database.

[28] K. Pratt and S. Sigward, "Inertial sensor angular velocities reflect dynamic knee loading during single limb loading in individuals following anterior cruciate ligament reconstruction," *Sensors*, vol. 18, no. 10, 2019. Last Retrieved on 11/22/2019 from MDPI database.

[29] J. Dowling, A.and Favre and T. Andriacchi, "The relationship between segment angular velocity and knee abduction moment during a drop jump: Implications for acl injury risk prediction," *57th Annual Meeting of the Orthopedic Research Society, January 13-16*, 2011. Last Retrieved on 11/22/2019 from Semantic Scholar database.

[30] G. Bastas, J. J. Fleck, R. A. Peters, and K. E. Zelik, "Imu-based gait analysis in lower limb prosthesis users: Comparison of step demarcation algorithms," *Gait & posture*, vol. 64, 2018. Last Retrieved on 10/09/2019 from PubMed Central database.

[31] N. Abhayasinghe and I. Murray, "Human activity recognition using thigh angle derived from single thigh mounted imu data," *International Conference on Indoor Navigation*, 2014. Last Retrieved on 10/12/2019 from IEEE Xplore database.

[32] H. Maqbool, *et al.*, "A real-time gait event detection for lower limb prosthesis control and evaluation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 9, 2017. Last Retrieved on 10/12/2019 from IEEE Xplore database.

[33] T. Seel, J. Raisch, and T. Schauer, "Imu-based joint angle measurement for gait analysis," *Sensors*, vol. 14, no. 4, 2014. Last Retrieved on 10/13/2019 from MDPI database.

[34] STEPPERONLINE, "Pull Out Torque Curve of 24HS39-4204D." Available at https://www.omc-stepperonline.com/download/24HS39-4204D_Torque_Curve.pdf (05/09/2020).

[35] CUI Devices, "AMT20 Series Datasheet - Modular | Absolute | CUI Devices." Available at https://www.cuidevices.com/product/resource/amt20.pdf (05/09/2020).

[36] National Instruments, "User Guide and Specifications. NI myRIO-1900." Available at http://www.ni.com/pdf/manuals/376047c.pdf (05/09/2020).

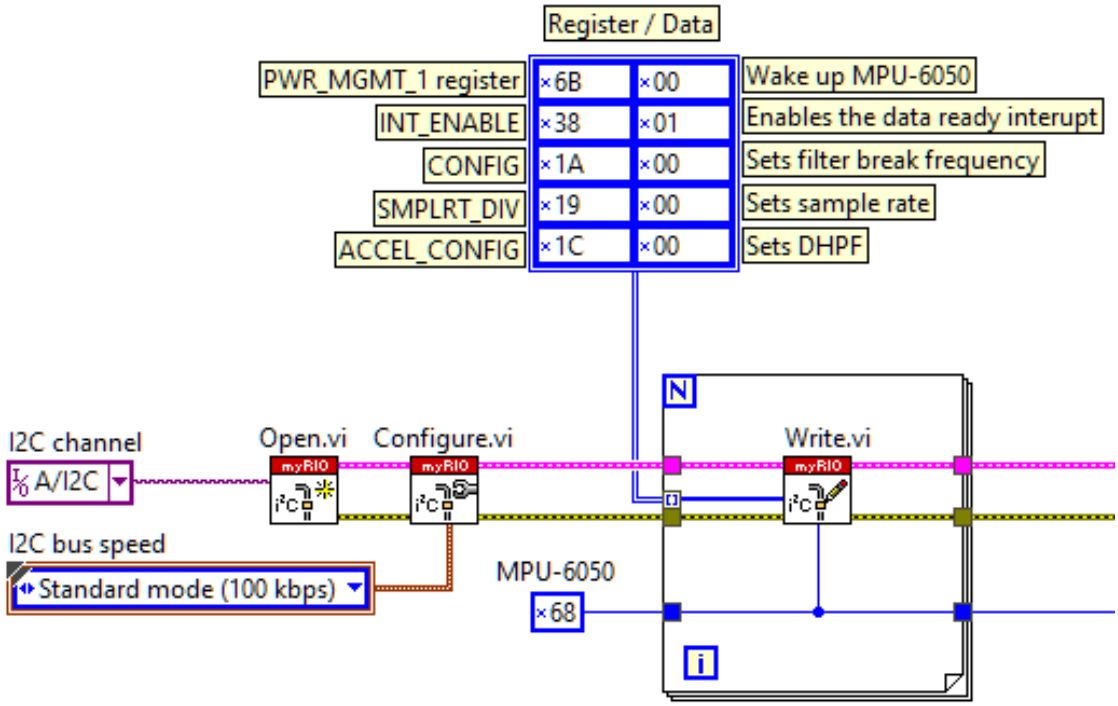
[37] InvenSense, "Mpu-6000 and mpu-6050 register map and description revision 4.2." Last Retrieved on 03/29/2020.

[38] Optitrack, "Quick Start Guide: Precision Capture." Available at https://v22.wiki.optitrack.com/index.php?title=Quick_Start_Guide:_Precision_Capture (05/18/2020).

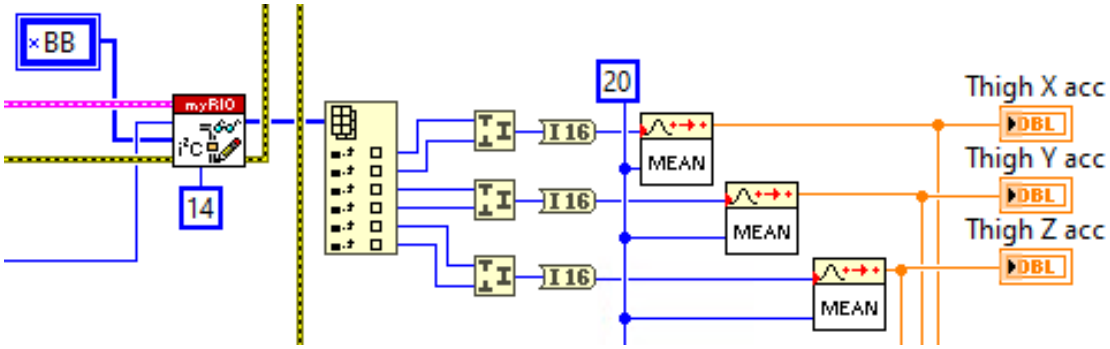
[39] S. Brunton, "Control bootcamp: Sensitivity and robustness." *YouTube*, uploaded March 8, 2017, https://www.youtube.com/watch?reload=9&v=7lzH-HnUFZg.

APPENDIX A: LABVIEW CODE

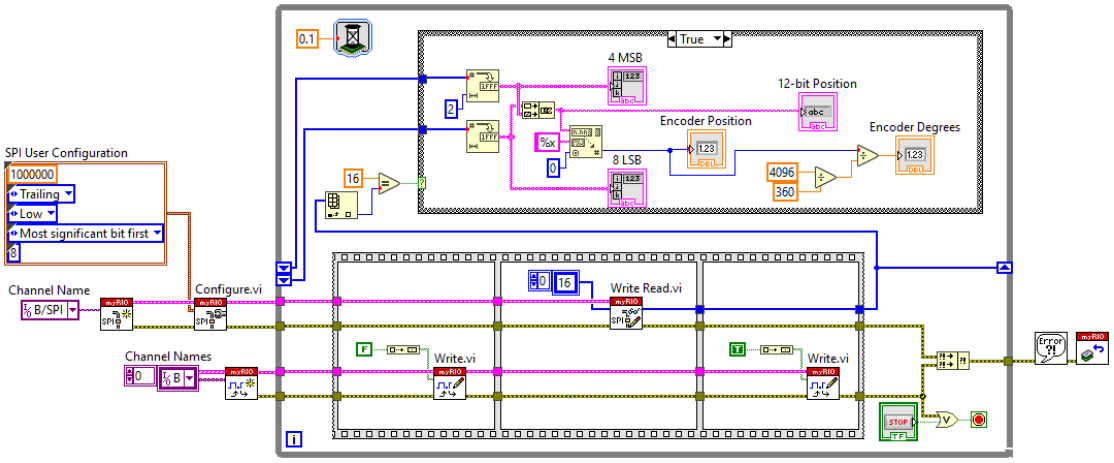
A.1 Configuring MPU-6050 over I2C



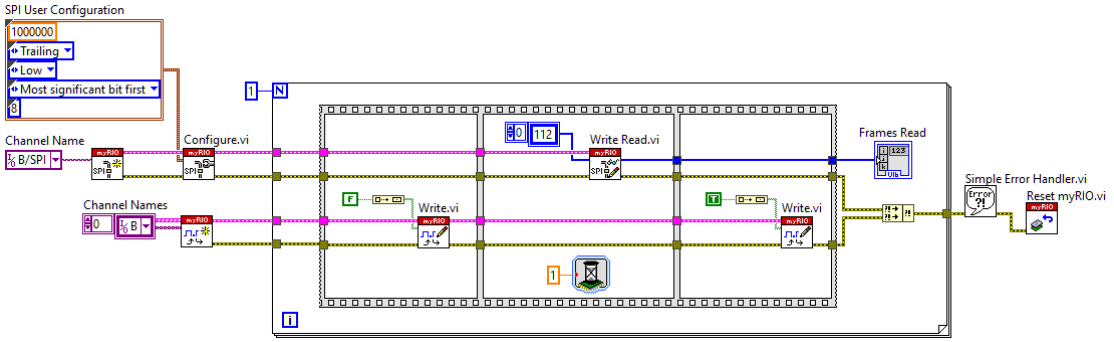
A.2 Reading MPU-6050 raw data over I2C



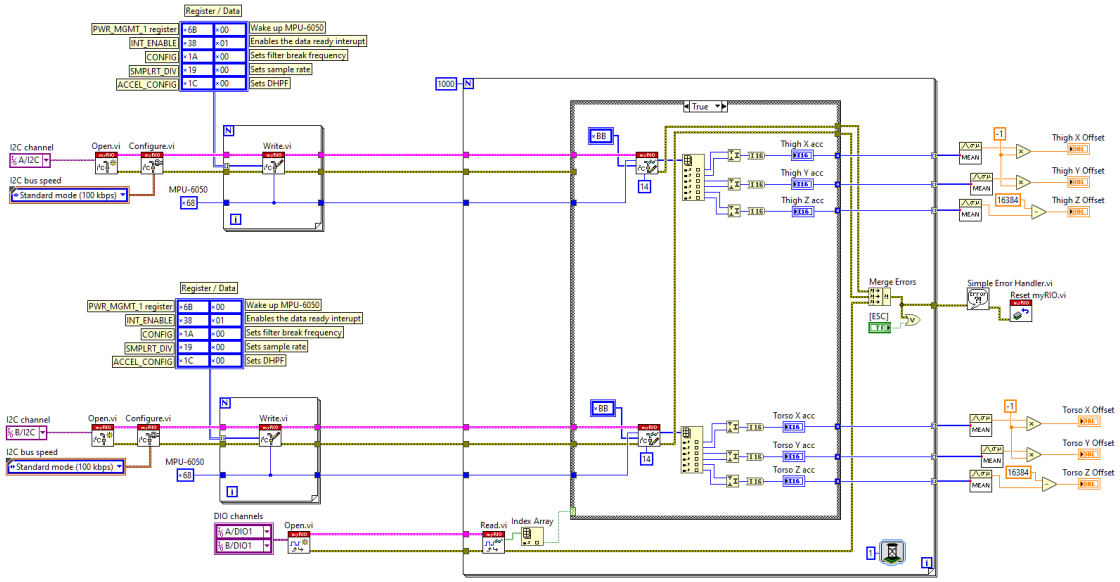
A.3 Obtaining encoder position over SPI



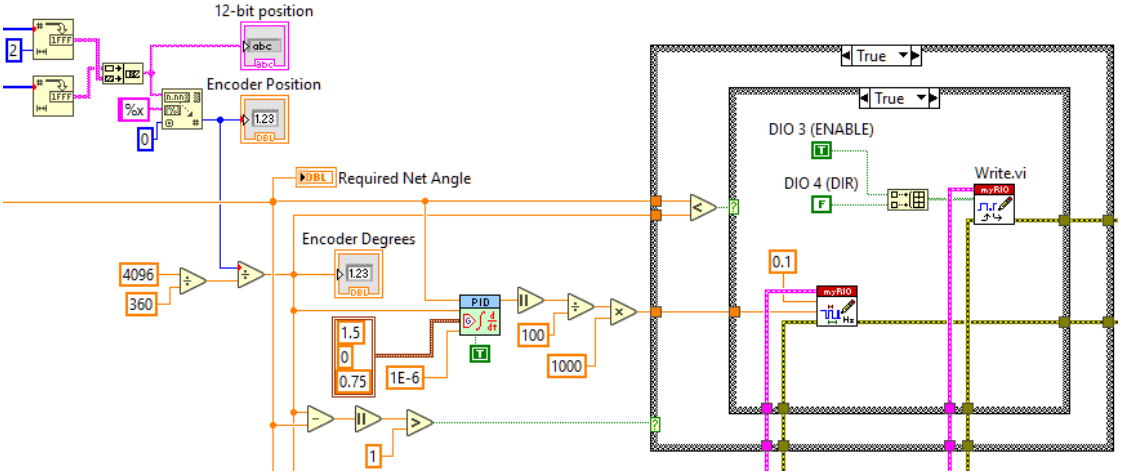
A.4 Zeroize encoder position over SPI



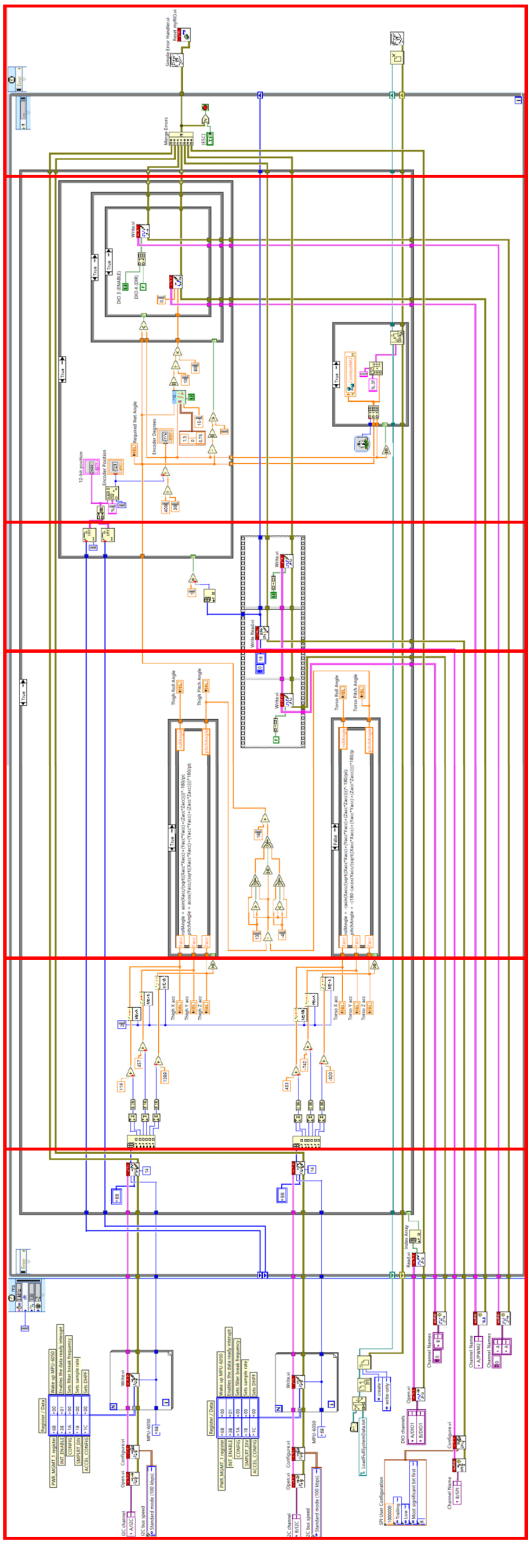
A.5 Calibrating 3-axis IMU accelerometer

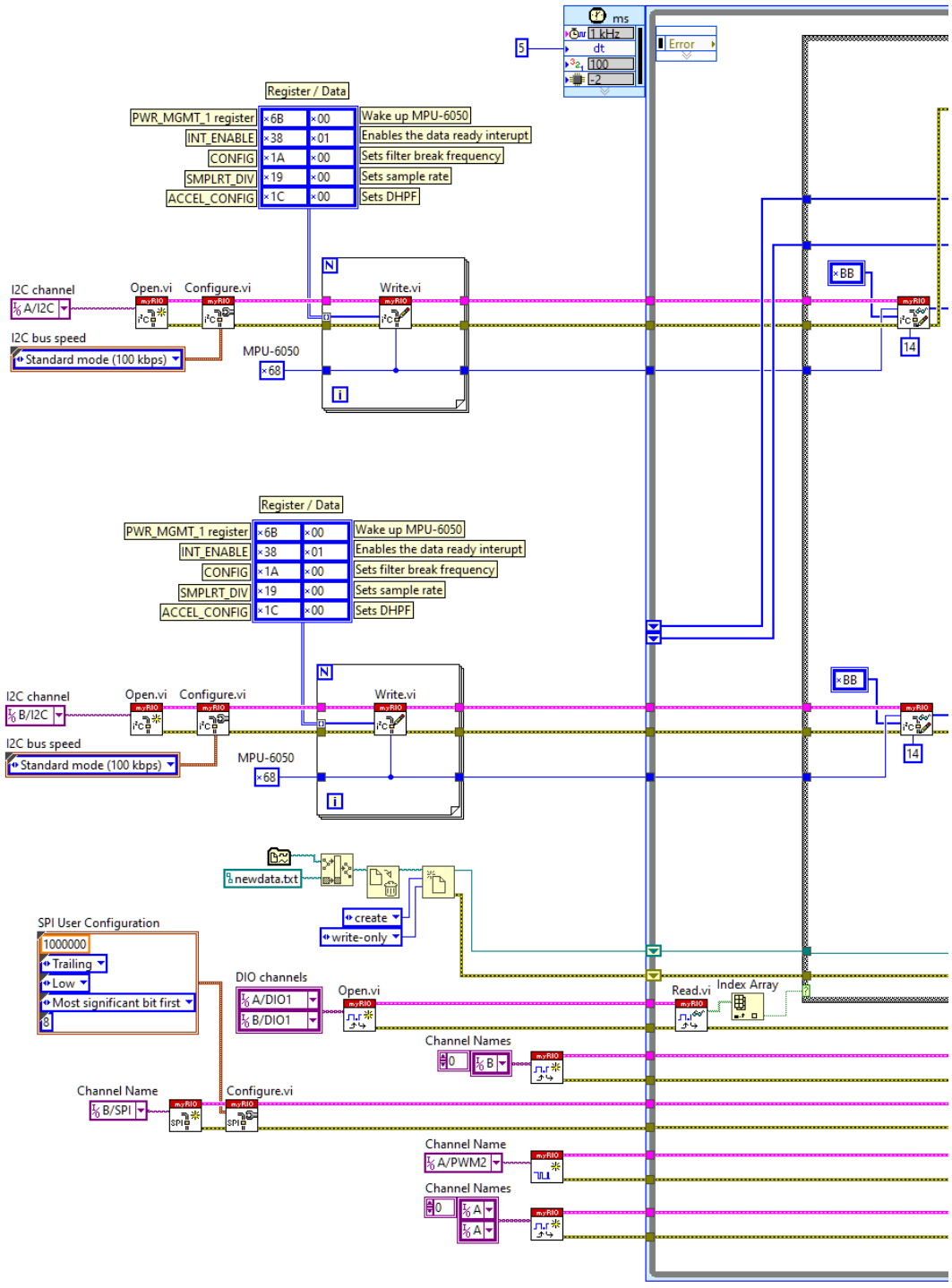


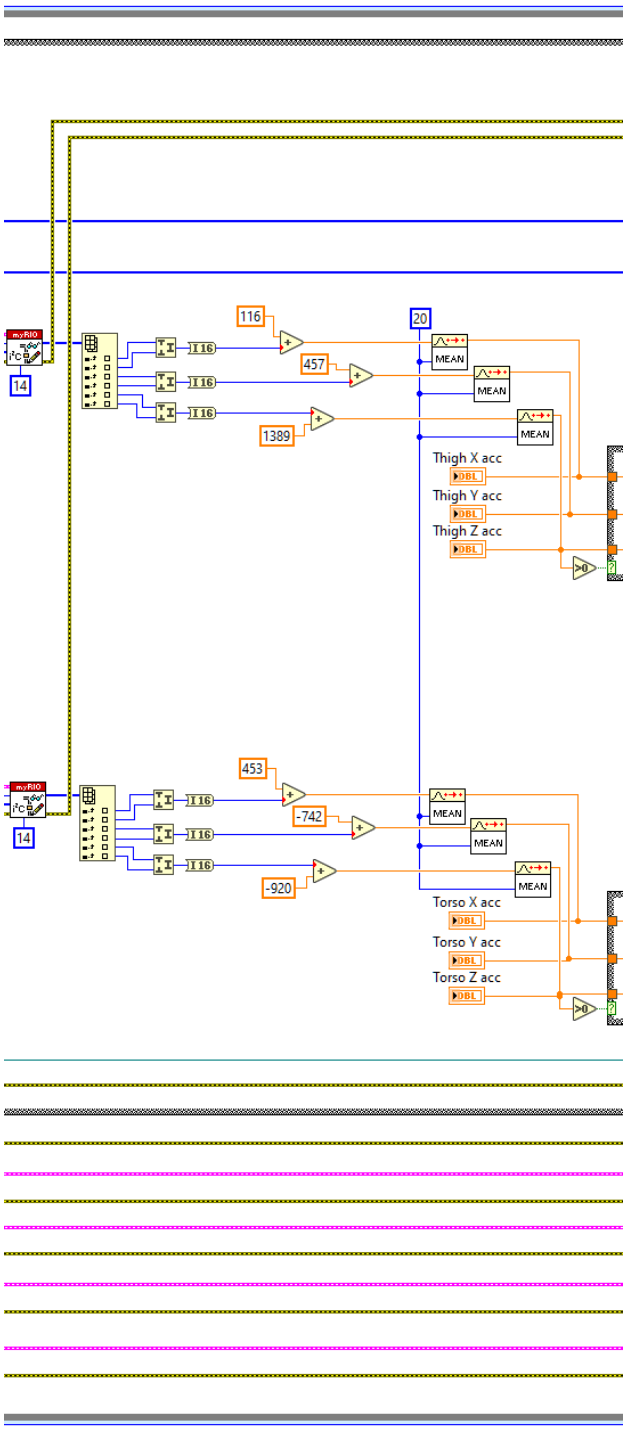
A.6 Stepper motor control with PD controller

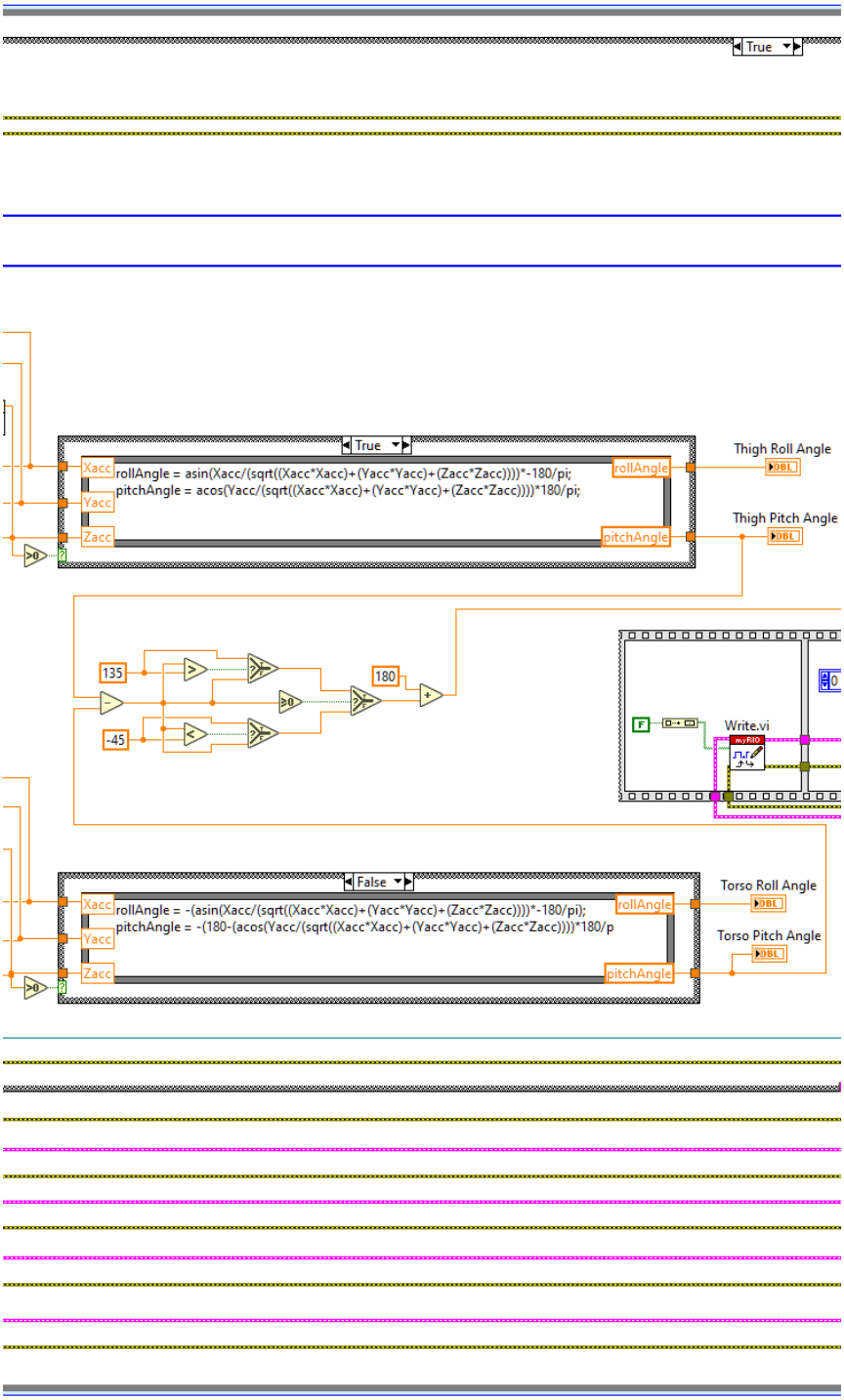


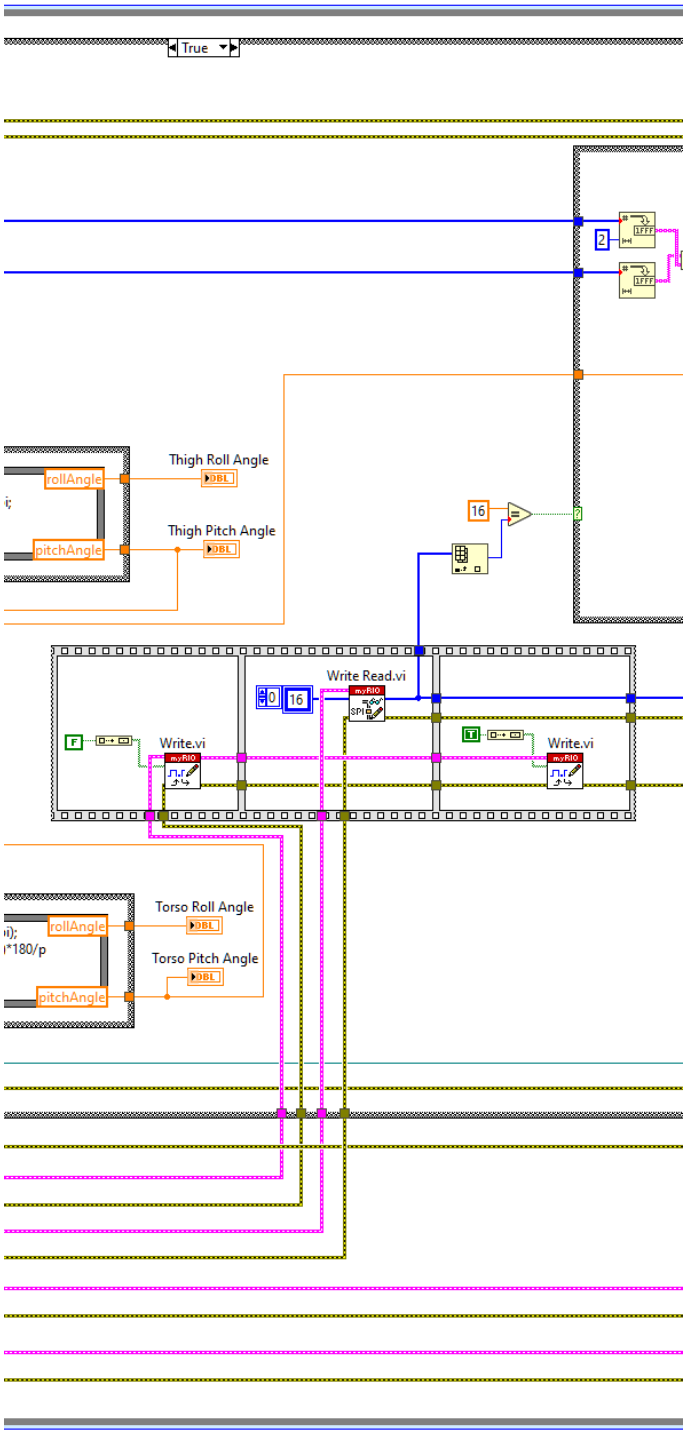
A.7 Entire LabVIEW Code

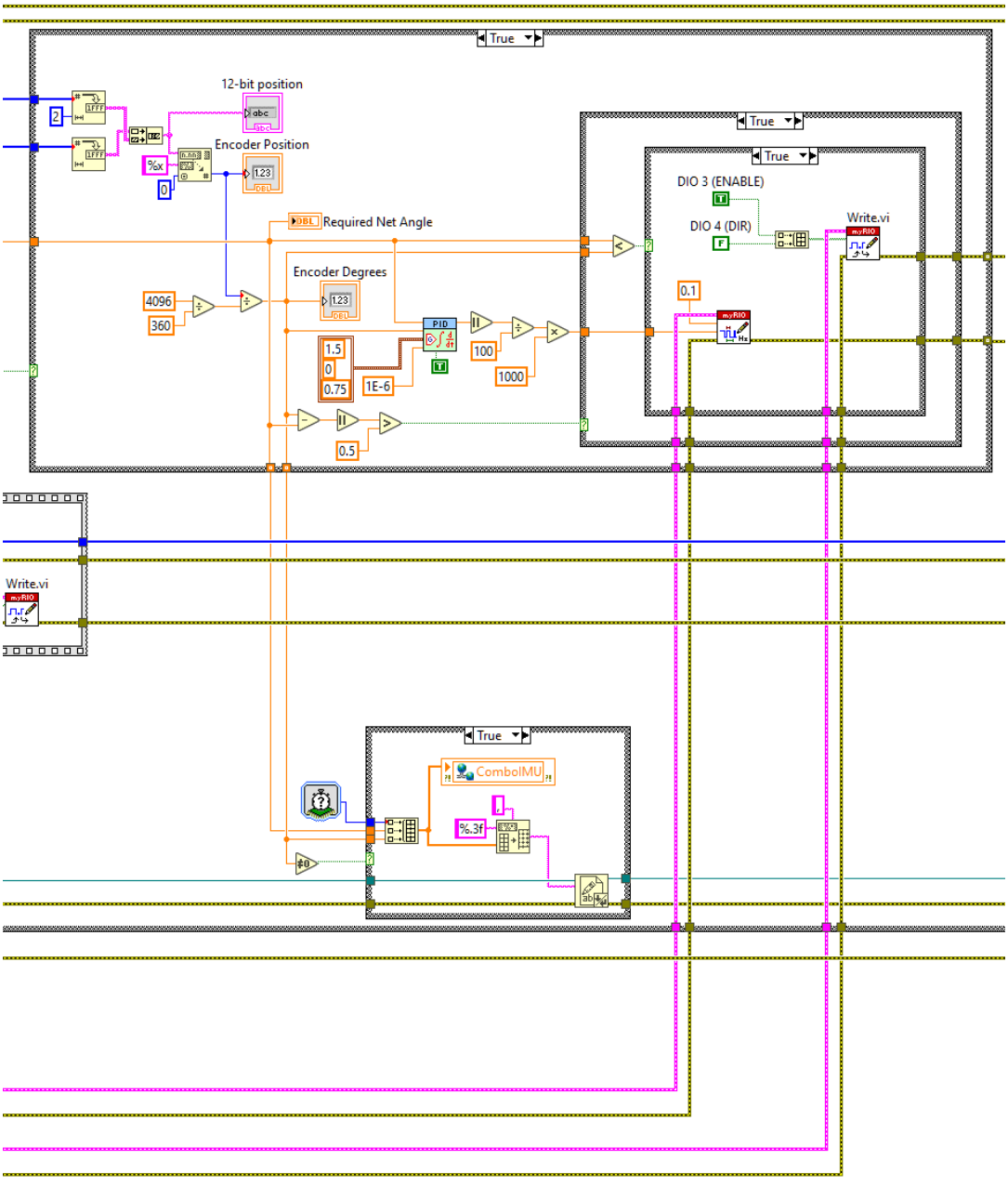


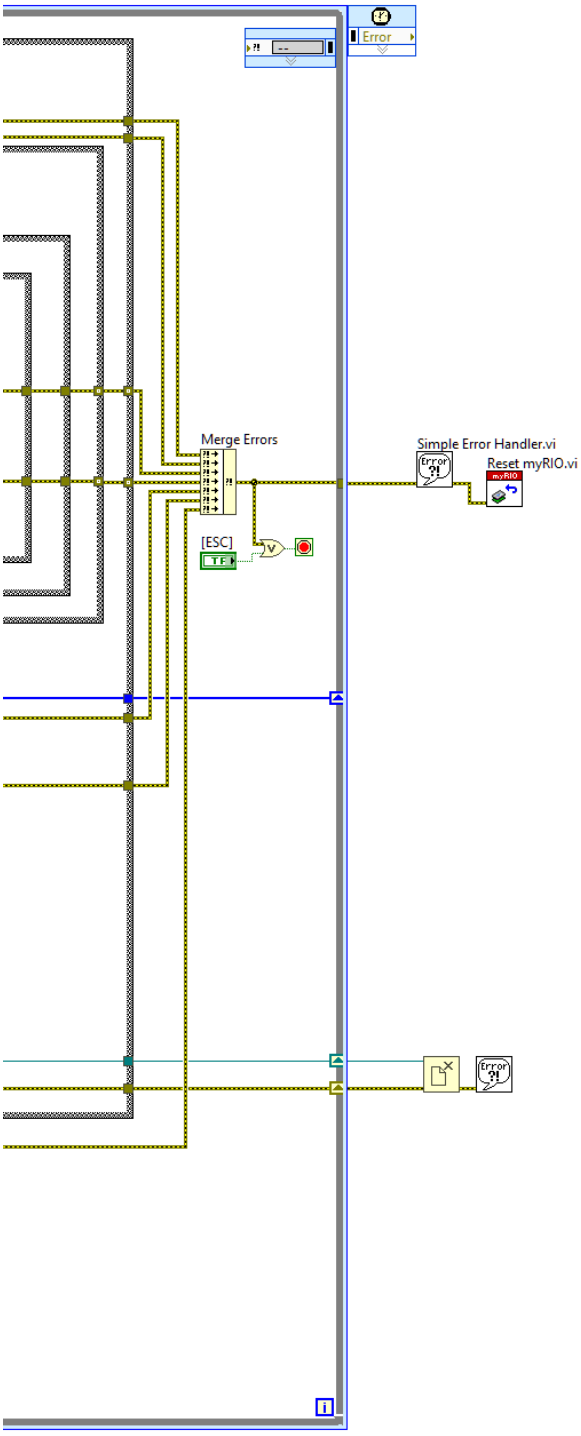

ms
1 kHz
dt
100
-2
5
Error
Register / Data
PWR_MGMT_1 register
INT_ENABLE
CONFIG
SMPLRT_DIV
ACCEL_CONFIG
×6B
×38
×1A
×19
×1C
×00
×01
×00
×00
×00
Wake up MPU-6050
Enables the data ready interupt
Sets filter break frequency
Sets sample rate
Sets DHPF
I2C channel
A/I2C
Open.vi
Configure.vi
Write.vi
I2C bus speed
Standard mode (100 kbps)
MPU-6050
×68
×BB
14
B/I2C
newdata.txt
create
write-only
SPI User Configuration
1000000
Trailing
Low
Most significant bit first
8
DIO channels
A/DIO1
B/DIO1
Read.vi
Index Array
Channel Names
Channel Name
B/SPI
A/PWM2
0


myRIO
I²C
14
I16
I16
I16
116
457
1389
20
MEAN
MEAN
MEAN
Thigh X acc
Thigh Y acc
Thigh Z acc
DBL
>0
453
-742
-920
Torso X acc
Torso Y acc
Torso Z acc


True
Xacc
Yacc
Zacc
rollAngle = asin(Xacc/(sqrt((Xacc*Xacc)+(Yacc*Yacc)+(Zacc*Zacc))))*-180/pi;
pitchAngle = acos(Yacc/(sqrt((Xacc*Xacc)+(Yacc*Yacc)+(Zacc*Zacc))))*180/pi;
rollAngle
pitchAngle
Thigh Roll Angle
Thigh Pitch Angle
135
-45
180
Write.vi
False
rollAngle = -(asin(Xacc/(sqrt((Xacc*Xacc)+(Yacc*Yacc)+(Zacc*Zacc))))*-180/pi);
pitchAngle = -(180-(acos(Yacc/(sqrt((Xacc*Xacc)+(Yacc*Yacc)+(Zacc*Zacc))))*180/p
Torso Roll Angle
Torso Pitch Angle

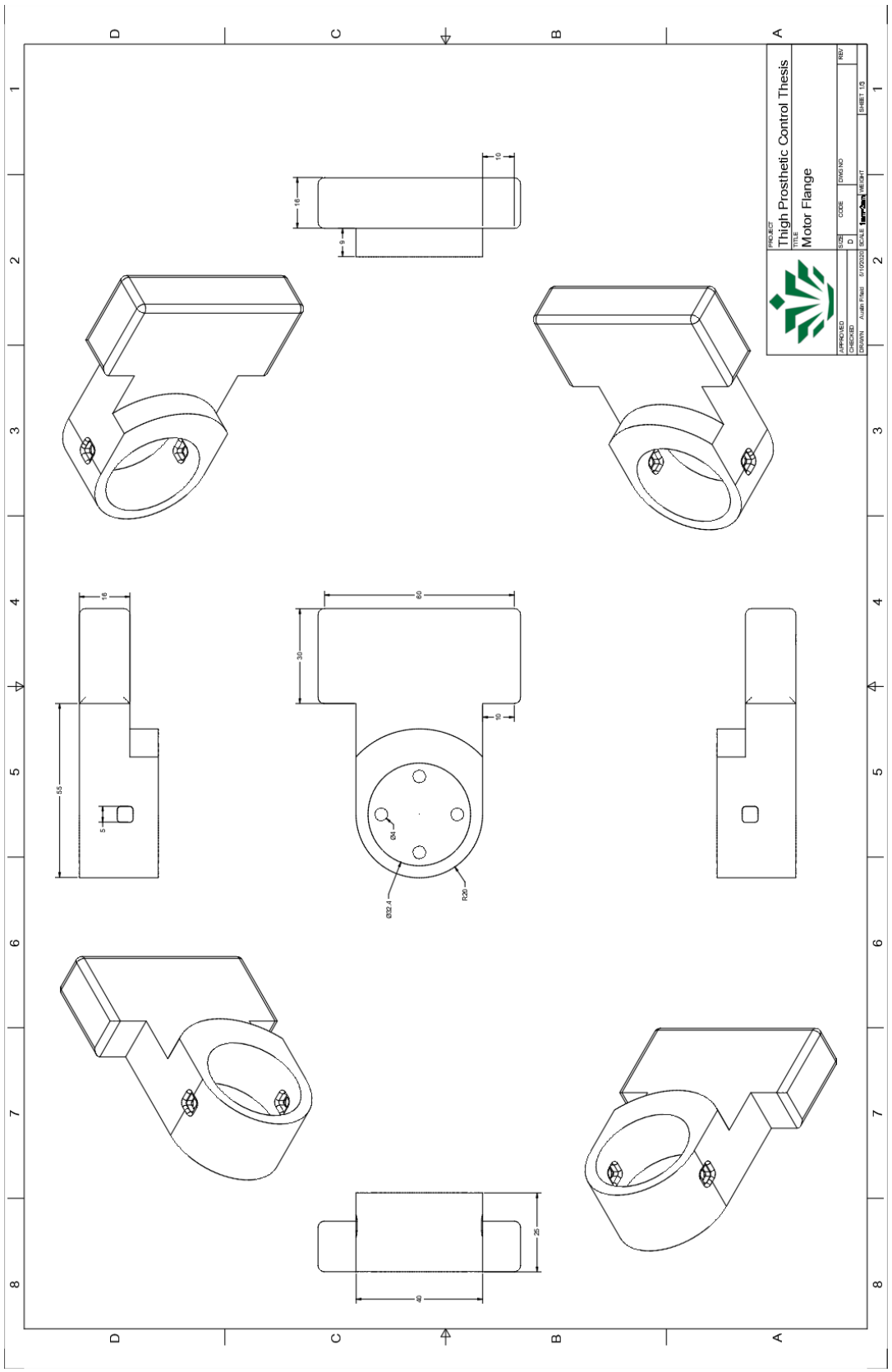

True
Thigh Roll Angle
Thigh Pitch Angle
Write Read.vi
Write.vi
Write.vi
Torso Roll Angle
Torso Pitch Angle


True
12-bit position
Encoder Position
Required Net Angle
Encoder Degrees
4096
360
PID
1.5
0
0.75
1E-6
100
1000
0.5
True
True
DIO 3 (ENABLE)
DIO 4 (DIR)
Write.vi
0.1
Write.vi
True
CombolMU
%.3f

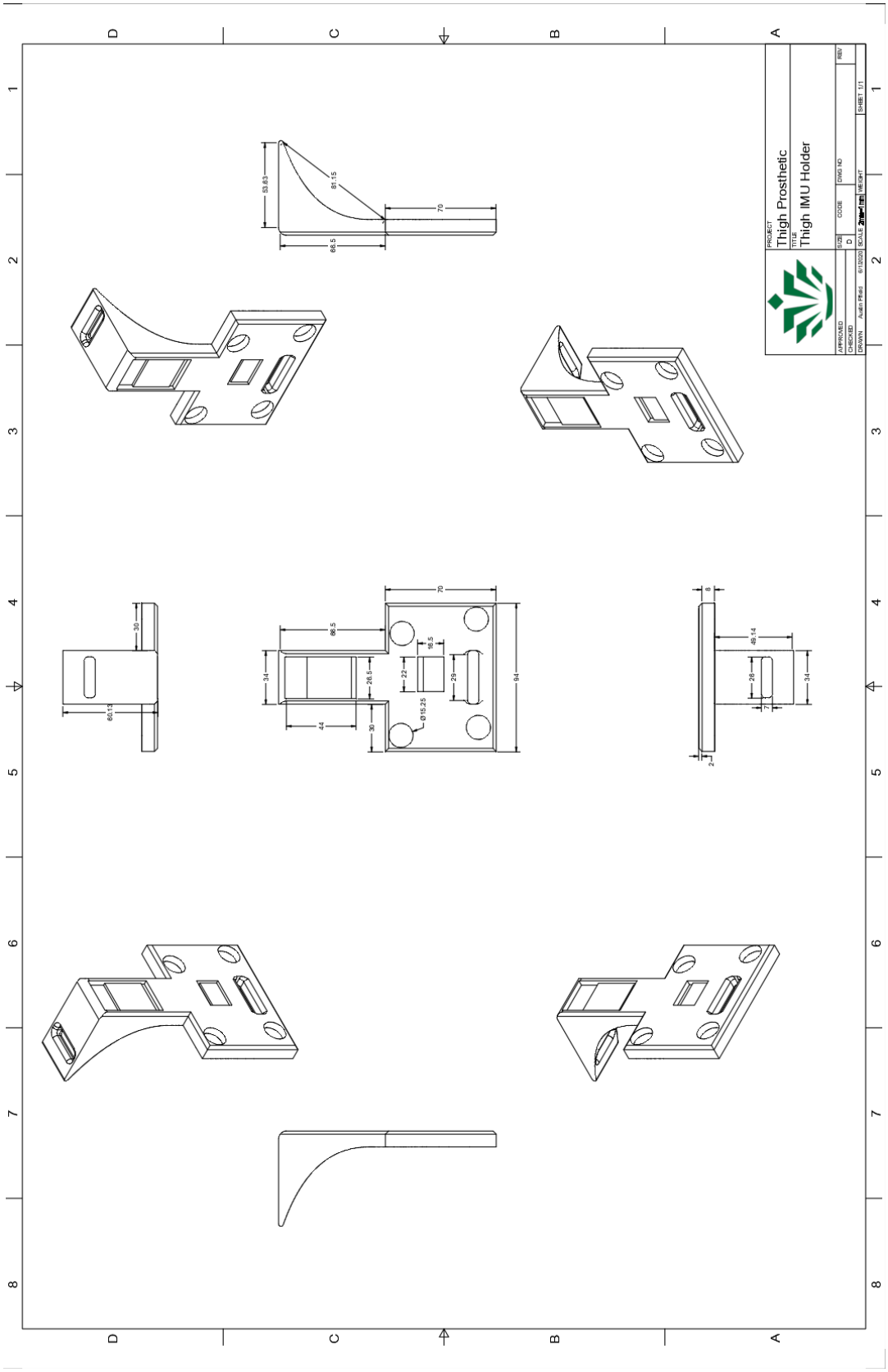

Error
Merge Errors
Simple Error Handler.vi
Reset myRIO.vi
[ESC]
TF

APPENDIX B: CAD DRAWINGS: 3-D MOTION COMPONENTS

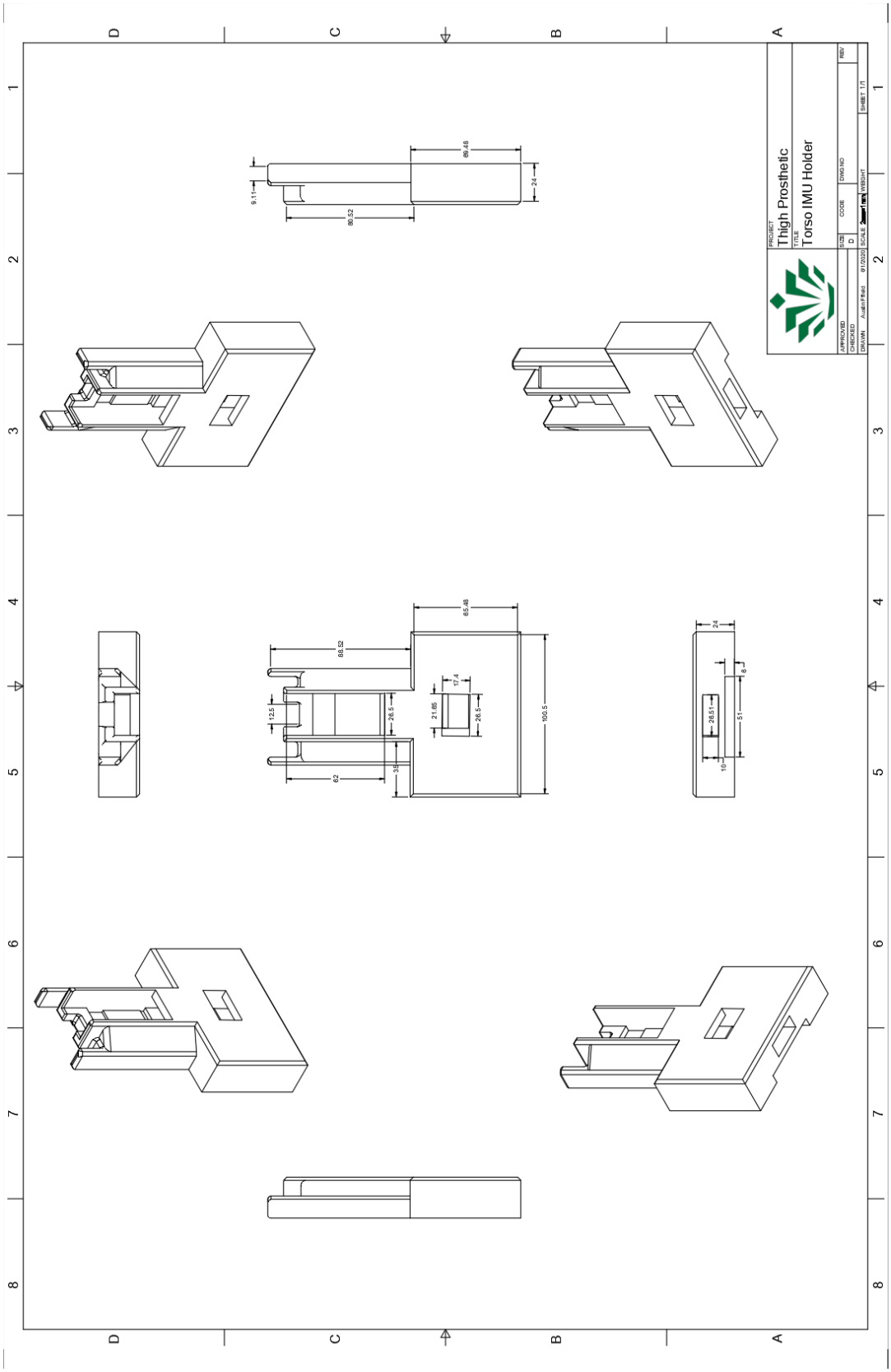
B.1 CAD Drawing of Motor Flange


PROJECT
Thigh Prosthetic Control Thesis
TITLE
Motor Flange
SIZE
D
CODE
DWG NO
REV
APPROVED
CHECKED
DRAWN
Austin Pillai
01/02/2020
SCALE
WEIGHT
SHEET 1/3
16
10
9
55
5
60
30
10
Ø4
Ø32.4
R20
40
25

B.2 CAD Drawing of Thigh IMU Holder


PROJECT
Thigh Prosthetic
TITLE
Thigh IMU Holder
SIZE
D
CODE
DWG NO
REV
SCALE
2in=1in
WEIGHT
SHEET 1/1
APPROVED
CHECKED
DRAWN
Austin Pinkos
8/13/2020
53.83
81.15
70
66.5
30
60.13
70
86.5
18.5
34
26.5
22
29
94
44
30
Ø15.25
6
48.14
28
34
7
2

B.3 CAD Drawing of Torso Holder IMU


PROJECT
Thigh Prosthetic
TITLE
Torso IMU Holder
SIZE
D
CODE
DWG NO
REV
SCALE
WEIGHT
SHEET 1/1
APPROVED
CHECKED
DRAWN
Austin Reid
6/1/2020
9.11
86.52
86.48
24
65.48
85.32
12.5
26.5
62
35
21.65
17.4
26.5
100.5
24
8
26.51
51
10

APPENDIX C: MATLAB SCRIPTS

C.1 Full System Data Analysis Script

```
clc;
clear;
close all;

file = 'C:\Users\Austin\Downloads\DataFiles\Matlab\
   FullSystemData\FullSystemData_1.csv';
G = load(file);

t = G(:,1)/20;
t1 = G(:,1);
tempMotorCamera = G(:,2);
tempEncoder = G(:,3);
motorOffset = 0;
motorCamera = G(:,2) - motorOffset;
encoder = circshift(tempEncoder,-1);
t(end) = [];
encoder(end) = [];
motorCamera(end) = [];

tempNetPitchCamera = G(:,6);
tempnetPitchIMU = G(:,7);
pitchOffset = -1.8;
netPitchCamera = tempNetPitchCamera - pitchOffset;
tempNetPitchIMU = movavg(tempnetPitchIMU,'simple',4);
```

```
netPitchIMU = circshift(tempNetPitchIMU,-1);


netPitchIMU(end) = [];
netPitchCamera(end) = [];


figure(1)
scatter(t, motorCamera,'r','.')
hold on
scatter(t,encoder,'b','.')
hold on
plot(t,motorCamera,'r',t,encoder,'b')
set(gca, 'FontSize', 18)
title('Camera vs Encoder Motor Position "Full System"')
xlabel('time(s)')
ylabel('Motor Position (Degrees)')
legend('Camera Measurement','Encoder Measurement','
   Location','southeast')


figure(2)
scatter(t,netPitchCamera,'r','.')
hold on
scatter(t,netPitchIMU,'b','.')
hold on
plot(t,netPitchCamera,'r',t,netPitchIMU,'b')
set(gca, 'FontSize', 18)
title('Camera vs IMU Net Angle "Full System"')
xlabel('time(s)')
```

```
ylabel('Net Angle (Thigh - Torso) (Degrees)')
legend('Camera Measurement','Net IMU Measurement','
   Location','southeast')


figure(3)
scatter(t,encoder,'r','.')
hold on
scatter(t,netPitchIMU,'b','.')
hold on
plot(t,encoder,'r',t,netPitchIMU,'b')
set(gca, 'FontSize', 18)
title('Encoder vs IMU Net Angle "Full System"')
xlabel('time(s)')
ylabel('Net Angle (Thigh - Torso) (Degrees)')
legend('Encoder Measurement','Net IMU Measurement','
   Location','southeast')
```

C.2 System Control and Response Script

```
clc;
clear;
close all;

load 'C:\Users\Austin\Downloads\DataFiles\Matlab\
   FullSystemData\FullSystemData.txt'
A = FullSystemData;

% Remove rows where extraneous data exists
A(abs(A(:,2)-A(:,3)) > 30, :) = [];

% Remove unwanted data before and after executed motion
A(1:225,:)=[];
A(310:end,:)=[];

% Convert time column in data to begin at t=0
Atemp = A(:,1)-A(1,1);
A(:,1) = Atemp;
A(:,1)=A(:,1);
[n,p] = size(A);

% Column 1 is the elapsed time in ms
time = A(:,1);

% Column 2 is commanded position from net IMU angle
```

```
commanded = A(:,2);

% Column 3 is actual measured limb angle from motor
   encoder
actual = A(:,3);

% Set time scale for plot
x_start = time(1,1);
x_end = time(n,1);

% Plot Commanded Angle vs Measured Angle
figure(1)
plot(time,commanded,time,actual)
axis([x_start x_end 180 290])
grid minor
datacursormode on
set(gca, 'FontSize', 18)
title('Commanded Angle & Measured Angle')
xlabel('time (ms)')
ylabel('Limb Location (deg)')
legend({'Commanded Angle','Measured Angle'},'Location','
   southeast')

% Create data object for transfer function estimation
data = iddata(actual,commanded,0.015);
fit = zeros(2,3);
```

```
% Iterate through all pole-zero possibilities up to 5
   poles & 5 zeros
opt = tfestOptions('InitializeMethod','n4sid','Display','
   on');
for i=2:1:2
    for j=0:1:2
        if i >= j
    sysc = tfest(data,i,j,opt);
    poles = i;
    zeros = j;
    fit(i,(j+1)) = sysc.Report.Fit.FitPercent;
    fprintf('Calculating fit estimation for %d Poles and %
       d Zeros\n', poles,zeros);
        end
    end
end

% Find best model fit percentage and extract number of
   poles and zeros
[maxValue, linearIndexesOfMaxes] = max(fit(:));
[maxValuePole, maxValueZero] = find(fit == maxValue);

% Perform Transfer Function estimation algorithm (tfest)
sysc = tfest(data,maxValuePole,maxValueZero-1,opt);

% Display results
fprintf('\nBest result is %d Poles and %d Zeros\n',
```

```
   maxValuePole, maxValueZero-1);
fprintf('Fit Estimation = %0.3f%%\n', maxValue);
fprintf('Mean Squared Error = %0.3f degrees\n',sysc.Report
   .Fit.MSE);
TransferFunction = tf(sysc.Numerator,sysc.Denominator)

Poles = pole(sysc)
Zeros = zero(sysc)

% Plot transfer function step response
figure(2)
step(sysc)
grid minor
datacursormode on
set(gca, 'FontSize', 18)
title('Step Response')
xlabel('time')
ylabel('Amplitude')
legend({'Continuous Response'},'Location','southeast')

% Plot characteristics
Continuous_Model_Info=stepinfo(sysc)

% Pole-zero plot
figure(3)
set(gca, 'FontSize', 18)
pzmap(sysc)
```

```
figure(4)
set(gca, 'FontSize', 18)
bode(sysc)
```

C.3 Motor In-Motion Data Analysis Script

```
clc;
clear;
close all;

file = 'C:\Users\Austin\Downloads\DataFiles\Matlab\
   MotorMotionData\MotorMotionDataValidation_10.csv';
A = load(file);

t1 = A(:,1);
tempCamera1 = A(:,2);
motorOffset = 0.3;
camera1 = tempCamera1 - motorOffset;
encoder1 = A(:,3);
encoder2 = circshift(encoder1,-1);
t1(end) = [];
camera1(end) = [];
encoder2(end) = [];

figure(1)
scatter(t1/100,camera1,'r','.')
hold on
scatter(t1/100,encoder2,'b','.')
set(gca, 'FontSize', 16)
title('Motor Encoder Data Validation "In Motion"')
xlabel('time (s)')
```

```
ylabel('Motor Angular Position (Degrees)')
legend('Camera Position','Encoder Position','Location','
   southeast')

figure(2)
difference = camera1 - encoder2;
scatter(encoder2,difference,'k','filled');
hold on
plot(encoder2, difference,'k');
set(gca, 'FontSize', 16)
xlim([40 320])
title('Motor Encoder Position Difference Optitrack vs
   Encoder')
xlabel('Encoder Position (Degrees)')
ylabel('Optitrack - Encoder (Degrees)')

modelFit = fitlm(camera1,encoder2)
figure(3)
plot(modelFit)
set(gca, 'FontSize', 16)
title('Motor Position Optitrack vs Encoder Correlation')
xlabel('Optitrack Position (Degrees)')
ylabel('Encoder Position (Degrees)')

MinDifference = min(abs(difference))
MaxDifference = max(abs(difference))
DifferenceMean = mean(difference)
```

```
file2 = 'C:\Users\Austin\Downloads\DataFiles\Matlab\
   AnalysisTables\Motor.csv';
D = load(file2);
CMean = D(:,1);
EMean = D(:,2);
differenceMean = CMean-EMean;


figure(4)
scatter(EMean,differenceMean,'k','filled');
hold on
plot(EMean, differenceMean,'k');
xlim([40 320])
set(gca, 'FontSize', 16)
title('Motor Position Difference in Means')
xlabel('Encoder Position (Degrees)')
ylabel('Optitrack - Encoder (Degrees)')
```

C.4 Motor Stationary Data Analysis Script

```
clc;
clear;
close all;

file = 'C:\Users\Austin\Downloads\DataFiles\Matlab\
   MotorStationaryData\MotorStationaryDataValidation_23.
   csv';
B = load(file);

t2 = B(:,1);
camera2 = B(:,2);
encoder3 = B(:,3);
encoder4 = circshift(encoder3,-1);
t2(end) = [];
camera2(end) = [];
encoder4(end) = [];

figure(1)
scatter(t2/40,camera2,'r','filled')
hold on
scatter(t2/40,encoder4,'b','filled')
plot(t2/40,camera2,'r')
hold on
plot(t2/40,encoder4,'b')
set(gca, 'FontSize', 16)
```

```
title('Motor Encoder Data Validation "Stationary"')
xlabel('time (s)')
ylabel('Motor Angular Position (Degrees)')
legend('Camera Position','Encoder Position','Location','
   southeast')

% figure(2)
% movavgCamera = movavg(camera2,'simple',20);
% plot(t2,movavgCamera)
% hold on
% movavgEncoder = movavg(encoder4,'simple',20);
% plot(t2,movavgEncoder)
% set(gca, 'FontSize', 16)
% title('Motor Encoder Data Validation "Stationary" Moving
    Average')
% xlabel('time')
% ylabel('Motor Angular Position (Degrees)')
% legend('Camera Position','Encoder Position','Location','
   southeast')

cameraAverage = mean(camera2);
encoderAverage = mean(encoder4);
difference = cameraAverage-encoderAverage;

cameraStd = std(camera2);
encoderStd = std(encoder4);
```

```
fprintf('\nCamera Average = %f degrees',cameraAverage);
fprintf('\nCamera Variance = %f degrees',cameraStd^2);
fprintf('\nEncoder Average = %f degrees',encoderAverage);
fprintf('\nEncoder Variance = %f degrees\n',encoderStd^2);
fprintf('\nDifference in Averages = %f degrees\n',
   difference);
```

C.5 IMU In-Motion Data Analysis Script

```
clc;
clear;
close all;

file = 'C:\Users\Austin\Downloads\DataFiles\Matlab\
   IMUMotionData\IMUMotionDataValidation_1.csv';
C = load(file);

t3 = C(:,1);
tempCameraThighIMU = C(:,2);
thighOffset = 0.95;
cameraThigh = tempCameraThighIMU - thighOffset;
labviewThighIMU = C(:,3);

tempCameraTorsoIMU = C(:,4);
torsoOffset = 2.0;
cameraTorso = tempCameraTorsoIMU - torsoOffset;
labviewTorsoIMU = C(:,5);

tempMovAvgLabviewThigh = movavg(labviewThighIMU,'simple'
   ,5);
tempMovAvgLabviewTorso = movavg(labviewTorsoIMU,'simple'
   ,5);
movavgLabviewThigh = circshift(tempMovAvgLabviewThigh,-3);
movavgLabviewTorso = circshift(tempMovAvgLabviewTorso,-3);
```

```
t3(end) = [];
cameraThigh(end) = [];
movavgLabviewThigh(end) = [];
cameraTorso(end) = [];
movavgLabviewTorso(end) = [];

figure(1);
scatter(t3,cameraThigh,'r','filled')
hold on
scatter(t3,movavgLabviewThigh,'b','filled')
plot(t3,cameraThigh,'r')
hold on
plot(t3,movavgLabviewThigh,'b')
set(gca, 'FontSize', 16)
xlim([0 95])
title('Thigh IMU Data Validation "In Motion"')
xlabel('time')
ylabel('Thigh Pitch IMU (Degrees)')
legend('Camera Measurement','IMU Measurement','Location','
   northeast')

figure(2);
scatter(t3,cameraTorso,'r','filled')
hold on
scatter(t3,movavgLabviewTorso,'b','filled')
plot(t3,cameraTorso,'r')
```

```
hold on
plot(t3,movavgLabviewTorso,'b')
set(gca, 'FontSize', 16)
xlim([0 95])
title('Torso IMU Data Validation "In Motion"')
xlabel('time')
ylabel('Torso IMU Pitch (Degrees)')
legend('Camera Measurement','IMU Measurement','Location','
   northeast')

figure(3)
difference = cameraThigh - movavgLabviewThigh;
plot(t3, difference)
set(gca, 'FontSize', 16)
title('Thigh Angle Difference Optitrack vs IMU')
xlabel('time')
ylabel('Optitrack - IMU (Degrees)')

figure(4)
difference = cameraTorso - movavgLabviewTorso;
plot(t3, difference)
set(gca, 'FontSize', 16)
title('Torso Angle Difference Optitrack vs IMU')
xlabel('time')
ylabel('Optitrack - IMU (Degrees)')

modelFit1 = fitlm(cameraThigh,movavgLabviewThigh)
```

```
figure(5)
plot(modelFit1)
set(gca, 'FontSize', 16)
title('Thigh Angle Optitrack vs IMU Correlation')
xlabel('Optitrack Angle (Degrees)')
ylabel('IMU Angle (Degrees)')

modelFit2 = fitlm(cameraTorso,movavgLabviewTorso)
figure(6)
plot(modelFit2)
set(gca, 'FontSize', 16)
title('Torso Angle Optitrack vs IMU Correlation')
xlabel('Optitrack Angle (Degrees)')
ylabel('IMU Angle (Degrees)')
```

C.6 IMU Stationary Data Analysis Script

```
clc;
clear;
close all;

file = 'C:\Users\Austin\Downloads\DataFiles\Matlab\
   IMUStationaryData\IMUStationaryDataValidation_2.csv';
D = load(file);

t4 = D(:,1);
tempCameraThighIMU = D(:,2);
thighOffset = -0.81;
cameraThighIMU = tempCameraThighIMU - thighOffset;
labviewThighIMU = D(:,3);

tempCameraTorsoIMU = D(:,4);
torsoOffset = -0.16;
cameraTorsoIMU = tempCameraTorsoIMU - torsoOffset;
labviewTorsoIMU = D(:,5);

figure(1);
scatter(t4/40,cameraThighIMU,'r','filled')
hold on
scatter(t4/40,labviewThighIMU,'b','filled')
plot(t4/40,cameraThighIMU,'r')
hold on
```

```
plot(t4/40,labviewThighIMU,'b')
set(gca, 'FontSize', 16)
title('Thigh IMU Data Validation "Stationary"')
xlabel('time (s)')
ylabel('Thigh Pitch IMU (Degrees)')
legend('Camera Measurement','IMU Measurement','Location','
   east')

figure(2);
scatter(t4/40,cameraTorsoIMU,'r','filled')
hold on
scatter(t4/40,labviewTorsoIMU,'b','filled')
plot(t4/40,cameraTorsoIMU,'r')
hold on
plot(t4/40,labviewTorsoIMU,'b')
set(gca, 'FontSize', 16)
title('Torso IMU Data Validation "Stationary"')
xlabel('time (s)')
ylabel('Torso Pitch IMU (Degrees)')
legend('Camera Measurement','IMU Measurement','Location','
   east')

cameraAverageThigh = mean(cameraThighIMU);
labviewAverageThigh = mean(labviewThighIMU);
differenceThigh = cameraAverageThigh - labviewAverageThigh
   ;
```

```
cameraStdThigh = std(cameraThighIMU);
labviewStdThigh = std(labviewThighIMU);

fprintf('\nCamera Average Thigh = %f degrees',
   cameraAverageThigh);
fprintf('\nCamera Variance Thigh = %f degrees',
   cameraStdThigh^2);
fprintf('\nIMU Average Thigh = %f degrees',
   labviewAverageThigh);
fprintf('\nIMU Variance Thigh = %f degrees\n',
   labviewStdThigh^2);
fprintf('\nDifference in Averages = %f degrees\n',
   differenceThigh);

cameraAverageTorso = mean(cameraTorsoIMU);
labviewAverageTorso = mean(labviewTorsoIMU);
differenceTorso = cameraAverageTorso - labviewAverageTorso
   ;

cameraStdTorso = std(cameraTorsoIMU);
labviewStdTorso = std(labviewTorsoIMU);

fprintf('\nCamera Average Torso = %f degrees',
   cameraAverageTorso);
fprintf('\nCamera Variance Torso = %f degrees',
   cameraStdTorso^2);
fprintf('\nIMU Average Torso = %f degrees',
```

```
   labviewAverageTorso);
fprintf('\nIMU Variance Torso = %f degrees\n',
   labviewStdTorso^2);
fprintf('\nDifference in Averages = %f degrees\n',
   differenceTorso);
```

C.7 Script used to obtain all motion data

```
clc;
clear;
close all;

trialNumber = 11;
trialType = 'FullSystemData';

% Setup file paths
curDir = pwd;
mainDir = fileparts(fileparts(curDir));
dllPath = fullfile(mainDir,'lib','x64','NatNetML.dll');
assemblyInfo = NET.addAssembly(dllPath);
theClient = NatNetML.NatNetClientML(0);

% Connect to Optitrack Motive data stream.
LocalIP = char('10.17.179.57');
ServerIP = char('10.17.179.61');
theClient.Initialize(LocalIP, ServerIP);

% Connect to LabVIEW OPC server
da = opcda('localhost','National Instruments.Variable
   Engine.1');
connect(da);

% Set Datalogging constraints
```

```
loopTime = 2.5;
pauseTime = 0.025;
loopSize = loopTime/pauseTime;
Data = zeros(loopSize,1);
pause(3);   % Allow 3 seconds to get into position
fprintf('Datalogging in progress');

% Perform datalogging with data from motiveData and
   labviewData scripts
for n = 1:1:loopSize
    if mod(n,10) == 0
        fprintf('.');
    end

    motiveData
    labviewData

    pause(pauseTime)
end
fprintf('\n');

% Assign necessary data from Motive data matrices
motorRoll = cameraDataMotor(:,4);

thighPitch = cameraDataThigh(:,2);
thighYaw = cameraDataThigh(:,3);
thighRoll = cameraDataThigh(:,4);
```

```
torsoPitch = cameraDataTorso(:,2);
torsoYaw = cameraDataTorso(:,3);
torsoRoll = cameraDataTorso(:,4);

netPitch = thighPitch - torsoPitch;
netPitch = netPitch + 180;

% Assign necessary data from LabVIEW data matrices
% encoderData = Data1(:,2);
% thighData = Data2(:,2);
% torsoData = Data2(:,3);
% netData = Data2(:,4);

netData = Data3(:,2);
encoderData = Data3(:,3);
t = (1:1:loopSize)';

% % Plot data
% figure(1);
% plot(t,motorRoll,t,encoderData);
% title('Motor Data Validation')
% xlabel('time')
% ylabel('Motor Angular Position (Degrees)')
% legend('Camera Data','Encoder Data');
%
% figure(2);
```

```
% plot(t,thighPitch,t,thighData);
% title('Thigh IMU Data Validation')
% xlabel('time')
% ylabel('Thigh Pitch IMU (Degrees)')
% legend('Camera Measurement','IMU Measurement','Location
   ','southeast')
%
% figure(3);
% plot(t,torsoPitch,t,torsoData);
% title('Torso IMU Data Validation')
% xlabel('time')
% ylabel('Torso IMU Pitch (Degrees)')
% legend('Camera Measurement','IMU Measurement','Location
   ','southeast')

% % Assign data to a single combo matrix
% dataMatrix1 = [t,motorRoll(:),encoderData(:)];
% dataMatrix2 = [t,thighPitch(:),thighData(:),torsoPitch
   (:),torsoData(:)];

figure(1)
plot(t, motorRoll,t,encoderData);
legend('Camera Measurement','Encoder Measurement','
   Location','southeast')

figure(2)
plot(t,netPitch,t,netData);
```

```
legend('Camera Measurement','IMU Measurement','Location','
    southeast')

% dataMatrix2 = [t,thighPitch,thighData,torsoPitch,
    torsoData,netPitch,netData];
dataMatrix3 = [t,motorRoll,encoderData,thighPitch,
    torsoPitch,netPitch,netData];

% Save data to file
file = sprintf('%s_%0.0f.csv',trialType, trialNumber);
writematrix(dataMatrix3,file);

% Disconnect client from Motive data stream
theClient.Uninitialize;

% Disconnect from LabVIEW OPC server
disconnect(da);
```

C.8 Script used to obtain LabVIEW data

```
% Add Encoder, IMU, and Combo shared variables as items in
    a group
grp = addgroup(da);
% itm1 = additem(grp,'\\.\DataLib\EncoderLocal');
% itm2 = additem(grp,'\\.\DataLib\IMULocal');
itm3 = additem(grp,'\\.\DataLib\ComboLocal');

% % Encoder datalogging
% x1 = read(itm1);          % Read encoder item
% y1 = x1.Value;            % Assign the data to y1
% Data1(n,1) = y1(1,1);     % Assign LabVIEW time data to
   1st column of y1
% Data1(n,2) = y1(1,2);     % Assign encoder position to 2
   nd column of y1
%
% % IMU(2) datalogging
% x2 = read(itm2);            % Read IMU item
% y2 = x2.Value;              % Assign the data to y2
% Data2(n,1) = y2(1,1);       % Assign LabVIEW time data
   to 1st column of y2
% Data2(n,2) = y2(1,2);       % Assign thigh pitch to 2nd
   column of y2
% Data2(n,3) = y2(1,3);       % Assign torso pitch to 3rd
   column of y2
% Data2(n,4) = y2(1,4);       % Assign net pitch to 4th
```

```
   column of y2

% Full system datalogging
x3 = read(itm3);            % Read encoder/IMU combo item
y3 = x3.Value;              % Assign the data to y3
Data3(n,1) = y3(1,1);       % Assign LabVIEW time data to 1
   st column of y3
Data3(n,2) = y3(1,2);       % Assign net pitch to 4th column
    of y3
Data3(n,3) = y3(1,3);       % Assign encoder position to 2nd
    column of y3
```

C.9 Script used to obtain Motive camera data

```
% Assign Rigid Bodies from Motive to identifyer numbers
RigidBody_Motor = 1;
RigidBody_Thigh = 2;
RigidBody_Torso = 3;

frameOfData = theClient.GetLastFrameOfData();
rigidBodyData1 = frameOfData.RigidBodies(RigidBody_Motor);
rigidBodyData2 = frameOfData.RigidBodies(RigidBody_Thigh);
rigidBodyData3 = frameOfData.RigidBodies(RigidBody_Torso);

% Calculate pitch, yaw, and roll angles for Motor Rigid
   Body
q1 = quaternion(rigidBodyData1.qx, rigidBodyData1.qy,
   rigidBodyData1.qz, rigidBodyData1.qw); % extrnal file
   quaternion.m
qRot1 = quaternion(0, 0, 0, 1);    % rotate pitch 180 to
   avoid 180/-180 flip for nicer graphing
q1 = mtimes(q1, qRot1);
angles1 = EulerAngles(q1,'zyx');
MotorPitch = -angles1(1) * 180.0 / pi;  % must invert due
   to 180 flip
MotorYaw = angles1(2) * 180.0 / pi;
MotorRoll = -angles1(3) * 180.0 / pi;  % must invert due
   to 180 flip
```

```
% Calculate pitch, yaw, and roll angles for Thigh Rigid
   Body
q2 = quaternion(rigidBodyData2.qx, rigidBodyData2.qy,
   rigidBodyData2.qz, rigidBodyData2.qw); % extrnal file
   quaternion.m
qRot2 = quaternion(0, 0, 0, 1);    % rotate pitch 180 to
   avoid 180/-180 flip for nicer graphing
q2 = mtimes(q2, qRot2);
angles2 = EulerAngles(q2,'zyx');
ThighPitch = -angles2(1) * 180.0 / pi;  % must invert due
   to 180 flip
ThighYaw = angles2(2) * 180.0 / pi;
ThighRoll = -angles2(3) * 180.0 / pi;  % must invert due
   to 180 flip

% Calculate pitch, yaw, and roll angles for Torso Rigid
   Body
q3 = quaternion(rigidBodyData3.qx, rigidBodyData3.qy,
   rigidBodyData3.qz, rigidBodyData3.qw); % extrnal file
   quaternion.m
qRot3 = quaternion(0, 0, 0, 1);    % rotate pitch 180 to
   avoid 180/-180 flip for nicer graphing
q3 = mtimes(q3, qRot3);
angles3 = EulerAngles(q3,'zyx');
TorsoPitch = -angles3(1) * 180.0 / pi;  % must invert due
   to 180 flip
TorsoYaw = angles3(2) * 180.0 / pi;
```

```
TorsoRoll = -angles3(3) * 180.0 / pi;  % must invert due
   to 180 flip

% Translate (-180 - 180) degree angles to (0 - 360)
   degrees
if MotorRoll < 0
    MotorRoll = 180 - abs(MotorRoll) + 180;
else
    MotorRoll = MotorRoll;
end

% Build data matrices from each frame of calculated data
cameraDataMotor(n,1) = frameOfData.fTimestamp;
cameraDataMotor(n,2) = MotorPitch;
cameraDataMotor(n,3) = MotorYaw;
cameraDataMotor(n,4) = MotorRoll;

cameraDataThigh(n,1) = frameOfData.fTimestamp;
cameraDataThigh(n,2) = ThighPitch;
cameraDataThigh(n,3) = ThighYaw;
cameraDataThigh(n,4) = ThighRoll;

cameraDataTorso(n,1) = frameOfData.fTimestamp;
cameraDataTorso(n,2) = TorsoPitch;
cameraDataTorso(n,3) = TorsoYaw;
cameraDataTorso(n,4) = TorsoRoll;
```

APPENDIX D: DATA RESULTS: DATA VALIDATION

D.1 Collective data of 10 motor in-motion datasets

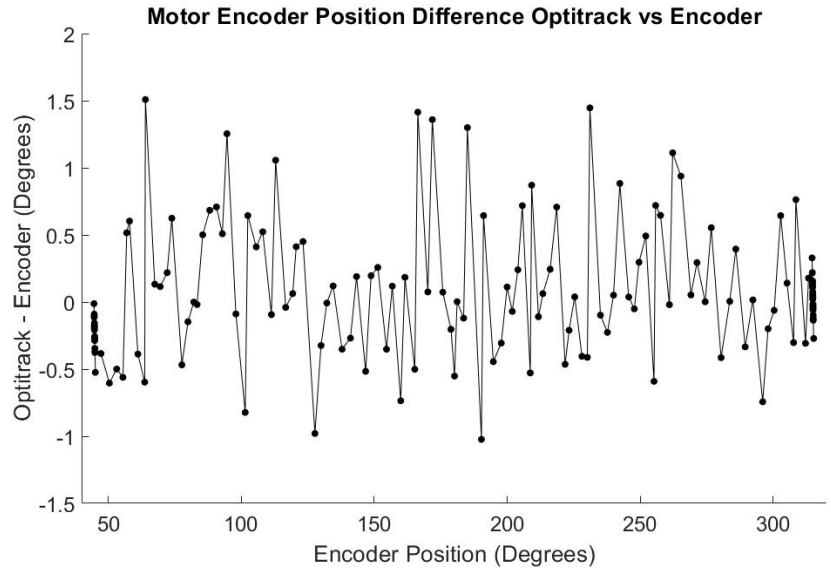


Figure D.1: Trial 1: Motor in-motion difference in measured angle.

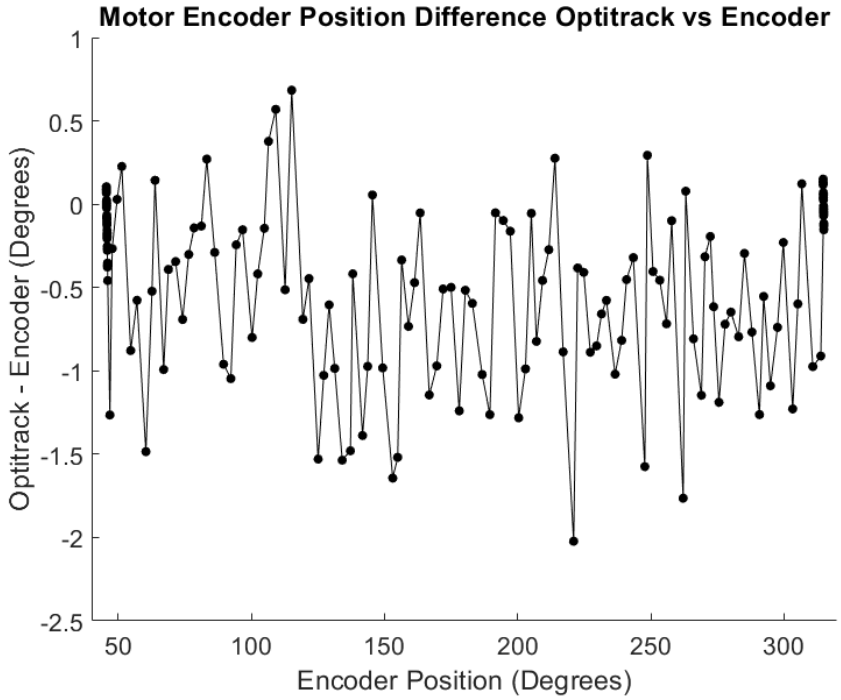


Figure D.2: Trial 2: Motor in-motion difference in measured angle.

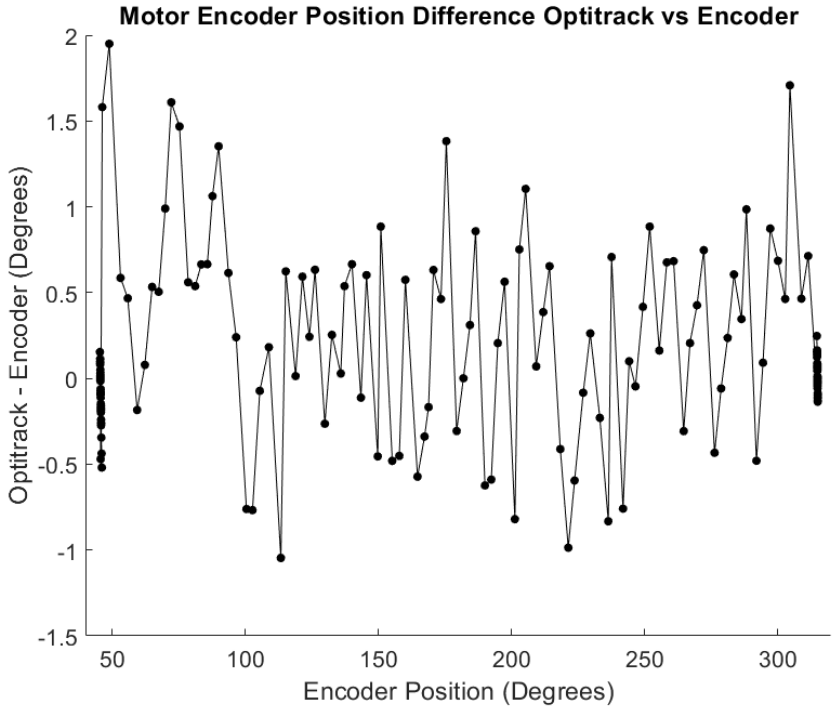


Figure D.3: Trial 3: Motor in-motion difference in measured angle.

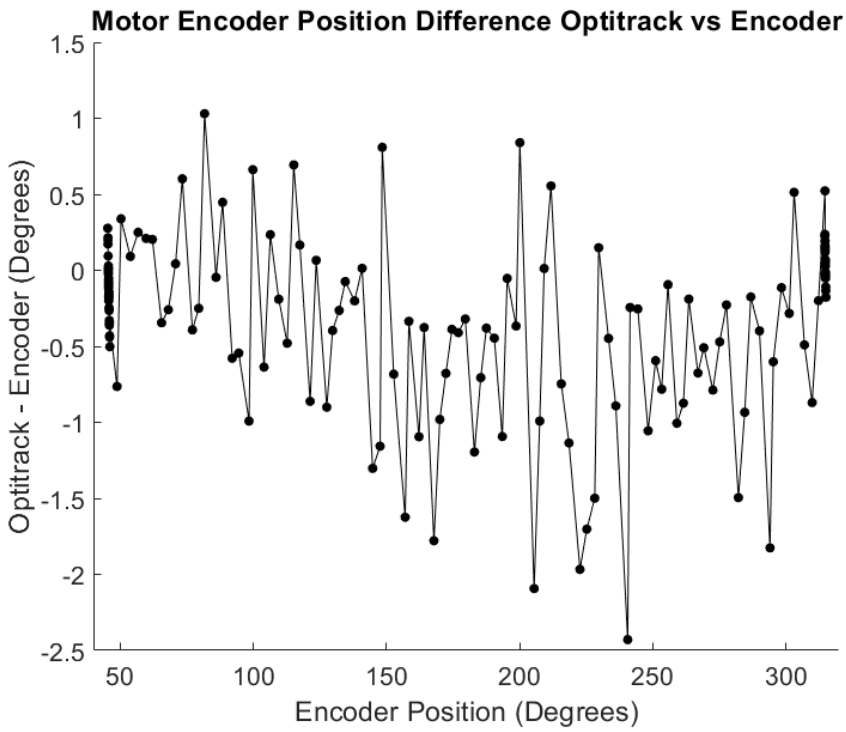


Figure D.4: Trial 4: Motor in-motion difference in measured angle.

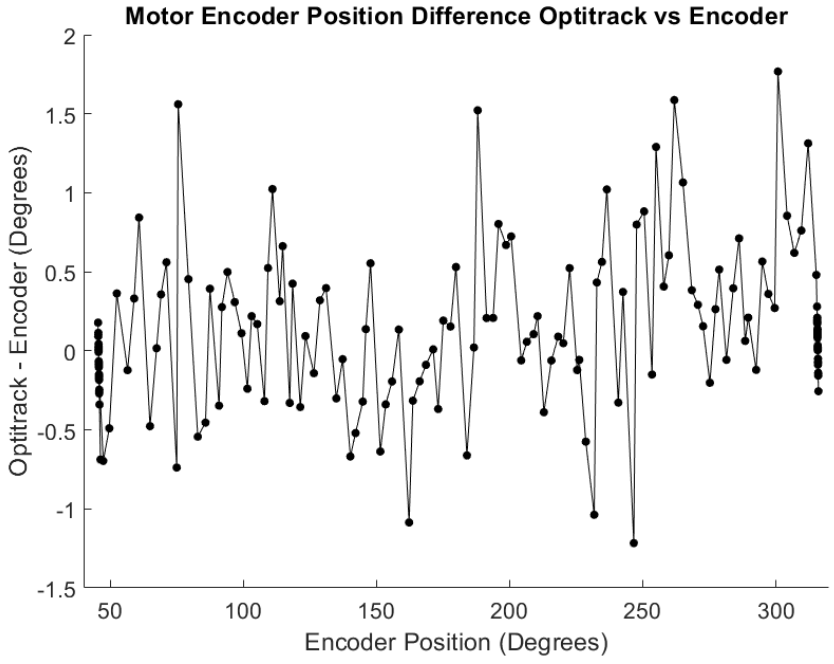


Figure D.5: Trial 5: Motor in-motion difference in measured angle.

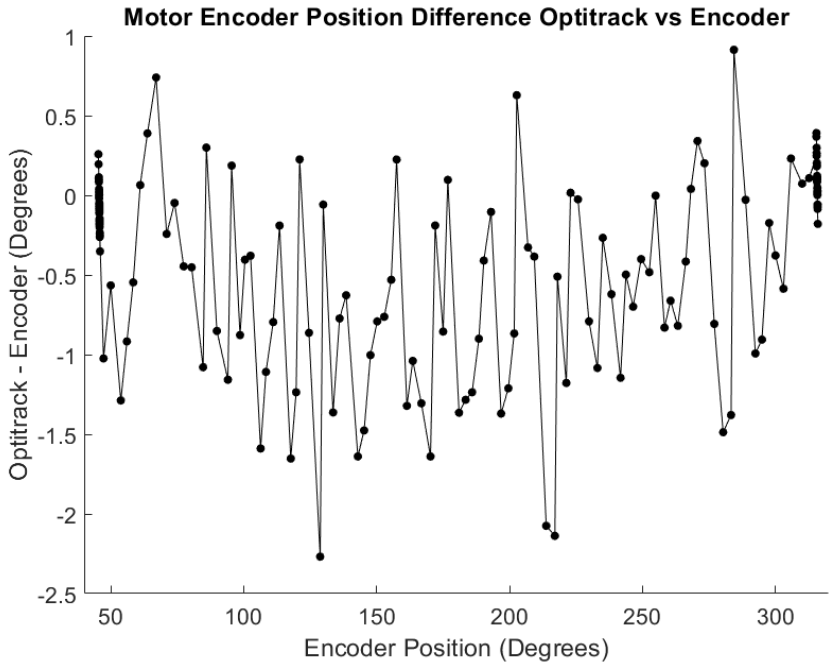


Figure D.6: Trial 6: Motor in-motion difference in measured angle.

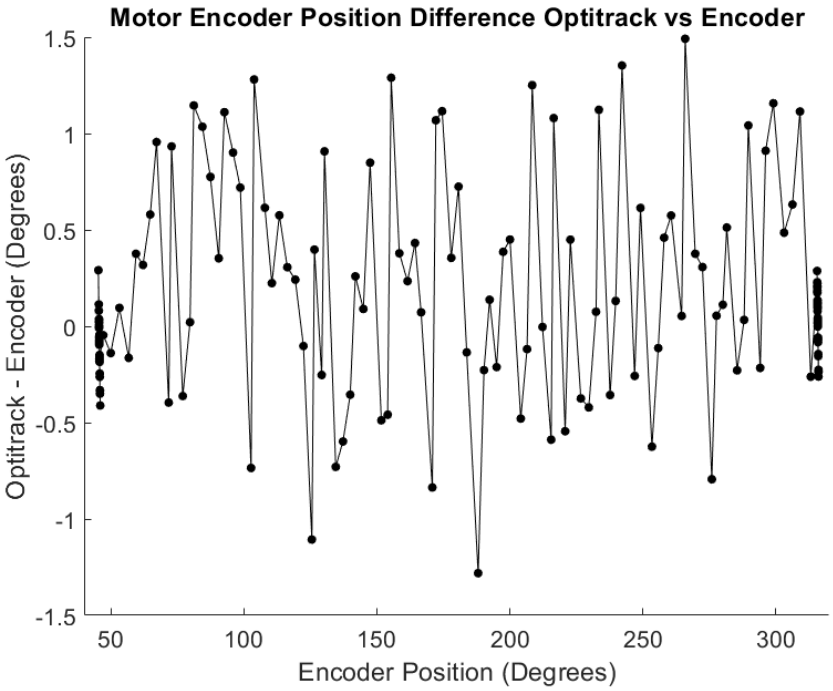


Figure D.7: Trial 7: Motor in-motion difference in measured angle.

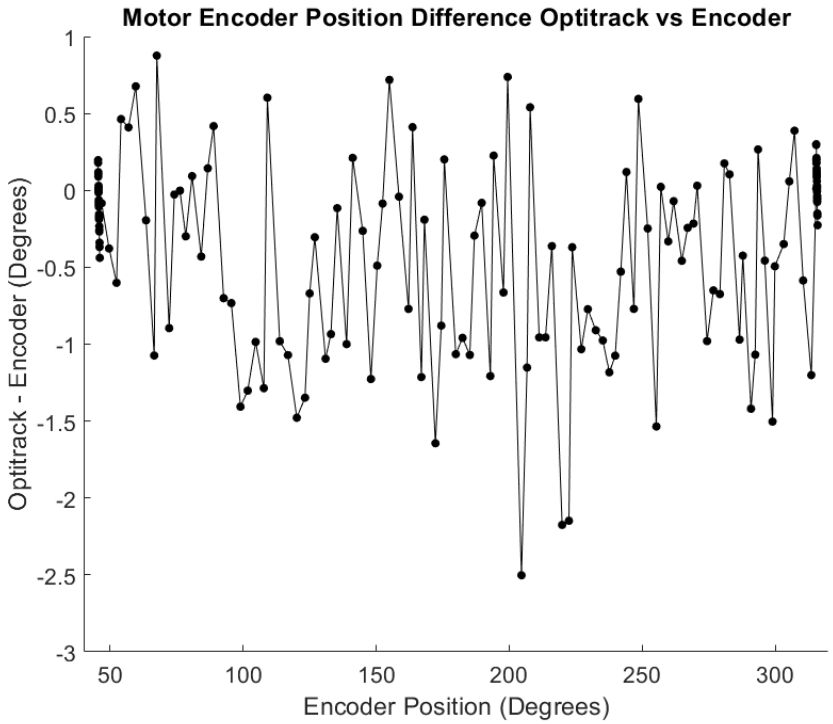


Figure D.8: Trial 8: Motor in-motion difference in measured angle.

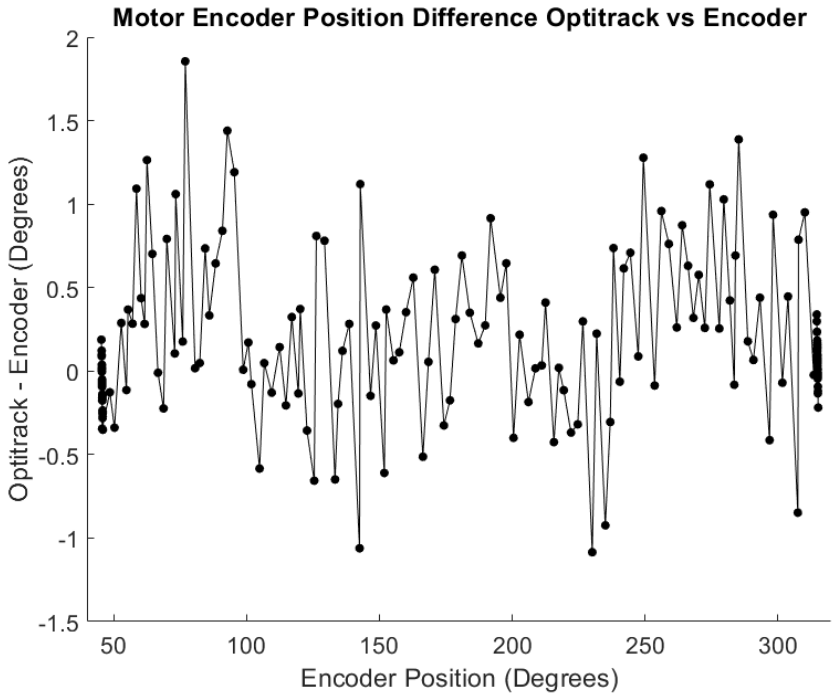


Figure D.9: Trial 9: Motor in-motion difference in measured angle.

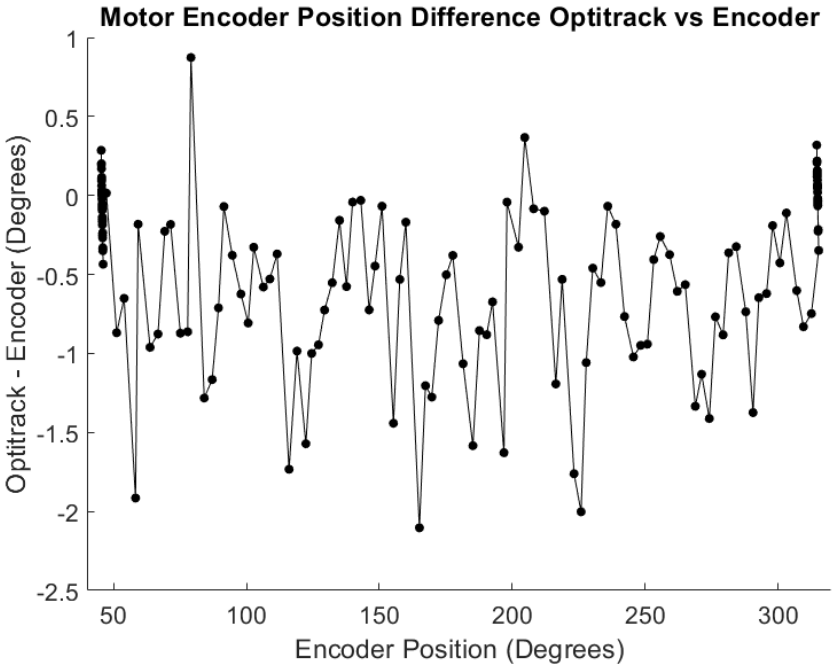


Figure D.10: Trial 10: Motor in-motion difference in measured angle.

D.2 Collective data of 28 motor stationary datasets

	Camera Data		Encoder Data			
Dataset	Mean Angle	Variance	Mean Angle	Variance	Difference in Means	% Difference
1	44.954105	0.000284	44.904119	0.010734	0.049986	0.111255
2	55.268620	0.000159	55.074574	0.012547	0.194047	0.351715
3	65.370111	0.000186	65.047053	0.014360	0.323059	0.495424
4	75.327606	0.000147	75.146484	0.013085	0.181122	0.240735
5	85.400841	0.000194	85.224609	0.013400	0.176231	0.206571
6	95.454859	0.000081	95.275213	0.010244	0.179646	0.188377
7	105.318573	0.000061	105.373757	0.009250	-0.055184	-0.052383
8	115.521219	0.003192	115.520241	0.008427	0.000977	0.000846
9	125.218518	0.000058	125.483842	0.015106	-0.265324	-0.211665
10	135.560550	0.000109	135.629439	0.013512	-0.068889	-0.050805
11	145.591839	0.000127	145.669389	0.011953	-0.077550	-0.053251
12	155.659509	0.005943	155.731534	0.008241	-0.072025	-0.046260
13	165.879006	0.000116	165.923295	0.013512	-0.044290	-0.026697
14	176.003829	0.000132	175.862038	0.015714	0.141791	0.080594
15	186.239004	0.000115	186.109730	0.014403	0.129274	0.069437
16	196.240233	0.000120	196.041371	0.011408	0.198862	0.101387
17	206.413905	0.000060	206.218040	0.012841	0.195865	0.094934
18	216.362973	0.000133	216.282848	0.011723	0.080125	0.037040
19	226.268790	0.000089	226.320135	0.014816	-0.051345	-0.022689
20	236.504060	0.000049	236.490589	0.016803	0.013471	0.005696
21	246.597144	0.000143	246.439098	0.011390	0.158046	0.064111
22	256.947158	0.000110	256.661932	0.012153	0.285226	0.111067
23	267.208452	0.000149	266.680575	0.009905	0.527877	0.197748
24	276.953591	0.000507	276.791548	0.008800	0.162043	0.058526
25	286.994919	0.000297	286.850142	0.012145	0.144777	0.050459
26	297.360700	0.000127	296.911399	0.015083	0.449300	0.151210
27	307.407451	0.000291	307.016158	0.007994	0.391293	0.127369
28	317.430657	0.000121	317.134233	0.011672	0.296425	0.093426

APPENDIX E: FULL SYSTEM MOTION RESULTS

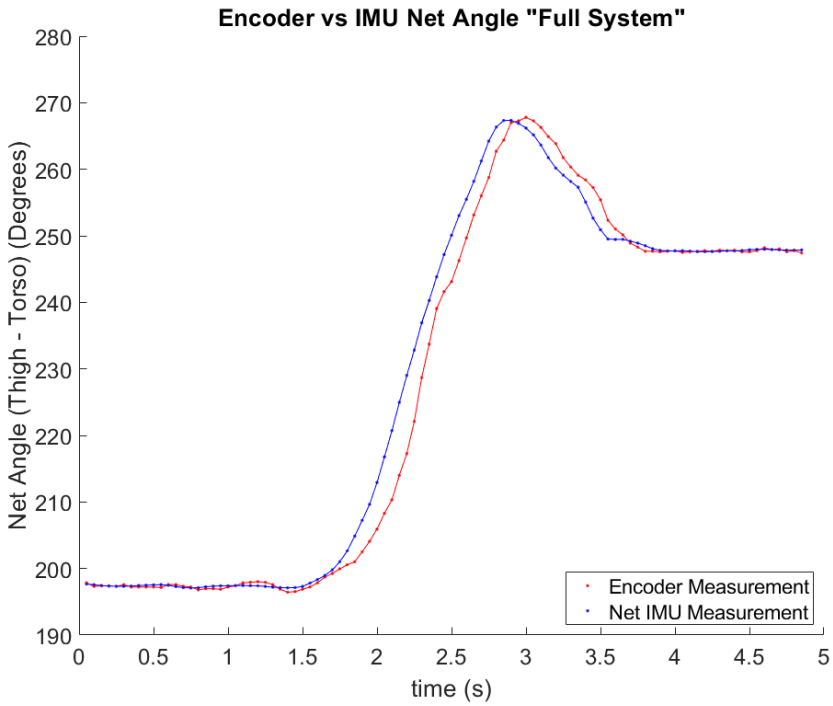


Figure E.1: Trial 1: Full system operation - IMU vs. encoder position.

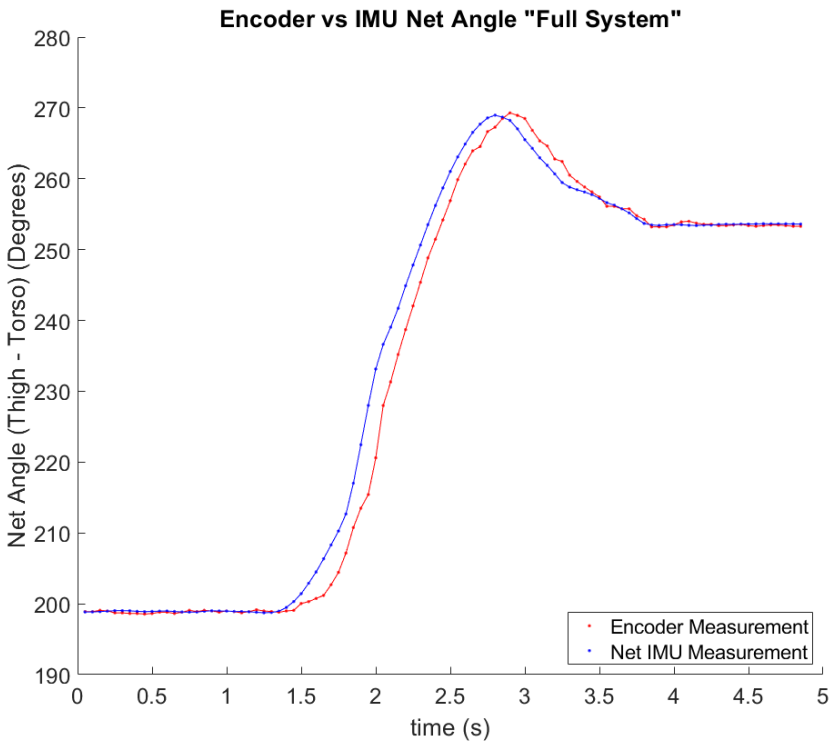


Figure E.2: Trial 2: Full system operation - IMU vs. encoder position.

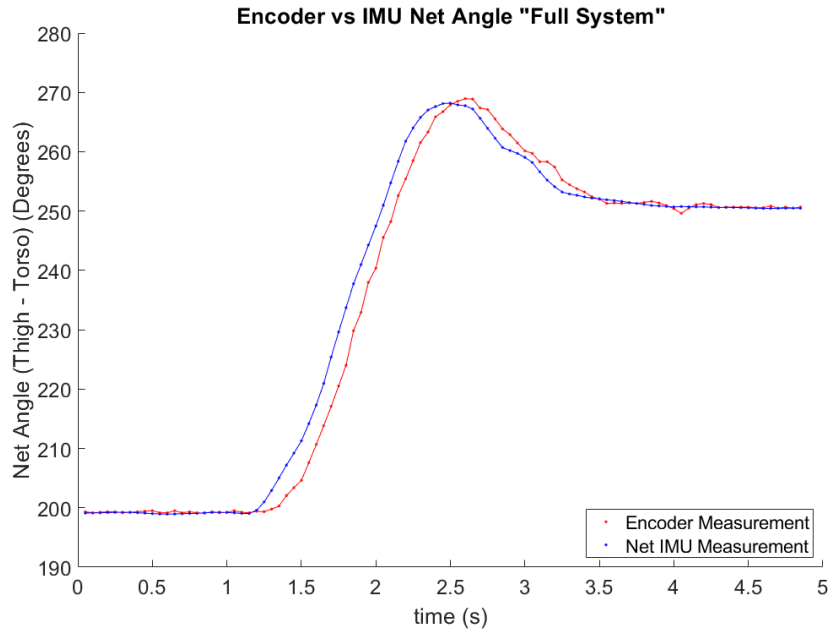


Figure E.3: Trial 3: Full system operation - IMU vs. encoder position.

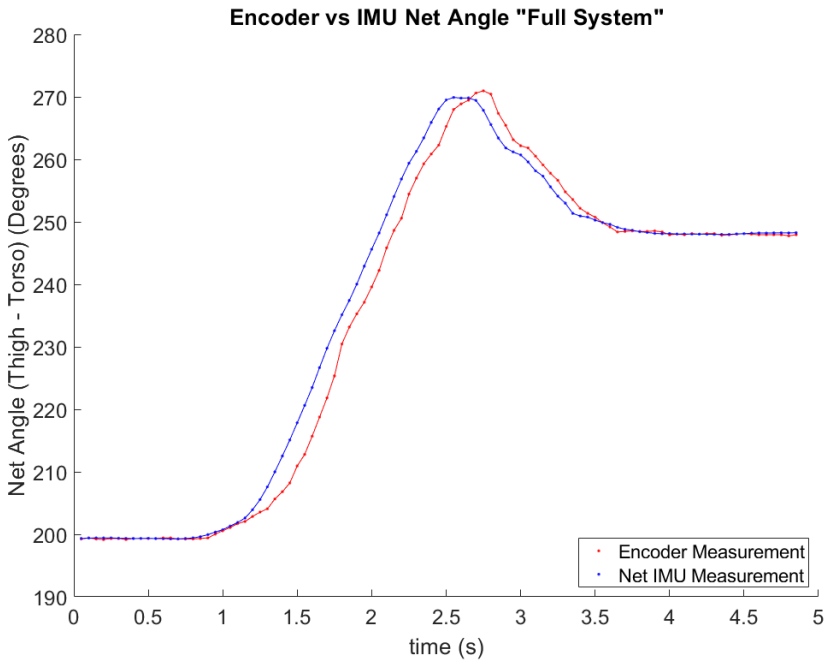


Figure E.4: Trial 4: Full system operation - IMU vs. encoder position.

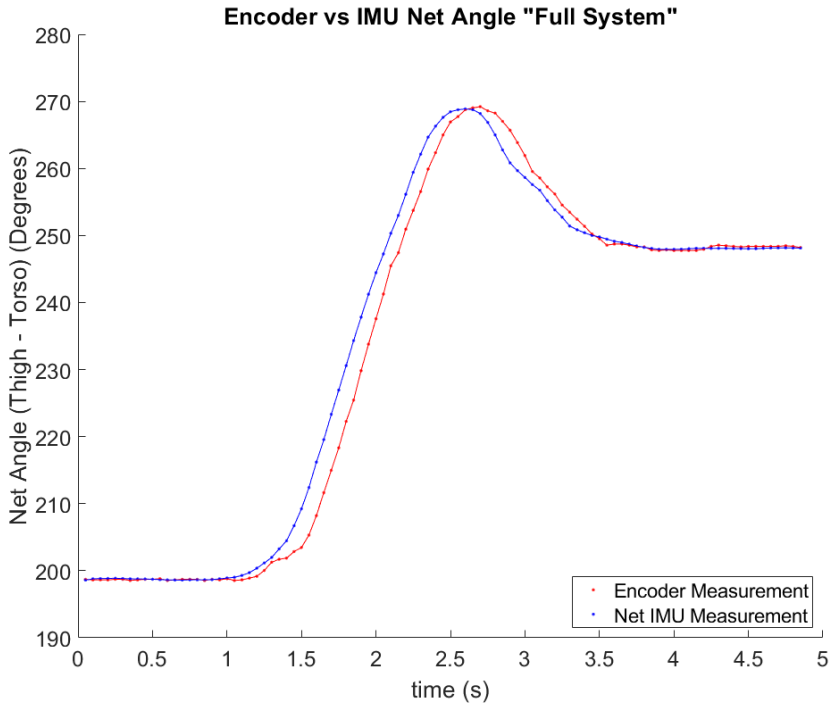


Figure E.5: Trial 5: Full system operation - IMU vs. encoder position.

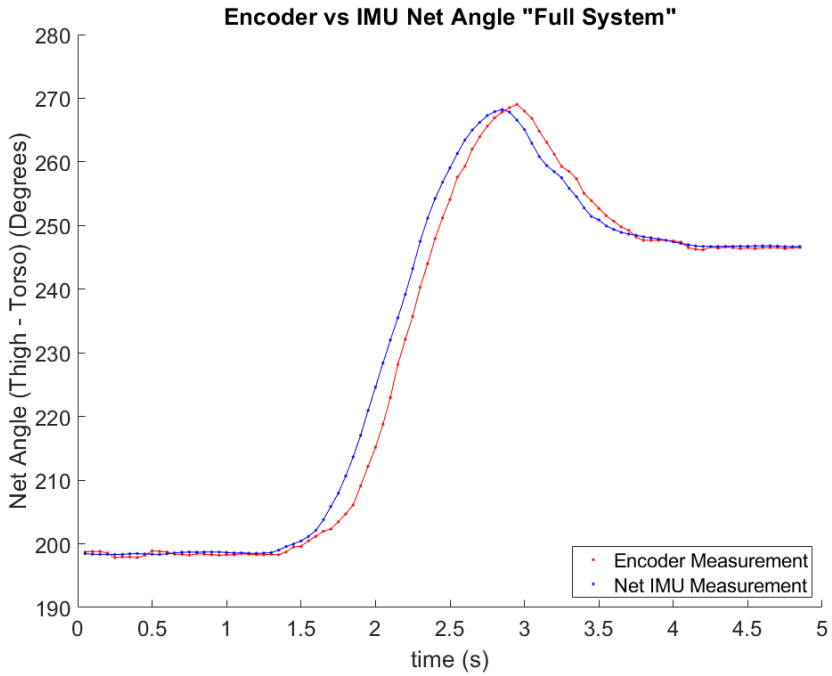


Figure E.6: Trial 6: Full system operation - IMU vs. encoder position.

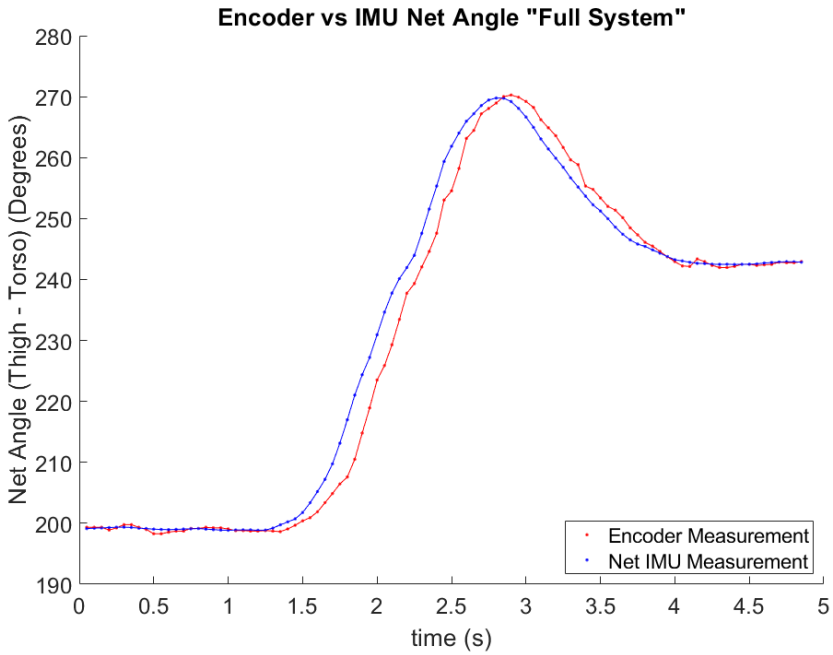


Figure E.7: Trial 7: Full system operation - IMU vs. encoder position.

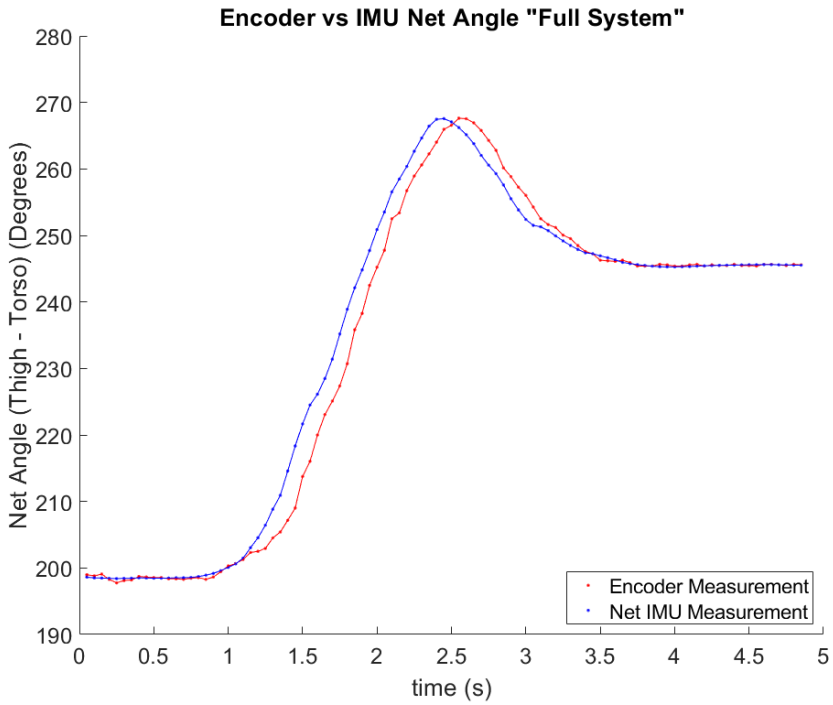


Figure E.8: Trial 8: Full system operation - IMU vs. encoder position.

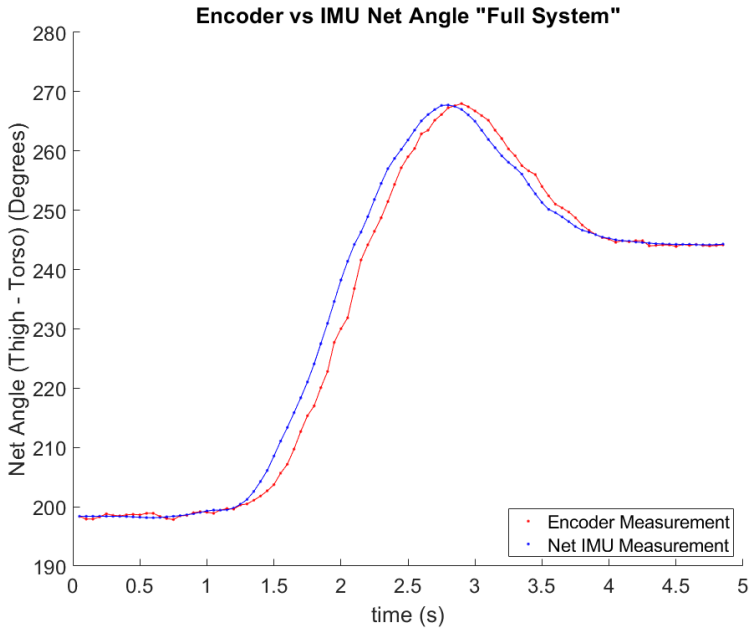


Figure E.9: Trial 9: Full system operation - IMU vs. encoder position.

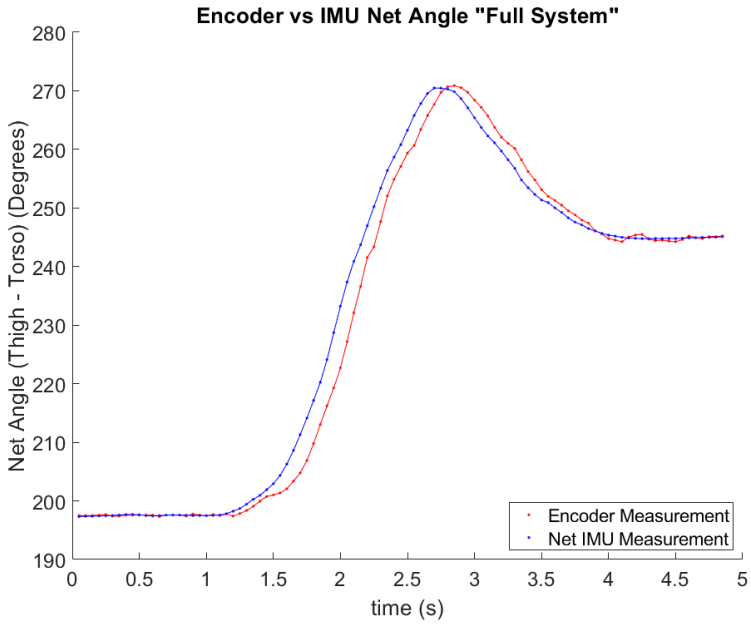


Figure E.10: Trial 10: Full system operation - IMU vs. encoder position.