Adaptive Symmetry Learning and Data-Driven Predictive Models for Personalized Control of Robotic Ankle-Foot Prostheses

Christopher Prasanna

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

Master of Science in Mechanical Engineering

University of Washington

2022

Reading Committee:

Glenn Klute, Chair

Eric Rombokas

Jonathan Realmuto

Katherine Steele

Krithika Manohar

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Adaptive Symmetry Learning and Data-Driven Predictive Models for Personalized Control of Robotic Ankle-Foot Prostheses

Christopher Prasanna

Chair of the Supervisory Committee:
Affiliated Professor Glenn Klute
Department of Mechanical Engineering

The main goal of this research is to develop a personalized and adaptive control scheme for powered ankle-foot prostheses. Individuals with below-knee amputation generally have gait patterns that are more asymmetrical about their two limbs when compared to the general population. This is likely due to a lack of proper prosthetic ankle function. It has been observed that these individuals compensate for this lack of mobility in ways that can lead to a host of long-term musculoskeletal impairments (e.g., osteoarthritis in the knee and hip of the intact limb). Powered prostheses have the ability to better emulate biological ankles since they are capable of generating power, unlike clinically-prescribed limbs that are completely passive and can only return stored energy. However, current powered prosthesis control methods are over-reliant on able-bodied data, require extensive amounts of tuning by experts, and cannot adapt to the user's unique gait patterns. This work directly addresses all the listed limitations with an adaptive and data-driven control strategy. The controller utilizes time-invariant control signals enabled by phase-based state estimation, an impedance-based feedback loop, and trajectory planning that iteratively adjusts the motion of the prosthetic ankle to match the intact ankle motion. This robotic ankle controller was experimentally evaluated on two individuals with below-knee amputation. The control scheme successfully increased ankle angle symmetry about the two limbs, significantly increased peak prosthetic ankle power output at push-off, and significantly reduced factors associated with osteoarthritis in the intact limb. In addition, control signals were bounded, personalized, and unique across subjects without the use of able-bodied data or extensive

tuning. This research also applied deep learning to build models capable of accurately predicting prosthetic ankle torque values from inverse dynamics. Computing inverse dynamics is a time-intensive task, is limited to a laboratory setting, and requires a large array of motion capture markers. The models developed in this research enable total prosthetic ankle torque from inverse dynamics to be observed using signals only from common wearable sensors. This application of machine learning provides an avenue to the development of model-based control systems for powered limbs aimed at optimizing prosthetic ankle torque.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I wish to acknowledge and express my sincere appreciation to the following parties:

- Dr. Glenn Klute for his knowledge, encouragement, feedback, and patience in addition to making this research possible and granting me this opportunity
- Dr. Jonathan Realmuto for allowing me to continue his research, his assistance throughout the project, and his technical support with regards to hardware, system analysis, and the learning algorithm
- Anthony Anderson for his feedback and ideas throughout the project as well as his key contributions to the development and testing of the embedded system software
- Dr. Eric Rombokas for his contagious energy, exciting ideas, inspiration, and contributions toward the machine learning applications of this research
- Krista Cyr for her expertise and assistance with motion capture equipment, human subject research, and biomechanical data analysis
- Christina Carranza for fitting the prosthetic limbs and adjusting their alignment during data collections
- Jane Shofer for her knowledge and contributions toward the statistical analyses of this project
- Professors Katherine Steele and Krithika Manohar for agreeing to serve on my committee on short notice, granting me their attention, and providing their insights toward this research

This research was completed with support from the Veterans Affairs Center for Limb Loss and Mobility resources and funding.

DEDICATION

To my parents for their love, respect, guidance, and inspiration.

Chapter 1

INTRODUCTION

1.1 Motivation

1.1.1 Consequences of Below-Knee Amputation

One of the most significant consequences of below-knee amputations is asymmetrical loading of the lower limbs during ambulation which can lead to a number of secondary conditions [1]. These conditions include degenerative joint diseases in the lower limbs and lower back, such as osteoarthritis (OA) in their intact knee and hip joints, osteopenia and subsequent osteoporosis in the residual limb, and back pain [2]. Traditionally prescribed ankle-foot prostheses rely entirely on passive mechanical components, which currently cannot completely replicate proper human ankle function. For instance, one of the reasons prescribed prostheses are limited is that the passive components responsible for plantarflexion cannot produce the same amount of mechanical power as human ankles [3]. Biological ankle plantar flexors in humans are responsible for over 80% of the power generated during walking and without this, individuals with below-knee amputation exhibit a number of neuromuscular adaptations to compensate for the loss of ankle function [4–7]. These adaptations include less stance time on their prosthetic limb [8–10], overloading their intact limb [11–15], increased hip extension [6], and increased knee flexion on the intact side [16]. Ankle-foot prostheses with improved push-off performance have shown to reduce some of the risk factors associated with the degenerative diseases [17, 18]. In addition, users that wear these prostheses decrease their required metabolic energy and increase their preferred walking speed [19]. Therefore, there is a need to explore prosthesis designs that include and control actively-powered ankles which have the potential to increase mobility and prevent secondary musculoskeletal conditions.

1.1.2 Current Limitations of the State-Of-The-Art

Powered ankle-foot prostheses (PAFPs) can restore some ankle function by utilizing active power (i.e., additional power generated by electrical motors) at the prosthetic joint [20–26].

However, current approaches are limited by:

Lack of Personalization

Emerging PAFPs have limited amounts of customization both in terms of structure and control. Able-bodied individuals have relatively symmetrical angles, torque, and power across the gait cycle for the ankle, knee, and hip joints when compared to individuals with unilateral below-knee amputation. However, individuals with below-knee amputation exhibit different severities of asymmetries and their walking strategies can be considerably different [27]. These differences provide qualitative evidence of the need for personalized approaches, as every individual has a unique neuromuscular strategy [28].

Reliance on Able-Bodied Gait Trajectories and Manual Tuning

Many PAFPs utilize controller trajectories based on able-bodied gait data [29–32]. However, metabolic rate, gait mechanics and muscle activity can vary widely across able-bodied users as well as those with amputated limbs. Thus, able-bodied data should not be the only consideration when formulating control strategies; alternate timing, personalized tuning, personalized subject characteristics, and/or other factors should be taken into consideration [33]. Even when nominal able-bodied trajectories are used, control parameter adjustments still need to be made, usually through trial and error [29, 32]. Commercial PAFPs (e.g., BiOM, BionXMedical Technologies Inc.) must be tuned by a certified prosthetist through their expertise and the user's preferences. However, discovering the best user parameters is challenging since tuning during clinical evaluations fail to demonstrate the same benefits as found in the laboratory [34].

Limitations of PAFP Controllers to Adapt and Act Continuously

Current PAFP devices are limited in terms of how they can adapt to the user and their environment. These devices are typically restricted to changes in slopes [35] or speed [32]. A human-in-the-loop algorithm, where the control signal is learned based on the observed human-robot response, could allow for long-term and more generalizable adaptations [36]. Additionally, flexible control laws, e.g., methods based on time-invariant phase variables [37, 38] or joint impedance [32, 39], can be used not only to increase robustness with respect to speed variations but also allow for a more comprehensive trajectory library, e.g., interpolation for different variations in the environment, loading conditions, or continuous

changes in terrain. This method would also be continuous and unified across the gait cycle without any explicit modes or transitions.

1.2 Research Goals

To address the needs discussed in the previous section, the goals of this research are:

1.2.1 Personalizing the Active Response of the PAFP

A learning algorithm is developed to discover the prosthesis control trajectory that results in symmetrical ankle mechanics. The reasoning is that (i) asymmetries are functionally limiting and are believed to contribute to secondary impairments, thus addressing asymmetries could have beneficial long-term clinical implications, and (ii) asymmetries could be accurately estimated in real time, e.g., with portable low-cost embedded sensors, as opposed to other metrics such as metabolic cost, which is extremely challenging to measure and not feasible outside a laboratory. The research problem is to modify the control signal such that the prosthetic mechanics (i.e., angles, torque, and power) match the biological mechanics. The prototype powered prosthesis takes the control input command and outputs an active torque about the ankle joint, which is summed with the passive torque generated by the device's passive components and equates to the total ankle torque. The control trajectory is iteratively learned by adapting to the biological ankle angles, which act as a reference signal. The symmetry learning controller directly addresses the limitations of using able-bodied gait data and the need for manual tuning by a clinical expert since the controller automatically learns and tunes the control trajectory based on the user. The controller could also conceivably allow for long-term adaptations via additional learning trials over time. Additionally, impedance-based feedback and phase-based control methods are employed to reduce the need for learning every possible condition and allow for adaptations to variations in the user's gait. Additionally, the controller acts continuously, eliminating the need to discretize the gait cycle and tune parameters to each sub-phase.

1.2.2 Experimental Evaluation

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

- **H1.1** The control law will reduce ankle angle asymmetry, when compared with the passive and prescribed conditions.

- **H1.2** The control law will reduce ankle moment asymmetry, when compared with the passive and prescribed conditions.

- **H1.3** The control law will reduce ankle power asymmetry, when compared with the passive and prescribed conditions.

- **H2.1** The control law will reduce the intact peak knee adduction moment, when compared with the passive and prescribed condition.

- **H2.2** The control law will reduce the intact peak hip adduction moment, when compared with the passive and prescribed condition.

1.2.3 Develop Models of the Human-PAFP System

Towards this, various system modeling (i.e., system identification) methods and architectures are implemented and trained to test the feasibility of predicting total prosthetic ankle torque values using only the real-time outputs of wearable sensors. Additionally, system models are useful for analysis and the design of control systems because it is very costly to evaluate different control inputs and parameters repeatedly. A representation of the human-robot system would allow for demonstrative and appreciable interpolation of the process of interest. Additionally, predictive models would allow for the design of model-based predictive PAFP control systems that optimize control commands over a receding horizon. Finally, high-fidelity models can be used to assist in long-term adaptations, reduce actuator requirements, and personalize the controller to the user's unique gait patterns.

1.3 Safety Challenges

An important attribute to powered prosthesis research is user safety. Thus, caution must be taken when iteratively changing the control trajectory. Deploying learning laws with fixed learning gains could results in an unbounded control signal growth. Another scenario is that the magnitude of the error signal in the learning algorithm could decrease at each

iteration, but the magnitudes of each ankle joint's angular trajectory could increase. Such a scenario would also lead to unbounded growth of the control signal and pose a safety issue for the user. If we assume the human can maintain stability (i.e., bound the biological ankle angle reference) for small fluctuations in control inputs, then guaranteeing a bounded control signal would result in a bounded reference signal. To address the need for boundedness of the control signal, an adaptive learning gain is implemented for the learning law.

1.4 Thesis Outline

Chapter 2 provides a literature review, outlines relevant background information, and introduces the reader to key concepts which include ankle mechanics, deviations found in the gait of individuals with below-knee amputation, common secondary medical conditions that these individuals face, gait alterations during load carriage, common PAFP control strategies, an overview of the prototype device's design, and the previously implemented control strategy. Chapter 3 presents the adaptive phase-based symmetry controller and iterative learning method. Subsystems such as the phase state estimator, impedance-based feedback control law, powered drivetrain model, and adaptive iterative learning update law are discussed. Additionally, benchtop testing results are presented in order to validate the proposed control strategy and ensure that it is safe for use. Chapter 4 covers system integration, including the control system architecture, instrumentation, low-level algorithms, real-time software, and embedded hardware. Chapter 5 details the experimental evaluations, including measured outcomes and results. Chapter 6 introduces various model architectures used to characterize and forecast the mechanics of the human-PAFP system. Training methods and model results are presented and implications toward deploying the models for future control methods are discussed. Chapter 7 provides a summary of contributions, key findings, future work, and conclusions.

Chapter 2

BACKGROUND

2.1 Ankle Mechanics

Lower-limb gait analysis involves the measurement of kinematics, kinetics, and energies in order to assess and treat individuals with conditions affecting their ability to walk [4, 40–42]. The gait cycle is defined as the period between the heel strike (HS) of one leg and the next heel strike of the same leg. There are many sub-phases of gait that are defined as well such as foot flat (FF) which occurs after HS and when the whole sole of the foot is in contact with the ground. FF indicates the transition between ankle plantarflexion (PF) to dorsiflexion (DF). DF is defined as moving the foot so that the top of the foot moves towards the anterior shank. PF is the movement of the heel of the foot from the ground or pointing the toes away from the anterior shank.

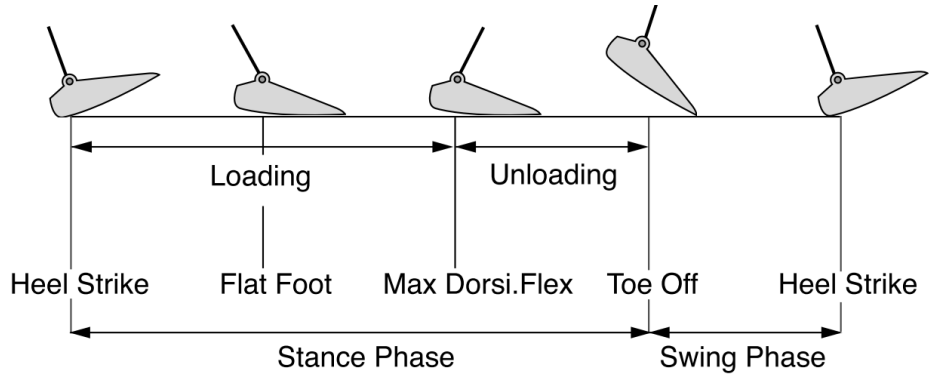


Figure 2.1: Biomechanics of level-ground walking [27]. The gait cycle can be divided into a stance phase and a swing phase.

In this work, ankle displacement and torque promoting DF are considered positive. The neutral ankle position is considered to be at zero-displacement, or when the shank and foot create a right angle. A few other important events of gait are toe-off (TO), where the toe leaves the ground, and maximum dorsiflexion (MDF), which marks the transition from DF

to PF during stance. Stance phase is defined as HS to TO, about 60% of gait. The remaining 40% of gait is defined as swing phase, the period between TO and the next HS. Figure 2.1 summarizes the events and phases of level-ground gait.

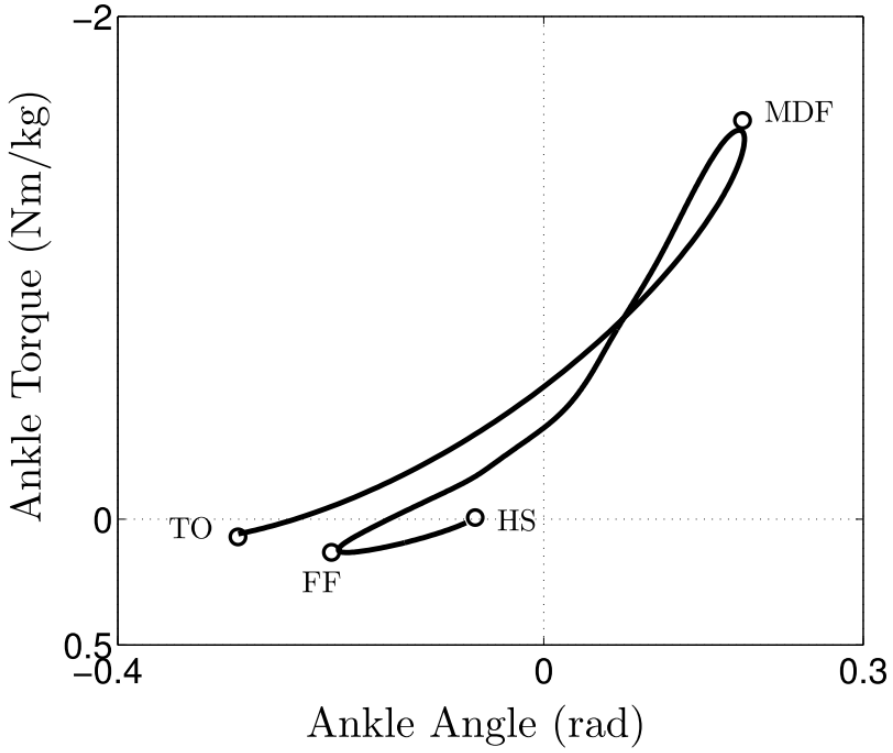


Figure 2.2: Ankle angle-torque profile (ankle torque normalized by body mass versus ankle angle) during stance phase for an individual without amputation walking at their self-selected speed [27]. The loading phase begins with HS and ends with MDF. Unloading begins with MDF and ends with TO. During the unloading period, the ankle joint power is positive and maximal just before TO.

The shape and duration of the gait cycle varies across steps and conditions while also depending on factors such as gait speed and subject mass. Stance phase can be generally split into three sub-phases: controlled plantar flexion (HS to FF), controlled dorsiflexion (FF to MDF), and powered plantar flexion (FF to TO) [43]. Loading occurs between the controlled plantar flexion and controlled dorsiflexion phases while unloading occurs through powered plantar flexion. Plantar flexion functionality during stance performs two purposes:

(i) assisting in controlling forward rotation of the leg in contact with the ground during early stance (controlled plantar flexion) and (ii) providing over 80% of the the mechanical power needed at push-off (powered plantar flexion) [22, 26, 40]. This work focuses mostly on stance phase and the controller's only job during swing phase will be to prevent toe drag and reset to the user-selected neutral position of the ankle.

One of the biggest challenges of designing PAFPs is to mimic and properly time the highly nonlinear behavior of the human ankle (as depicted in Figure 2.2). The ankle and foot consist of complex structures made of skin, muscle, bone, cartilage, and connective tissue. The mechanical properties of these elements are highly nonlinear and depend on several factors such as deformation rate, orientation, and gait speed [44]. Additionally, the mechanical impedance of the ankle can be affected by the individual's reflexes and muscle reactions. This behavior is also very personalized and can vary across different individuals [45]. For the development of PAFPs, it is important to take the intrinsic dynamics of the human ankle-foot complex into account and how below-knee amputation can affect lower-body dynamics.

2.2 Gait of Individuals with Below-Knee Amputation: Deviations from Able-Bodied Gait

The gait of individuals with below-knee amputation has been studied extensively [4–7], and the following common deviations from able-bodied gait have been observed:

- Increased hip and knee flexion of the prosthetic side, and increased hip extension, knee flexion, and ankle dorsiflexion of the intact side [6].
- Increased total knee work of the intact side [16].
- Reduced stance phase on the prosthetic limb [8–10].
- Greater loading of the prosthetic limb when compared to the intact limb [11–14].
- Increased loading rate, load magnitude, impulse, and loading time duration of the intact limb, compared with able-bodied gait [15].
- Prosthetic-side center of mass work reduction from push-off, increased intact-side colli-

sion work, increased overall work for various walking speeds, and increased intact-side impact peak ground reaction forces (GRF) [7].

- Increased metabolic cost [46–48].

Other than increased metabolic cost, the deviations from able-bodied gait listed above are directly related to asymmetries between the two limbs (i.e., loading of the prosthetic limb less than the intact limb). Even though human-in-the-loop optimization (HILO) has been used to successfully tune the parameters of active exoskeleton systems [49–53], metabolic energy-based objective functions have not been successful when implemented into PAFP control schemes. In previous research, four different PAFP controller architectures were tested using a cost function that attempted to minimize the metabolic cost, however, they failed to produce any meaningful changes [54]. One explanation is the inherent differences in user mechanics, feedback, and neural control between people with amputation and able-bodied individuals wearing lower limb exoskeletons. Due to the additional importance of stability and comfort, individuals with amputations may adapt their gait patterns without much change in their metabolic cost despite what the PAFP does.

In a previous study, the researchers theorize that the weaker prosthetic push-off torque leads to overcompensation and additional work to be generated elsewhere in the body [7]. This could potentially explain the increased metabolic energy expenditure and how it is indirectly related to gait asymmetries for people with lower-limb amputations. Therefore, the goal of this research will be to build a PAFP control strategy that minimizes asymmetries between the two limbs. Furthermore, it is important to minimize gait asymmetries between limbs since increased work and magnitude of collisions may increase the risk of secondary conditions in the target populations [1, 17].

2.3 Gait of Individuals with Below-Knee Amputation: Secondary Medical Conditions

Individuals with lower-limb amputations have increased risk of musculoskeletal problems such as osteoarthritis (OA) in the knee and hip, and chronic lower back pain [2]. It is common for these individuals to experience knee pain in the intact limb, and to a lesser degree, hip pain [55]. There is strong evidence that there is an increase in prevelance of knee and hip OA of the intact limb with indivuals with below-knee amputation compared with

the general population [56–58]. However, they are five times less likely to develop knee pain in their amputated limb when compared to the general population [59]. These secondary conditions are theorized to be a result of increased loading of the intact limb and altered body mechanics.

Many studies have shown that increased loading of the lower-limb joints, particularly knee adduction moment (KAM) and hip adduction moment (HAM), are correlated with OA severity and progression in the general population [60, 61]. In one study, researchers observed increased asymmetry in peak KAM in individuals with below-knee amputation compared with the control group [62]. Others have reported that the intact limb of individuals with below-knee amputations have a 65% larger peak KAM and HAM in their intact limb versus their prosthesis side [63]. Additionally, the KAM and HAM for both limbs were larger than the general population. These studies show how altered mechanics as a result of gait asymmetries can lead to knee and hip OA. There is a need for prosthesis systems that provide proper ankle function and PAFPs have the potential to better replicate this behavior, thus providing an advantage over traditional passive prostheses. These devices also have the potential to support a greater range of activities for the user such as load carriage.

2.4 Load Carriage Gait

Many studies in the biomechanics community have focused on load carriage. This mobility task has been shown to alter the dynamics of both able-bodied subjects as well as human-prosthesis systems. Common daily tasks such as carrying grocery bags, work/school bags, equipment, toddlers, and substantial weight gain due to pregnancy can all affect the biomechanics of the individual, with or without an amputation. Some gait alterations as load increases are:

- A decrease in knee and pelvic ranges of motion [64, 65].
- Greater trunk flexion [64, 66].
- Increase in double support [65].
- Decrease in preferred stride length [65].
- Increase in maximum braking force [67].

- Greater increase in average GRF than increase in load alone [68]

Additionally, Lucas-Cuevas et al. reported that able-bodied male subjects have lower whole-body and shank accelerations compared to females, which leads to lower impact forces [69]. This has implications towards carrying infants which is a task performed predominantly by females.

Common injuries associated with prolonged load carriage include foot blisters, stress fractures, back strains, metatarsalgia, and knee pain [70]. Even though significant kinematic and kinetic changes occur due to additional loads, it has been observed from biomechanical analyses that the general population adapts to various loading conditions symmetrically [71, 72]. Conversely, several gait alterations were demonstrated by individuals with transtibial amputations and not by able-bodied subjects. These adaptations include increased DF on the prosthetic side, greater knee flexion of the intact limb due to increased loading, and greater eccentric knee power during weight acceptance [73, 74]. The greater reliance of the intact limb is apparent, yet, most current PAFP designs rely on fixed control parameters that do not take load variations into account. Some research groups have shown that control parameters specific to the loading condition can improve performance and reduce gait asymmetries [75, 76]. However, further development is needed to determine a continuous and adaptive control strategy for load carriage that can minimize gait asymmetries.

2.5 Powered Prosthesis Controllers

The purpose of the PAFP controller is to coordinate the action of the robotic system to best assist the user. Many control strategies have been proposed, however, no single control strategy has been universally accepted by the community [20, 23–25]. This section will briefly summarize some of the most popular control approaches for PAFP systems.

2.5.1 Finite State-Based Control

By far, the most popular PAFP control approaches is using finite-state machines (FSMs) [31, 35, 77–107]. These controllers decompose the periodic gait cycle into distinct phases that, in most cases, are stance flexion, preswing, swing flexion, and swing extension (see Figure 2.3). The FSM control law deploys a set of fixed control parameters that are distinct for each phase of gait. The control parameters help construct unique control laws for each

phase of gait and sensor feedback is used to detect gait events that transition between the finite states. The most popular control law choices for FSM-based controllers are position and impedance control. The latter is expanded on in Section 2.5.3.

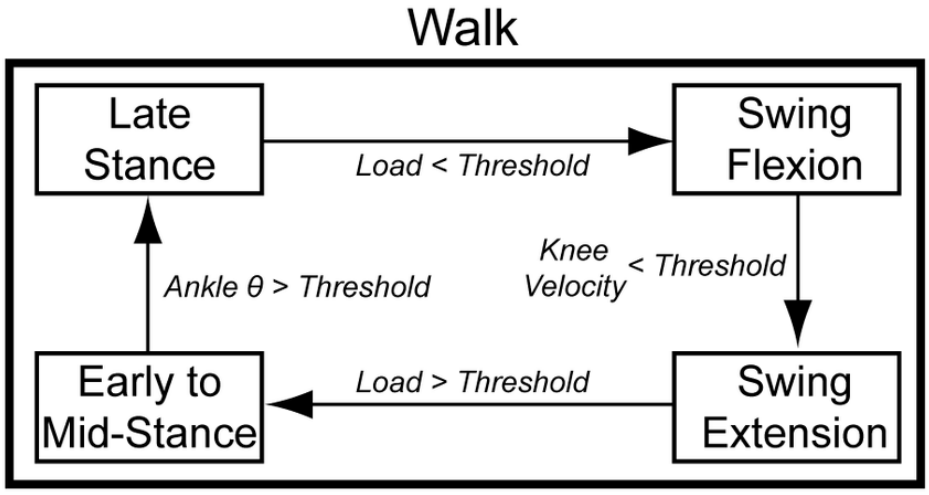


Figure 2.3: Diagram of an example FSM for level ground walking [108].

The gait cycle can be split into more sub-phases than previously stated, however, this can cause the number of tunable control parameters to increase substantially [91]. The number of parameters can also increase as the control law includes different activity modes, as discussed in Section 2.5.2. FSM-based methods typically requires a manual tuning protocol performed by an expert, which is necessary for comfort and performance reasons given the variability in gait patterns across individuals. This has led to studies that have focused on reducing tuning time [108] and using reinforcement learning for automatic tuning [109, 110]. Furthermore, intent recognition algorithms are necessary when using FSMs in order to differentiate and transition between control laws.

2.5.2 Mode-Based Control

Most intent recognition algorithms are based on different types of activities or tasks that the user may perform. Some examples include steady-state walking, running, stair ascent, stair descent, ramp ascent, ramp descent, and turning. Early controllers utilized thresholds

to trigger control law transitions when a different activity mode was detected [80]. However the number of rules and thresholds that must be established for this type of controller can increase very easily. A manual tuning process is only feasible with a handful of predefined tasks and would require re-tuning as the user adapts their gait to the device.

Due to the access to large datasets recently, there has been a shift toward using labeled examples of activity modes to train classifiers using modern machine learning techniques [80, 82, 111, 112]. The inputs to these types of classification models may include sensor data of the device, the environment, and the user. Some popular data-driven classification strategies are Linear Discriminant Analysis [113–115], Support Vector Machines [116], Dynamics Bayesian Networks [117, 118], and Artificial Neural Networks (ANNs) [77, 114]. The main shortcoming of this approach is the large amount of data that needs to be collected for the individual subject performing each activity. In addition, performance has improved when activity mode transitions are included in the training data which can further increase the amount of data necessary for a successful classifier [118].

For these strategies, latency and classification error tolerances for each mode transition need to be addressed further. One challenge these mode-based controllers face is the large variations in movements for every unstructured task. One could continue to include all these cases in the training data but then the task of classifying the correct mode becomes more and more difficult. A classification or delayed transition can lead to poor result ranging from sub-optimal performance to risking the safety of the user. Additionally, human behavior is difficult to categorize into discrete modes based on our ability to adapt, select, and optimize unique movements based on a given real-world scenario. These reasons illustrate why the commercial PAFPs have not adopted mode-based control and when they do, they favor manual mode switching that is robust and unambiguous. An alternative is direct volitional control which is based on myoelectric signals from the user and can grant them the ability to voluntarily switch between device modes [89, 119]. However, this technique is limited by signal processing challenges, cross-talk between nearby muscles, electrode-skin conductivity, skin motion artifacts, repeatability across long time periods, and the potential misalignment of sensors [120]. Due to these limitations, research must be made toward developing continuous and mode-free control methods for portable PAFPs.

2.5.3 Impedance Control

Mode-based control is an example of control at the high level, meaning that the controller perceives the user's intent. As depicted in Figure 2.4, high-level controllers are typically paired with mid and low-level controllers which are responsible for translating the information from the high level to a control signal. One of the most popular low-level control strategies for PAFPs is impedance-based control [29, 32, 103].

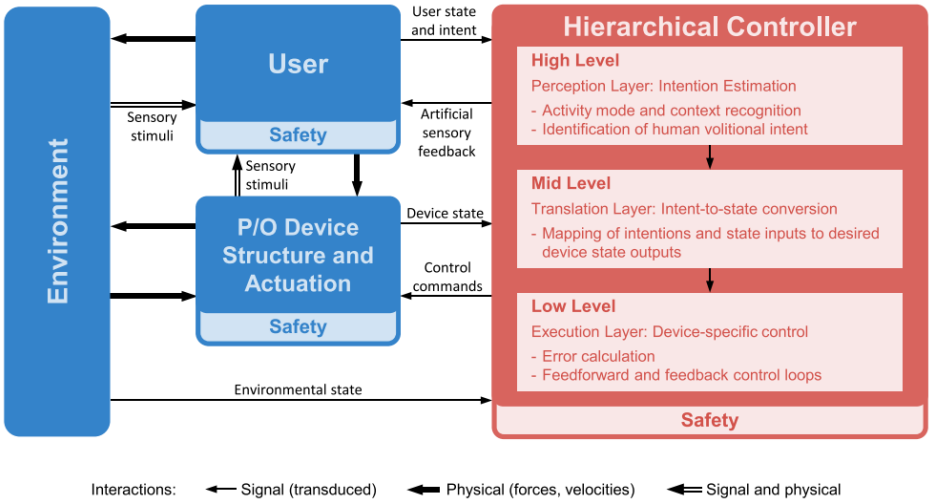


Figure 2.4: Hierarchical Control Architecture adapted from Varol et al. [80]. An estimation of the user's locomotive intent taking place at the high level, translation of the user's intent to a desired device state at the mid level, and a device-specific controller responsible for realizing the desired device state at the low level [24].

In mechanical rotation, impedance is the relationship between angular kinematics and the yielded torque [121]. Impedance control is a flexible method that allows for a wide variety of virtual dynamics to be imposed. PAFP impedance controllers are commonly deployed as a linear virtual spring-damper system [117, 119, 122–124] where the virtual spring stiffness (k), spring equilibrium angle (θ_e), and damping coefficient (b) can be adjusted to modify the desired ankle torque (Equation 2.1).

$$\tau = -k(\theta - \theta_e) - b\dot{\theta} \tag{2.1}$$

A commonly-accepted theory is that the central nervous system controls the limb through impedance control [121, 125], which can explain why human movement can be robust to perturbations despite delays in sensory feedback. One of the reasons why impedance control is so popular among PAFP designers is because it is directly inspired by biomechanics and neuroscience. How impedance varies throughout the gait cycle and across different conditions is still inconclusive and there are ongoing efforts toward characterizing these dynamics [39, 126]. Since the impedance characteristics of the human ankle are not fully explored or widely-accepted, this research will instead focus on the development of a novel strategy that iteratively learns a virtual impedance trajectory for the PAFP controller (see section 3.3). The hypothesis is that there exists some optimal virtual trajectory that when tracked by the low-level PAFP controller, results in the PAFP emulating human ankle behavior (i.e., their impedance properties), thus minimizing gait asymmetries.

2.5.4 Continuous Phase-Based Control

Along with high and low-level controllers, PAFP control architectures typically use mid-level control laws that convert information about the device state to a desired output for the low-level controller to track. Popular mid-level controllers in PAFP design are responsible for determining the gait phase. Controllers that are dependent on the gait phase are referred to as phase-based. Early phase-based control methods programmed a set of actions based on a time delay following an identified gait event [38, 127–131]. This method is easy to implement due to its simplicity, however, it is limited to very repeatable gait cycle periods. Additionally, it cannot adapt to irregularities or perturbations. Echo control presents an alternative that replays position trajectories from the sound limb to the PAFP with some time delay and magnitude scaling [132, 133]. However, this approach assumes temporal symmetry between the two limbs and can be unsafe when undesired/compensatory movements are replayed on the assisted limb. Since then, techniques were developed that present a solution to the limitations of time-based controllers [134–138].

A recent and successful example of time-invariant phase-based control uses virtual constraints to define trajectories as a function of a biomechanical phase variable which increases monotonically across the gait cycle [139–145]. The phase variable is computed from the global thigh motion on the PAFP side using a single inertial measurement unit (IMU) and an extended Kalman filter (EKF). Using the thigh angle measurements, the phase angle

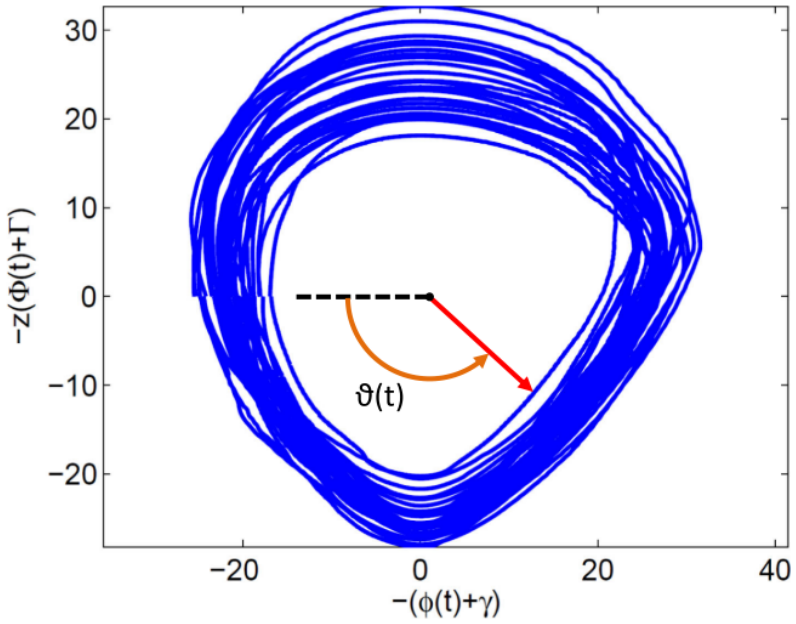


Figure 2.5: Phase plane of the thigh angle $\phi(t)$ vs. its integral $\Phi(t)$ from prosthetic leg experiments conducted by Quintero et al. [139]. The phase plane has been scaled by z and shifted by (γ, Γ) to achieve a circular orbit across the stride, which improves the linearity of the time-invariant phase variable $\vartheta(t)$. Changes in circular orbit diameter are associated with changes in walking speed.

ϑ is computed using the thigh angular position and its integral (refer to Figure 2.5 for a visualization of the thigh phase portrait). Since this method is computed from the residual thigh motion, it gives the user more control over the timing of the PAFP joint dynamics while requiring very few tunable parameters. This research will utilize the phase variable developed by Gregg et al. (see Section 3.4) for computing and identifying virtual trajectories in the phase domain. Note that the virtual constraint controller developed by Gregg et al. will not be used in this research, rather, the biomechanical phase variable will be used to input phase information to a novel iteratively-learned impedance-inspired controller.

2.5.5 Data-Driven System Modeling for Continuous Control

Given the increasing amount of data available in PAFP research, the development of advanced machine and deep learning techniques in the field is now possible [146, 147]. Most of data-driven biomechanics research up to this point has been used to build models that relate environmental, system, and user data to intent recognition [77, 80, 82, 111–114, 114–118] or gait phase estimation [148–152]. While these studies show the potential of using large datasets for assistive device applications, none have addressed using non-categorical training data to build predictive models of human-robot dynamical systems. Regression neural networks have the ability to act as state estimators by learning the underlying relationship between real-time measurements (i.e. joint kinematics from IMUs and translational forces from load cells) and the joint moments computed from motion capture (i.e., inverse dynamics). Additionally, these networks could be used to predict dynamics on a finite-time horizon in order to anticipate the future behavior of the system and act accordingly. These modern modeling techniques learn through examples in the data rather than relying on governing equations based on first principles. If highly-accurate models can be derived from data, many model-based and optimal control methods would possible in the context of PAFP research [153–157].

Some early studies used neural networks and EMG signal inputs to predict ankle dynamics, however their predictions were noisy and less accurate when compared to a muscle model [77, 119]. One explanation is that the EMG signals were aggressively filtered and therefore, there were likely large time delays. Additionally, only simple shallow neural network architectures were used and the hyperparameter optimization was not discussed in detail. The feasibility of using a simple time delay neural network to predict sagittal ankle dynamics was also explored in [158] but similarly, the study did not test more complex architectures capable of learning long-term dependencies [159] or that utilize attention mechanisms [160].

Lately, there has been a big surge in deep learning that leverages expressive neural network architectures, advanced optimizers, parallel computation, and large datasets. Some successful applications include speech recognition [161], computer vision [162], stock market predictions [163], weather forecasting [164], and complex dynamical system modeling [165]. Since the human ankle demonstrates very high-dimensional and nonlinear behavior, many future PAFP control applications can benefit from these promising modeling techniques.

Some studies have successfully used more advanced neural network classes such as recurrent neural networks (RNNs) [166] and long short-term memory networks (LSTMs) [167–169] to generate reference trajectories for complex movements, however, these model architectures have not been used to predict human-PAFP dynamics directly using only inputs from common wearable sensor (e.g., IMUs, encoders, load cells, etc.).

2.6 Previous Work: Prototype Powered Ankle-Foot Prosthesis and Iterative Learning Controller

2.6.1 Mechanical Design

The prototype PAFP designed by Jonathan Realmuto [27, 170, 171] will be used in this research. Figure 2.6 shows the prototype and all its subcomponents. A cam-based spring acts across the ankle joint and provides a nonlinear elastic response which mimics the elastic response of a biological ankle [171]. The spring acts in parallel to the powered drive train which provides active torque and consists of a motorized link acting across the shank and ankle links. Including a passive element parallel to the actuator decreases the power requirements on the active component when compared to a series-elastic actuator [172]. Previous research showed that the addition of the cam-based passive element can reduce the peak actuator torque requirement substantially, by $\sim 74\%$. Moreover, experimental results are presented to demonstrate that the cam-based design can achieve the desired nonlinear response to within 10%.

The drive train consists of the following: a brushless DC motor (Maxon EC-22, 120 W, 18 V), attached to a pin joint on the shank link (pin joint (A)), in series with a planetary gearhead with a transmission ratio of $R_g = 19$ (Maxon GP 22 HP), and followed by a $l = 4$ mm pitch linear ball screw (Thompson NEFF Rolled Ball Screw) attached to a pin joint on the ankle link (pin joint (B)) that acts with a moment arm $r_a = 6$ mm from the ankle joint (ankle pin joint (O)). A compliant bumper (polyester/rubber blend with a durometer hardness rating of 40D), located between the ball screw nut and ball screw housing, protects the transmission from shocks, i.e., it engages near maximum plantarflexion. An energy store and return passive foot (Ossur LP Vari-Flex), attached to the ankle link, protects the drive train from shocks and provides additional energy storing and releasing capability. A detailed analysis of the transmission stages and derivation of the robot model can be found in Section

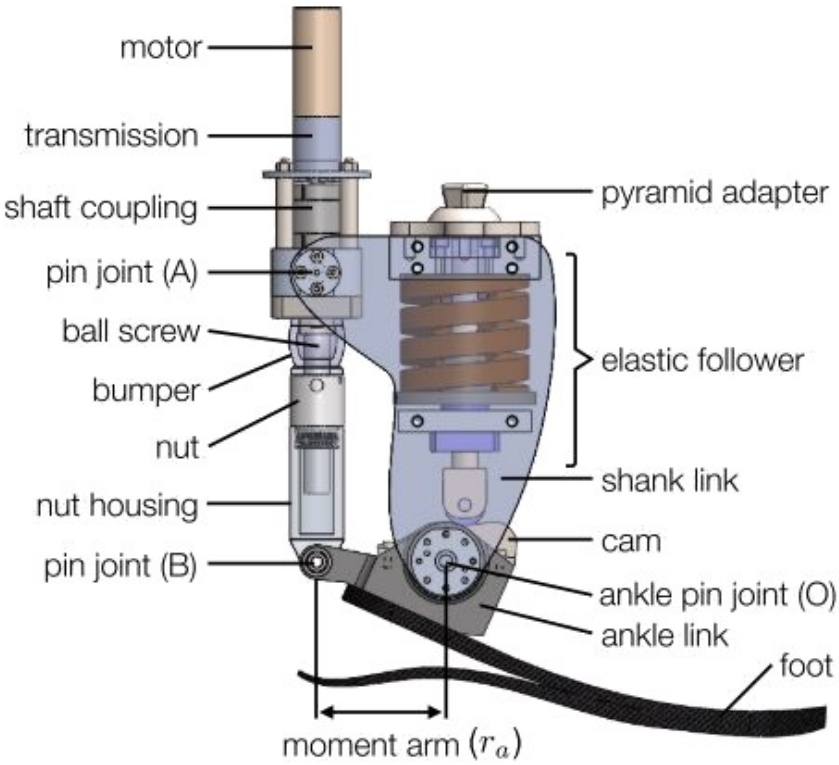


Figure 2.6: Illustrative rendering of the prototype PAFP with major components labeled (note some components are transparent for ease in visualization) [27].

3.5.

2.6.2 Symmetry Controller

The objective of the controller was to coordinate the action of the PAFP while overcoming major limitations in common PAFP approaches. These limitations include: (i) relying on able-bodied trajectories that don't account for personalized altered body mechanics of individuals with amputation, (ii) the need for qualitative tuning of subject-specific control parameters, and (iii) an inability to automatically adapt to the user, especially in the long-term as the user's ability either degrades or improves. Ankle torque asymmetry was proposed as a candidate for HILO. The rational is two fold: (i) as described in section 2.1, asymmetries

are functionally limiting, therefore, in theory, asymmetry may provide a proxy measure of comfort, and (ii), asymmetries could potentially be accurately estimated online, e.g., with low-cost embedded insole sensors. The challenge, as with any in HILO approach, is to avoid divergent responses caused by time-varying human dynamics.

The symmetry control method utilized an adaptive iterative learning control (ILC) algorithm [173]. ILC is a control system design method that is able to achieve high tracking performance by repeatedly executing a task and learning the optimal input from previous attempts of performing the task [174]. This method originated in robotics applications [175–177], however, these approaches converge to the desired output only if the modeling error is small. Accuracy improvements of the model through parameter adaptation with data acquired across iterations was demonstrated in [178, 179]. The previously-developed PAFP controller utilized the adaptive ILC methodology to learn a symmetrical torque signal at the PAFP which matched that of the intact ankle. The controller architecture is illustrated as a block diagram in Figure 2.7.

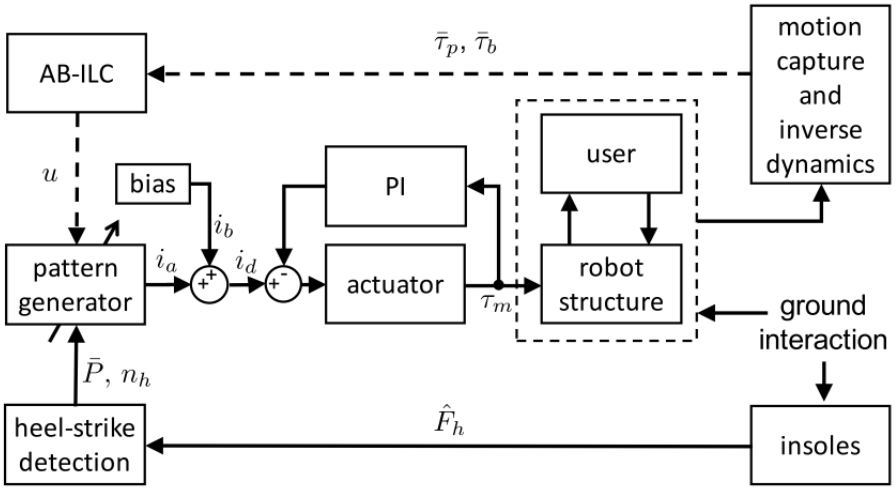


Figure 2.7: Block diagram of the controller architecture developed by Realmuto et al. [170]. Solid lines represent real-time signals and dashed lines represent offline signals.

Note that the desired output is not fixed, rather, it varies as the PAFP torque changes. Offline, after each walking trial (e.g., learning iteration k), inverse dynamics are computed using motion capture (MoCap) data. The mean prosthetic and biological torque signals

across the gait cycle are used to compute the error signal. Subsequently, using an adaptive gain ILC algorithm, a new control signal is then encoded in a pattern generator (i.e., lookup table) within the prototype PAFP's embedded programming.

All lower-limb joints were affected by the symmetry learning controller. Although originally targeting ankle torque asymmetries, the most significant adaptations were observed in the knee and hip. This suggests that lower-limb mechanics are strongly coupled, which is expected since the knee and hip compensate for the loss of ankle function. The intact ankle moment adapted significantly during learning. In particular, as the active ankle torque increased, the intact ankle moment decreased.

One limitation of this study's control method is that it can not positively adapt to large variations in the reference signal. Instead, it backsteps, thus eliminating the potential for more learning. In addition, this controller used a time-based approach by utilizing force sensitive resistor (FSR) signals to identify time-stamps of the most recent HS and estimate the mean step period. The pattern generator was then used to modulate the control input based on the most recent mean step period. This method was observed to not be robust to small perturbations in gait or user adaptations within a gait cycle as indicated by the large variations in the ankle moment. A time-invariant phase-based approach could address this limitation [38, 142]. Moreover, building a model of the human-robot response dynamics, using the data at each iteration, would allow the algorithm to account and adapt to the human contributions [180].

Chapter 3

PERSONALIZED IMPEDANCE-BASED ROBOTIC PROSTHETIC ANKLE CONTROLLER WITH PHASE STATE ESTIMATION AND DATA-DRIVEN VIRTUAL TRAJECTORY PLANNING TO IMPROVE MOBILITY

3.1 Introduction

This chapter introduces the phase-based symmetry learning controller with impedance-inspired feedback. The control algorithm learns a periodic virtual trajectory that can be indexed based on a biomechanical phase variable. The virtual trajectory is learned and adapted by iteratively reducing ankle angle asymmetries of the prosthetic and intact limbs in the frequency domain. Divergence is avoided by monitoring the error, i.e., the difference between the prosthetic and intact ankle angles, and adjusting the frequency-dependent iterative learning gain accordingly. The virtual trajectory is then deployed to a time-invariant and real-time impedance-inspired PAFP feedback control law. With this method, extensive subject parameter tuning is unnecessary, there is no need for able-bodied trajectory data, and both the human and controller jointly adapt.

3.2 Problem Formulation

3.2.1 Concept of Biomechanical Symmetry

For the PAFP system, the active torque generated by the motor τ_a, summed with the passive torque during loading τ_l, results in a total prosthetic torque τ_p.

$$\tau_p(t) = \tau_a(t) + \tau_l(t) \tag{3.1}$$

The goal of the controller is to find the control signal u such that τ_p matches the intact torque τ_b.

$$\tau_p(t) = \tau_b(t) \tag{3.2}$$

where t in Equations 3.1 and 3.2 represent the time-normalized gait cycle which begins with HS (0% gait) and ends with the next HS of the same limb (100% gait). Note that all biomechanical variables should be assumed to be in the sagittal plane unless explicitly stated otherwise. Note that τ_l and τ_b are assumed to be periodic signals affected by τ_a and can vary depending on the individual.

3.2.2 Adaptive Impedance-Inspired Feedback Control

In order to adapt u to achieve the desired active torque, the PAFP ankle kinematics θ_p will be tracked based on a learned virtual kinematic output θ_v. Using an impedance-inspired formula, the error between these two signals is used to compute the required active torque to minimize this error.

$$\tau_a(t) = K\left(\theta_p(t) - \theta_v(t)\right) \tag{3.3}$$

where K is an arbitrary constant control gain. The value of the control gain is tuned once to reduce the time needed for the learned virtual trajectory to converge, however, the gain value ultimately does not matter since the adaptive ILC algorithm (Section 3.3) will learn any necessary changes in magnitude across training iterations. The virtual kinematic trajectory is updated iteratively across walking trial experiments and is a periodic signal based on the gait cycle (see Section 3.2.3 for more details). Note that Equation 3.3 differs from other PAFP impedance-based control methods since the virtual trajectory (i.e., the setpoint) is time-varying as opposed to the gain (i.e., the stiffness parameter). In other methods, the ankle kinematics are subtracted by a fixed equilibrium angle. The reasons for varying the setpoint rather than the stiffness parameter are as follows: (i) the time-varying stiffness parameter is not well-defined in the literature, (ii) personalized time-varying impedance parameter values cannot be computed without an experimental perturbation test, and (iii) an optimal virtual setpoint trajectory can be iteratively learned based on each user's unique dynamics in order to achieve gait symmetry.

Since the virtual kinematic trajectory is periodic based on the gait cycle, a continuous gait percent estimator is required. Most impedance-based control laws involve FSMs that parameterize the gait cycle. However, to avoid the limitations outlined in Section 2.5.1, a single biomechanical variable will be used to parameterize the PAFP dynamics to the gait cycle. This method not only accounts for unexpected (e.g., non-steady) behaviors in gait,

but also makes the control law time-independent. With the correct choice of phase variable ϑ, the appropriate virtual kinematic value will be computed based on the user's location in the gait cycle, even under perturbations.

$$\tau_a(\vartheta) = K\left(\theta_p(\vartheta) - \theta_v(\vartheta)\right) \tag{3.4}$$

Note that by indexing the gait cycle through ϑ, it is possible to control the PAFP in a time-independent manner, meaning that the user can walk at different or inconsistent speeds using one control law.

3.2.3 Iterative Learning Control

The ILC method is used to update the virtual trajectory at each iteration k in the frequency domain:

$$\theta_{v,k+1}(\omega) = \theta_{v,k}(\omega) + \rho_k(\omega)\, G^{-1}(\omega)\, e_k(\omega) \tag{3.5}$$

where ρ_k is the adaptive learning gain, e_k is the error signal, and G is a data-driven transfer function model of the robot dynamics that maps motor current to active torque

$$G(\omega) = \frac{\theta_p(\omega)}{\theta_v(\omega)} \tag{3.6}$$

The error signal e_k in Equation 3.5 represents the measured kinematic asymmetry and is defined as the difference between the time-normalized mean biological ankle angle signal and the mean prosthetic ankle angle signal,

$$e_k(\omega) = \bar{\theta}_{b,k}(\omega) - \bar{\theta}_{p,k}(\omega) \tag{3.7}$$

In practice, the mean ankle kinematic signals at each iteration k are computed over a sufficient experimental 30-second walking trial by taking the mean value, at each discrete percent gait, over all full gait cycles. Note that the frequencies ω that index each variable in Equations 3.5-3.7 depend on the discretization of the time-normalization, and are related by

$$\omega \in \left[0 \quad \omega_0 \quad 2\omega_0 \ldots \quad \frac{N}{2}\omega_0\right], \quad \omega_0 = \frac{2\pi}{T} \tag{3.8}$$

where T is the mean step period, ω_0 is the fundamental frequency of the mean step, and N is the even number of points in the time-normalized gait cycle. Note that the frequencies in Equation 3.8 represent the bin assignment (i.e., harmonic frequencies) of the discrete Fourier transform (DFT). Note that the first element in Equation 3.8 corresponds to the DC gain of the signal and the last element represents the Nyquist frequency. In practice, higher frequencies are removed since they can slow down learning and are more sensitive to small changes in error. In contrast, if not enough harmonics are used, critical information could be excluded from the learner. Note that the biomechanics data used in this study was processed, namely filtered within 6 Hz, to have very little high frequency content. Hence, harmonics higher than 10 won't contribute any new information to the learning algorithm.

Ideally, the ILC update law (Equation 3.5) will be able to achieve high tracking performance and convergence of the error signal by repeatedly executing the walking task. Convergence of the update law can be achieved at each frequency ω provided: (i) the phase error in the model G_m and the choice of learning gain ρ_k are sufficiently small [181] and (ii) the reference signal (e.g., $\bar{\theta}_{b,k}$, in this case) is fixed. Since the control signal u affects the biological kinematics θ_b, the reference signal can vary according to the human-PAFP system dynamics, and thus it may not be possible to guarantee convergence under these criterion. The reference signal $\bar{\theta}_{b,k}$ is driven by the human-response dynamics so the frequency-dependent learning gain ρ_k must be adapted such that the control signal remains bounded. The ILC update law in Equation 3.5 with a fixed learning gain could result in an unbounded actuator response, and poses a safety risk for the user.

3.3 Virtual Trajectory Generation via Adaptive Iterative Learning

To address the problems outlined in Section 3.2.3, this work applies an adaptive ILC technique that learns a virtual impedance trajectory for the PAFP feedback control law in Equation 3.4. A frequency-dependent and adaptive learning gain ρ_k is implemented to this algorithm which assists in convergence toward a viable trajectory that provides powered assistance to the PAFP. Additionally, an inverse modeling method based on measured input-output data is discussed but was ultimately not deployed to the final control system. The reasoning for this decision is discussed further in Section 3.6.3.

3.3.1 Adaptive Learning Gain Technique

The main purpose of the adaptive learning gain that is tuned across iterations k is to avoid potential divergence of the control input u which could risk the safety of the user. In previous studies, iteration conditions and convergence were investigated based on the selection of a frequency-dependent learning gain ρ_k for a human-robot collaborative task [173]. The results showed that the error achieved by the proposed adaptive learning gain was less than that of closed-loop tracking error.

This adaptive learning gain monitors the output tracking error within the frequency domain across training iterations and reduces the magnitude of the gain if the tracking error increases. The tracking error at a training iteration k is defined as:

$$e_k(\omega) = \theta_{b,k}(\omega) - \theta_{p,k}(\omega) \tag{3.9}$$

where $\theta_{b,k}(\omega)$ and $\theta_{p,k}(\omega)$ are the intact and prosthetic ankle angles respectively indexed by the harmonic frequencies ω. If the error e is decreasing at a specific frequency in training iteration k, i.e.,

$$|e_k(\omega)| \leq |e_{k-1}(\omega)| \tag{3.10}$$

then the learning gain at the specified frequency will not change from iteration $k-1$ to k. However, if the error increases at iteration step k for a specific frequency, i.e.,

$$|e_k(\omega)| > |e_{k-1}(\omega)| \tag{3.11}$$

then the iteration gain ρ at frequency ω is scaled down until the error at this frequency decreases below the value at iteration step $k-1$. These conditions are represented by the Boolean variable E_i which is defined as,

$$E_i(\omega) = \begin{cases} 1, & \text{if } \ |e_k(\omega)| > |e_{k-1}(\omega)| + \varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$

where ε is an small optional padding value to assist in convergence. Additionally, a Boolean variable is implemented in the learning gain update based on a specified deadzone in order

to further avoid potential divergence,

$$D(\omega) = \begin{cases} 1, & \text{if } \ |e_k(\omega)| < \frac{1}{100}|\theta_{b,k}(\omega)| \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

The adaptive learning rate is updated utilizing these Boolean variables by the following formula,

$$\rho_k^*(\omega) = (\neg D(\omega))\, E_i(\omega)\, \rho_{k-1}(\omega)\, \alpha \ + (\neg E_i(\omega))\, \rho_{k-1}(\omega)\, \zeta \ + D(\omega)\, \rho_{k-1}(\omega) \tag{3.14}$$

where α and ζ represent the maximum allowable rate of learning decrease and increase respectively. Note that in Equation 3.14, for all $|\omega|$ greater than the maximum harmonic set by the user, ρ_k is set to zero. In this work, the maximum harmonic is set to 10 Hz which is a large enough frequency to describe the user's steady state walking dynamics. Hence, the learned virtual kinematic trajectory from the proposed ILC application is limited to lower frequencies (i.e., frequencies $1 - 10$). Finally, a maximum possible ρ can be set and the following conditions are implemented to assist in avoiding divergence,

$$\rho_k(\omega) = \begin{cases} \rho_{max}\, e^{i\phi_k(\omega)}, & \text{if } \ |\rho_k(\omega)| > \rho_{max} \\ \rho_k^*(\omega), & \text{otherwise} \end{cases} \tag{3.15}$$

where i is the imaginary unit and $\phi_k(\omega)$ is the phase angle in the interval $[-\pi, \pi]$ for $\rho^*(\omega)$ at training iteration k.

3.3.2 Inverse Model Generation via Averaging

One of the main advantages of ILC methods is that future output tracking information can be utilized through noncausality [182–184], which is particularly useful when trying to control non-minimum phase systems [182–187]. A non-minimum phase system is any system that, when undergoing an impulse or step response, "goes in the wrong direction" first before moving toward and finally reaching its desired steady state value. Some examples of these types of systems include:

- An aircraft (VTOL, rocket, UAV, etc.) where in order to ascend, the elevator is moved

downward which gives the wings a more positive angle of attack. This causes the center of mass of the aircraft to move downward before ascending to the desire altitude.

- Parallel parking a car where the front of the car moves into the the middle of the road before backing into the parking spot.

- Flexible structures commonly used in fields such as soft robotics, piezoelectric sensors, or pneumatic actuator design. For these systems, the structure experiences a delay where it flexes in the wrong direction due to the material's elastic response before reaching a steady state value.

- When jumping vertically, people lower their center of gravity to store energy in their tendons before releasing this energy and launching into their jump.

All these examples illustrate challenges in control and stability since the system dynamics introduce uncertainties and time delays where undesired behavior occurs before the system can achieve its desired position. However a noncausal ILC application such as inversion-based iterative learning control (IIC) can implicitly account for these behaviors, thereby assisting in convergence of the feedforward input. However, modeling dynamics is not always straightforward and IIC performance is directly related to the quality of the process model.

Kim et al. developed an ILC strategy where the inverse model is updated at each training iteration k by using the measured input-output data in the frequency domain [188]. This iterative update virtually removes the modeling error when the noise effect can be ignored and no separate dynamics modeling process is needed. In many applications such as in human-robot systems, the output noise level can be relatively easy to quantify in contrast to the modeling process of the system dynamics. In addition, the variation in output noise level is small. By applying this technique on the output tracking of a piezotube scanner, Kim et al. showed that the tracking error dramatically reduced when generating an inverse model via averaging and including it in the ILC update law.

In this work, inverse model generation via averaging is implemented using the input-output PAFP system data,

$$\begin{aligned} u_k(\omega) &= \theta_{v,k} \\ y_k(\omega) &= \theta_{p,k} \end{aligned} \tag{3.16}$$

where u is the learned virtual trajectory (system input), y is the PAFP ankle angle response (system output), and G_m^{-1} is the inverse transfer function model. These variables are collected across each harmonic frequency ω at each training iteration k. At the initial training iteration ($k = 1$), the inverse transfer function model is computed as the following,

$$G_k^{-1}(\omega) = \frac{u_k(\omega)}{y_k(\omega)} \tag{3.17}$$

On every future training iteration ($k > 1$), recently-acquired input-output data and the average inverse transfer function model across all previous iterations are used to update the inverse transfer function model

$$\begin{aligned} G_k^*(\omega) &= \frac{u_k(\omega) \; + \; (k-1)\, y_k(\omega)\, G_{k-1}^{-1}(\omega)}{k\, y_k(\omega)} \\ G_k^{-1}(\omega) &= |G_k^*(\omega)|\, e^{i \angle G_k^*(\omega)} \end{aligned} \tag{3.18}$$

where G_k^* is the updated inverse transfer function model with the recently-acquired input-output data included. In order to simplify multiplication operations in the learning algorithm, the inverse model is converted to exponential identities using Euler's formula. Thus, G_k^{-1} is the same updated model but in exponential form, which uses the magnitude and phase angle attributes of G_k^*.

The use of this input-output modeling via averaging technique is meant to overcome common issues associated with traditional dynamical system modeling such as the effects of disturbances, uncertainty, signal noise, and hysteresis. Additionally, not only is the behavior of a human ankle very nonlinear and complex, but the prosthetic foot is elastic and likely exhibits non-minimum phase behavior. Therefore, this method has the potential to assist in the convergence of the ILC update law by learning and accounting for these dynamics over time.

3.3.3 ILC Update Law

The iterative update law, which utilizes the adaptive learning gain and average inverse transfer function model outlined above, computes the new virtual trajectory $\theta_{v,k+1}$, as:

$$\theta_{v,k+1}(\omega) = \theta_{v,k}(\omega) \; + \; \rho_k(\omega)\, G_k^{-1}(\omega)\, e_k(\omega) \tag{3.19}$$

The virtual trajectory is periodic along the gait cycle and is used within the impedance-inspired feedback control law (Equation 3.4) to generate an active ankle torque reference for the low-level control law (see Section 3.5 for more details). Note that in practice, the control signal is eliminated during the swing phase of gait. Therefore, the torque reference signal is set to zero during swing phase:

$$\tau_{ref}(\vartheta) = \begin{cases} 0, & \text{if (gait} > 70\% \text{ \& HSD State} = 1) \\ K(\theta_p(\vartheta) - \theta_v(\vartheta)), & \text{otherwise} \end{cases} \tag{3.20}$$

where ϑ is a biomechanical phase variable that is measured in real time to continuously estimate where the subject is along their gait cycle (Section 3.4) and the HSD State variable is the output of a FSM that determines if the foot is in contact with the ground (Section 4.3).

3.4 Biomechanical Phase Variable

The symmetry learning controller used in this work requires a phase variable to robustly and continuously represent the PAFP dynamics throughout the entire gait cycle. This variable must have a monotonic trajectory during steady state walking and needs to be computed from an unactuated state of the system. It has been shown in the neuroscience literature that the muscles at the hip joint are major contributors to the synchronization of all lower-limb joints [189]. Using the measured thigh angle with respect to the vertical gravity vector, Gregg et al. investigated a number of phase variable candidates and evaluated their robustness to phase-shifting perturbations [190]. This study revealed a phase variable computed from the global thigh angle and its integral, denoted as PHI in the manuscript, was the most practical for control applications. Even though a phase variable based on the global thigh angle and its velocity better parameterized the gait cycle for non-steady gait, the additional noise from numerical differentiation may cause stability issues. Filtering the noise could also cause a system delays, which takes away control from the user.

PHI avoids numerical and sensor measurement noise while only being slightly less representative of total lower-limb dynamics. Interestingly, this variable actually performed the best at representing ankle kinematics. For these reasons, as well as the variable's linearity and reliability, PHI will be used in this research as a time-invariant gait estimator. Note that

unlike some of the studies conducted by Gregg et al., this research focuses only on steady gait. However, this method can be implemented and modified to be applied to various non-steady tasks. The continuous and normalized phase angle ϑ is calculated as:

$$\begin{aligned} \vartheta[n] &= \frac{\text{atan2}(z\,Y[n],\ X[n]) - \vartheta^{+}}{\vartheta^{-} - \vartheta^{+}} \\ X[n] &= -\left(\theta_h[n] - \gamma\right) \\ Y[n] &= X[n]\,dt\ +\ \ Y[n-1] \end{aligned} \tag{3.21}$$

with $n \in [0, N]$ representing the instantaneous sample (i.e., timestep) of the current gait cycle ($n = 0$ at HS, $n = N$ at next HS). Note that N is not a fixed value and can vary across strides. The atan2 function is the arctangent for angles in any of the four quadrants of the x-y phase plane. Variables ϑ^{+} and ϑ^{-} normalize the phase variable trajectory across the gait period, i.e. $\vartheta \in [0, 1]$. The "+" and "–" indicate the phase variable starting value of the prosthetic-side stance period and ending value of the prosthetic-side swing period respectively. The phase variable is also saturated at an upper limit of 1 in order to avoid wraparound effects (i.e., discontinuities) caused by variations in thigh angle range of motion across gait cycles. Once the phase variable reaches a value of 1, it is held at this value until the next HS occurs which then resets the value to 0.

The parameter z is a scale factor that increases the monotonicity of the phase variable while γ is a phase shift value that centers the thigh orbit around the origin of the phase portrait. These adaptive parameters are computed at the most recent HS event m as:

$$\begin{aligned} z_m &= \frac{|X_{max} - X_{min}|}{|Y_{max} - Y_{min}|} \\ \gamma_m &= \frac{1}{N}\sum_{n=0}^{N} \theta_h[n] \end{aligned} \tag{3.22}$$

where X_{max}, X_{min}, Y_{max}, and Y_{min} represent the maximum and minimum values for X and Y within the previous gait cycle. Parameter γ represents the sum of global thigh angle measurements θ_h across only the previous gait cycle. This phase shift parameter is reset every gait cycle to prevent the accumulation of drift due to variations in thigh kinematics. Similarly, Y is reset to zero at every HS. A moving average filter with a window size of five is then used to update z and γ (Equation 3.23). This improves the robustness of the calculator

under variations in thigh range of motion and walking speed.

$$\begin{aligned} z &= \frac{1}{5}(z_m + z_{m-1} + z_{m-2} + z_{m-3} + z_{m-4}) \\ \gamma &= \frac{1}{5}(\gamma_m + \gamma_{m-1} + \gamma_{m-2} + \gamma_{m-13} + \gamma_{m-4}) \end{aligned} \tag{3.23}$$

Note that this moving average filter is applied to parameters updated every gait cycle. Therefore, it does not result in system delays related to the real-time signals (i.e., $X[n]$, $Y[n]$, and $\vartheta[n]$). Figure 2.5 shows the normalized and shifted thigh orbit in the phase portrait over several strides. Larger orbit diameters represent higher walking speeds. The adaptability of this algorithm is useful as it can be personalized to each user and can be implemented across various periodic locomotion tasks. For this research, phase variable ϑ will be used to estimate the phase of the gait cycle, which in turn selects, the appropriate value for θ_v (which is a function of gait phase) via a lookup table. Additionally, the gait phase estimation informs FSM conditions that switch the controller from stance to swing states at the appropriate time (Equation 3.20).

3.5 Inverse Actuator Model

The low-level PAFP controller takes the desired active torque τ_{ref} from Equation 3.20 and uses it to compute the desired motor current i_m. An inverse actuator model is derived in this section which transforms active ankle torque values in the sagittal plane to a motor current command based on the robot's physical configuration, geometry, and dynamics. To assist in illustrating this first principles modeling task, a simplified schematic of the PAFP was created and is shown in Figure 3.1.

The motor is placed parallel to the shank and as it rotates, the ball screw placed below the motor moves linearly. This creates a force on a pin placed where the Achilles tendon of the PAFP would be. This force causes a moment on the ankle via a link that connects the artificial Achilles tendon to the ankle joint. This moment is the active torque τ_a on the ankle provided by the motor. The active torque τ_a, generated by the powered drivetrain, can be estimated as

$$\tau_a = rF_s \tag{3.24}$$

where F_s is resultant linear force of the ball screw acting at joint B and r is the moment

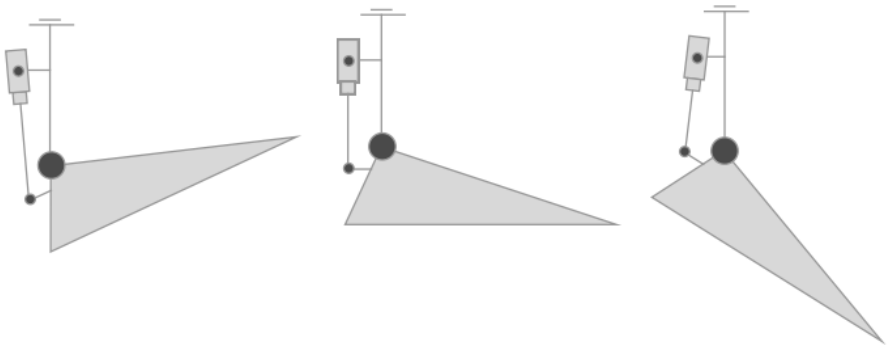


Figure 3.1: Simplified schematic of the prototype PAFP. This rendering outlines the PAFP and its main components going through DF (left) and PF (right) movements. Additionally, the neutral position (zero rotation) of the PAFP is shown (center). Adapted from notes (Jonathan Realmuto).

arm. Note that Equation 3.24 does not include the geometric nonlinearities caused by the drivetrain linkage mechanism. To better illustrates the geometry between the PAFP system linkages, a simplified diagram is presented in Figure 3.2. In addition, Table 3.1 summarizes the prototype PAFP's mechanical system parameters used in this study.

Table 3.1: PAFP System & Design Parameters

Name	Symbol	Unit	Value
Gearhead Reduction	R_g	-	19.2
Gearhead Efficiency	η_g	-	0.70
Torque Constant	k_τ	Nm/A	0.00775
Ball Screw Efficiency	η_s	-	0.90
Ball Screw Length	a	m	0.134
Ankle Link Moment Arm Length	d	m	0.06
Ankle Angle Offset	θ_0	rad	-0.0785
Screw Pitch	l	m	0.004

The schematic in Figure 3.2 shows that lengths b and d are fixed due to the constant moment arms from pin joint B to pin joint O and the rigid shank link. Length a, angle β, angle α, and angle ϕ are all functions of the ankle joint angle θ. Using the positioning of the

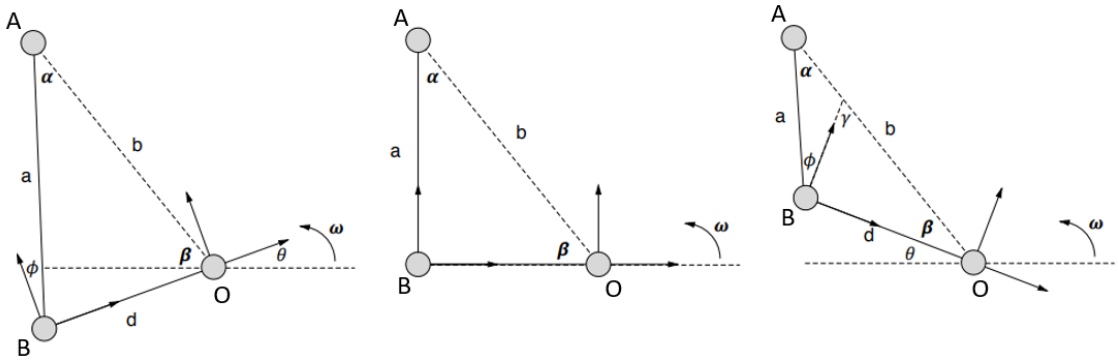


Figure 3.2: The prototype PAFP represented as two linkages connecting the drivetrain to the ankle joint. Pin joint A is where the ball screw connects to the shank link, pin joint B is where the ball screw nut housing connects to the ankle link, and pin joint O is where the ankle joint is connected to the drivetrain. The lowercase letters represent the lengths between the three pin joints and the Greek letters represent the angles. The PAFP is shown positively rotating (left), not rotating (middle), and negatively rotating (right). The variable ω represents the direction of positive angular velocity about the ankle joint. Adapted from notes (Jonathan Realmuto).

components, these variables can be defined as the following:

$$\begin{aligned} \beta &= \arccos\left(\frac{d}{b}\right) + \theta - \theta_0 \\ a &= \sqrt{b^2 + d^2 - 2bd\cos\left(\beta + \theta - \theta_0\right)} \\ \alpha &= \arcsin\left(\frac{d\sin\left(\beta\right)}{a}\right) \\ \phi &= \frac{\pi}{2} - \beta - \alpha \end{aligned} \tag{3.25}$$

where θ_0 is an angular offset to ensure that at maximum torque, the ball screw a and ankle link moment arm d are perpendicular. Using these definitions, the moment arm reduction from pin joint B and pin joint O is

$$\frac{\tau_a}{F_s} = d\cos\left(\phi\right) \tag{3.26}$$

Next, the mechanical advantage of using the ball screw is calculated. The lead is defined

as the linear distance the screw travels in one revolution and determines the mechanical advantage (i.e., smaller leads result in higher mechanical advantages). Using the law of conservation of energy and the principles of screw theory,

$$\begin{aligned} W_{in} &= W_{out} \\ 2\pi\tau_g &= lF_s \end{aligned} \tag{3.27}$$

the mechanical advantage of the ball screw when considering screw lead is defined as

$$\frac{\tau_g}{F_s} = \frac{l}{2\pi} \tag{3.28}$$

where W_{in} is the mechanical work applied from the motor to the ball screw, W_{out} is the mechanical work applied by the ball screw, τ_g is the torque outputted by the transmission, and l is the lead for the screw. Similarly, the frictional losses of the ball screw are considered using the conservation of energy,

$$\begin{aligned} W_{in} &= W_{out} + W_{fric} \\ \eta_s W_{in} &= W_{out} \\ \eta_s 2\pi\tau_g &= lF_s \end{aligned} \tag{3.29}$$

the mechanical advantage of the ball screw is redefined to include frictional losses as

$$\frac{\tau_g}{F_s} = \frac{l}{2\pi\eta_s} \tag{3.30}$$

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

$$\frac{\tau_g}{\tau_m} = R_g\eta_g \tag{3.31}$$

where R_g is the planetary gearhead reduction and η_g is the gearhead efficiency. Using Equations 3.26, 3.30, and 3.31, the total mechanical advantage from the motor to the ankle joint is defined as

$$\frac{\tau_m}{\tau_g}\frac{\tau_g}{F_s}\frac{F_s}{\tau_a} = \frac{1}{R_g\eta_g}\frac{l}{2\pi\eta_s}\frac{1}{d\cos(\phi)} \tag{3.32}$$

Simplifying 3.32 and inverting results in

$$R_\tau = \frac{\tau_a}{\tau_m} = \frac{d\cos(\phi)2\pi\eta_s\eta_g R_g}{l} \tag{3.33}$$

where R_τ is the drivetrain ratio (effective transmission) that relates the applied motor torque to the resultant active ankle torque component. The gain from the motor current command i_a to the active torque can be expressed as:

$$\tau_a = R_\tau k_\tau i_a \tag{3.34}$$

where k_τ is the rated motor torque constant. Equation 3.34 can be rearranged as

$$i_a = \frac{\tau_a}{R_\tau k_\tau} \tag{3.35}$$

Note that the active current i_a and the bias current i_b (see Section 4.8) combine to define the desired current i_d as,

$$i_d = i_a + i_b \tag{3.36}$$

Substituting the definition of i_a from Equation 3.35 into Equation 3.36 and replacing the active torque definition τ_a with the desired active ankle torque τ_{ref}, the low-level control law is defined as

$$i_d = \frac{\tau_{ref}}{R_\tau k_\tau} + i_b \tag{3.37}$$

Equation 3.37 determines the correct current command to the motor which can track the desired active torque provided by Equation 3.20. Even though this inverse actuator model is very comprehensive, it is possible that it is not perfect and may include some uncertainties due to additional loads, material deformities, or nonlinearities. However, the iterative learning procedure outlined in Section 3.3 is meant to implicitly learn and adapt to any underlying physics not accounted for in the low-level control law over time.

3.6 Controller Implementation & Discussion

Preliminary validation of the PAFP controller was conducted and details are discussed in this section. Even though the main outcome of this research is to test a novel PAFP control

scheme on human subjects with below-knee amputation, prior validation of the device's control algorithms is still necessary. A guarantee of reproducibility and proper actuator timings based on the gait cycle can ensure user safety before human subject experiments. Additionally, the tests presented in this section were essential in establishing data collection protocols, debugging electronic hardware, and qualitatively assessing the behavior of various control signals.

3.6.1 Validation of the Real-Time Phase Variable Calculator

The phase variable algorithm is essential to mapping the state of the human-robot system to the desired active ankle torque reference and subsequently, the motor current command. The purpose of this algorithm is to output a strictly monotonic and increasing phase variable that represents the progression from 0.1% to 100% of the gait cycle using a single thigh-mounted IMU. A preliminary experiment involving an able-bodied volunteer was conducted to investigate how accurately the phase variable parameterized gait progression in real-time. The setup consisted of the able-bodied volunteer walking on an instrumented treadmill with the IMU sensor mounted within the sagittal plane of the thigh using Velcro straps (see Figure 3.3).

The measured IMU and GRF signals were read on a myRIO real-time embedded device and the phase variable algorithm outputs were observed within the embedded computing user interface (LabVIEW VI). The protocol involved the volunteer initially walking at 1 meter per second (mps) while the phase calculator underwent an initial principal components analysis to decouple thigh motion into the anatomical planes (see Section 4.6). IMU sensor data, heel strike detection (HSD) data from the treadmill's force plates, and real-time phase variable values were then sampled at 500 Hz.

The normalized phase variable with scale and shift parameters resulted in a sufficiently monotonic signal that can accurately map to the user's location along the gait cycle (see Figure 3.4). However, one observation during these preliminary tests is that, at times, the IMU sensor outputs were sensitive to residual motion independent of thigh kinematics. For instance, it is evident that the IMU shifted independent of the thigh due to the impact of some HS events. This caused oscillations in the thigh angle signal which results in the phase variable having undesirable non-steady slopes or, less commonly, small non-monotonic behavior of the phase variable signal (see Figure 3.5).

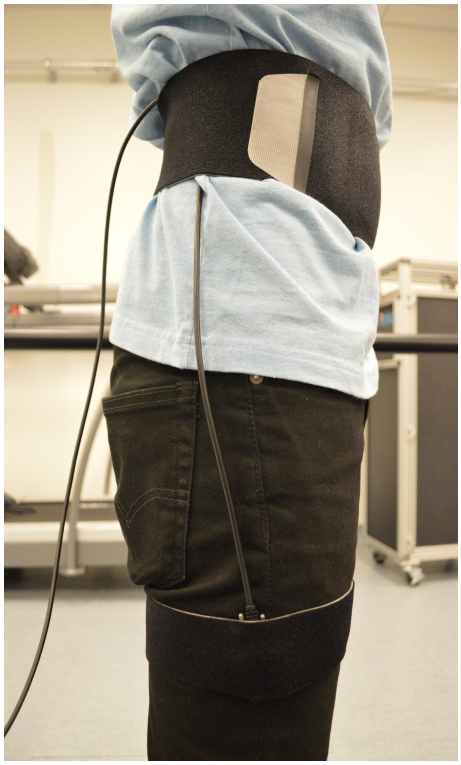


Figure 3.3: Photo of the IMU sensor placed on the thigh of a control test subject during the first preliminary phase variable calculator validation experiment.

A second validation experiment was conducted which utilized self-adhesive wrap (Coban) and had the user wear spandex to ensure that the IMU was fixed to the thigh (see Figure 3.6). As with the first experiment, sensor data were sampled at 500 Hz. Additionally, the subject walked at speeds of 0.8, 1.0, 1.2, and 1.4 mps. Figure 3.7 shows that the real-time phase variable calculator output displays linear behavior and stays monotonic within a tolerance of 0.01 across the 427 collected gait cycles.

The results indicate that the experimental setup and phase variable calculator successfully provide a robust estimate the gait cycle's phase in real time. In addition, the algorithm adapts and performs well across various walking speeds.

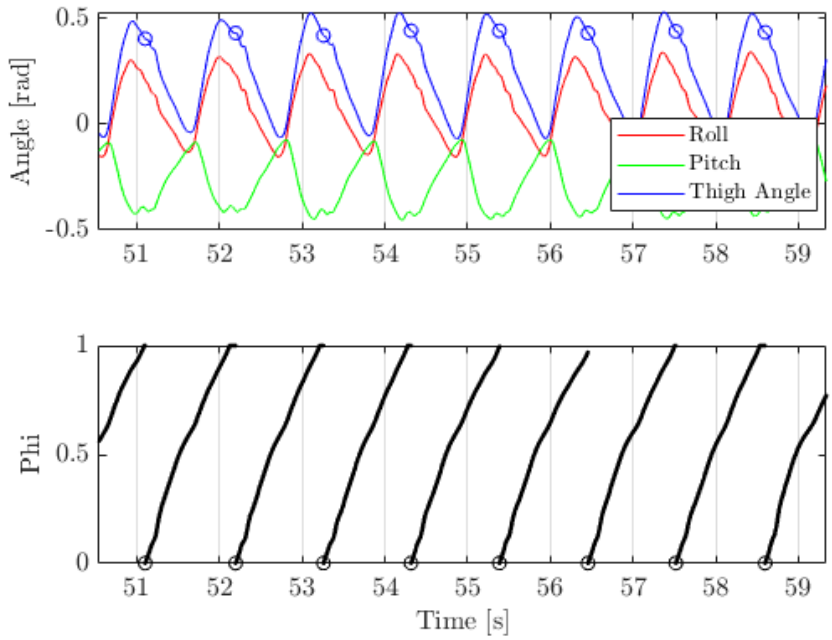


Figure 3.4: Sampled IMU data and real-time phase variable calculator results across part of a validation trial. The top plot shows the sampled IMU roll and pitch angles as well as the thigh angle estimate computed by the PCA coordinate transformation. The bottom plot shows the real-time phase variable output (Phi) which exhibited monotonically increasing behavior. In both plots, HS events are represented by circular markers.

3.6.2 System Integration Benchtop Test

Once the phase variable calculator was validated, benchtop tests were conducted to assess the ankle kinematics and motor current command signals produced by the control algorithms. The kinematic tracking accuracy and boundedness of the two signals were both analyzed. The purpose of this benchtop test was to demonstrate the latency and safety of the low-level control software prior to experiments with human subjects.

The prototype PAFP was fixed to a table-mounted frame where the device was not in contact with the ground (see Figure 3.8). An arbitrary trajectory was used as the virtual setpoint in the impedance-inspired feedback control law (Equation 3.20). This trajectory

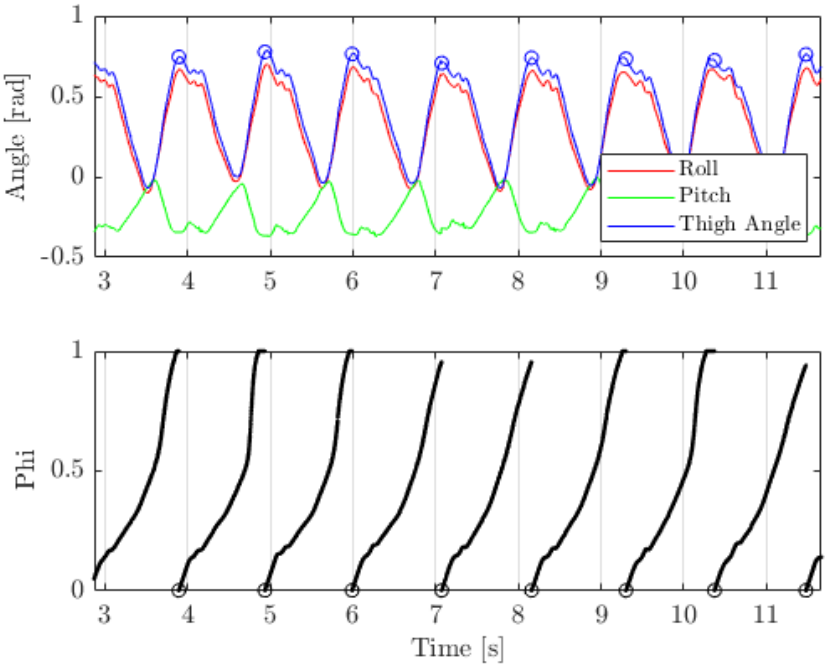


Figure 3.5: An example of a validation trial where the phase variable output was non-steady and non-monotonic. The top plot shows the sampled IMU roll and pitch angles as well as the computed thigh angle estimate. The bottom plot shows the resulting real-time phase variable output (Phi). HS events are displayed as circular markers.

was designed to dorsiflex during early stance, plantarflex during late stance, and be held constant during swing. An able-bodied control test subject was used to generate GRF and IMU signals in order to simulate cyclical gait for the virtual trajectory lookup table (see Section 4.7) and actuate the PAFP. The subject wore the thigh-mounted IMU and ambulated on an instrumented treadmill at their self-selected speed. Tracking of the virtual trajectory and behavior of the control input were both observed using LabVIEW VI front panel visuals. The researcher then tuned the control gain K of the impedance-inspired model in order to reduce the tracking error. Note that the control gain was manually adjusted in this experiment since the ILC update law was not deployed. In an actual experiment, the magnitude of the desired active torque will be scaled and adapted by the ILC update law based on the user's gait symmetry.

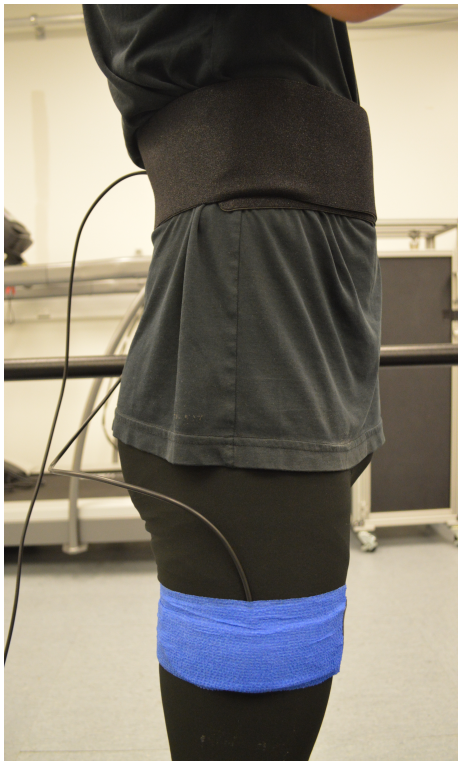


Figure 3.6: Photo of the IMU sensor placed on the thigh of a control test subject during the second preliminary phase variable calculator validation experiment. In addition to the Velcro straps, this setup also utilized Coban wrap and had the user wear spandex shorts.

Using inputs from the control subject and after tuning the control gain, the controller was able to track the desired kinematic trajectory with an average root-mean-square-error (RMSE) of 0.015 radians ($<$ 1 degree). Additionally, there was an average lag of 0.033 seconds in the response of the PAFP. The tracking results are shown in Figure 3.9 and it is worth noting that the majority of the tracking error occurs during the simulated swing phase (i.e., where the set point is set to a constant until the following HS event). These errors were likely caused by the PAFP oscillating as a result of the sharp step input between stance and swing phases. Additionally, the inertial properties of the PAFP and very high stiffness of the spring in the plantarflexion direction could have been factors. In practice, these oscillations would be dampened by the mass of the user and there would be little effect. The swing phase error could have also been reduced further by finely adjusting the bias current. Since the

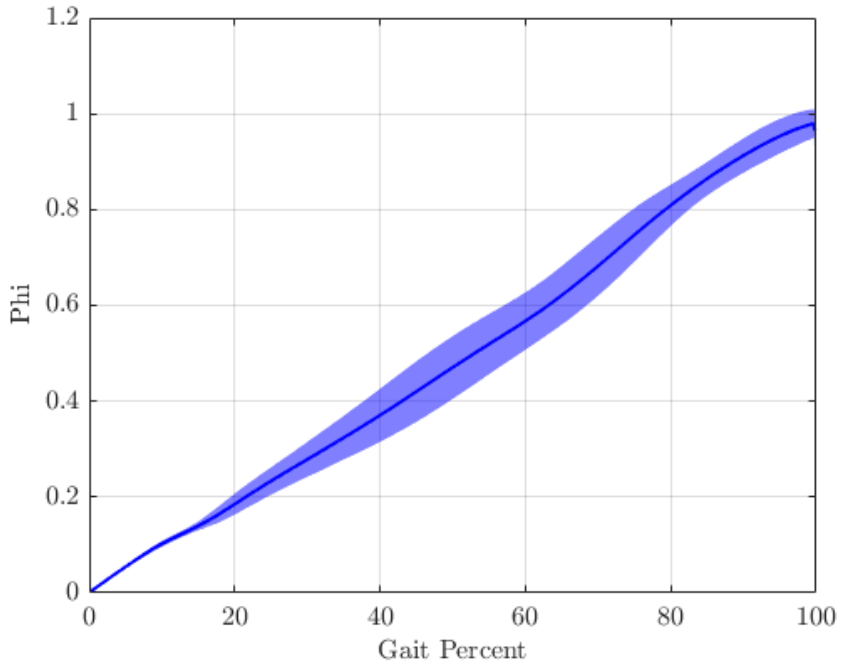


Figure 3.7: Collected outputs of the real-time phase variable calculator for the second validation experiment. The outputs are time-normalized to the gait cycle. The solid line indicates the average profile across 4 walking speeds and 427 total strides, while the shaded areas indicate ±1 standard deviation.

active current is automatically set to zero during swing, the bias current is the only signal responsible for achieving a set point during this period. During human subject experiments, this bias current is determined during the acclimation period to prevent toe drag during swing. When analyzing the performance of the low-level software during stance phase only, the feedback control law tracked very well.

The benchtop test demonstrated that the embedded hardware and control law were able to run reliably in real time. The PAFP adapted to the individual's state and the controller was able to achieve satisfactory tracking performance during stance phase, however, the fixed-gain impedance formula needed to be manually tuned for this experiment for improved tracking performance. Implementing the ILC update law will address some of the shortcomings seen in this benchtop test. Namely, the ILC algorithm will adapt and learn based on

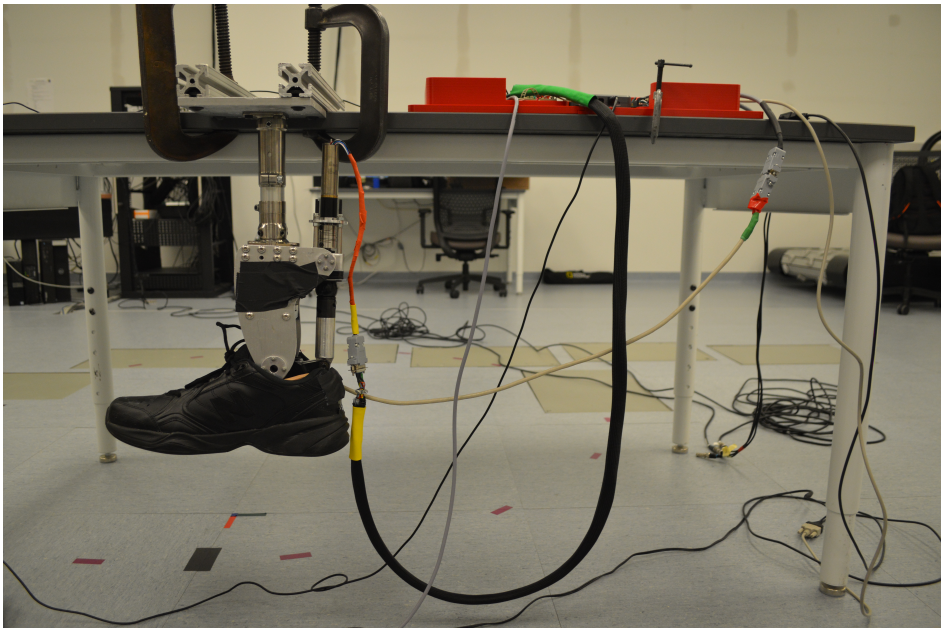


Figure 3.8: Photo of the benchtop test setup. The embedded system used wearable sensor data from a healthy control subject to actuate the table-mounted PAFP in order to track an arbitrary phase-indexed kinematic trajectory.

the user's motion capture data, which removes the need to manually tune the control gain past the first ILC iteration. In addition, the adaptive learning gain of the ILC algorithm also has advantages in terms of overall control system stability, e.g., the error growth must be bounded since the learning weights always decrease with increasing error [173]. Additional benchtop tests were performed to determine the ILC algorithm's capabilities and are discussed next.

3.6.3 ILC Benchtop Tests

A number of benchtop tests were conducted to assess the performance of the impedance-inspired control law with the ILC updates. All tests used preliminary walking data from the experimental subject (Section 5.2.2) which included left and right GRF signals as well as thigh-mounted IMU outputs (roll, pitch, and yaw Euler angles). All data were averaged over the gait cycle and used as simulated signals to drive the behavior of the table-mounted

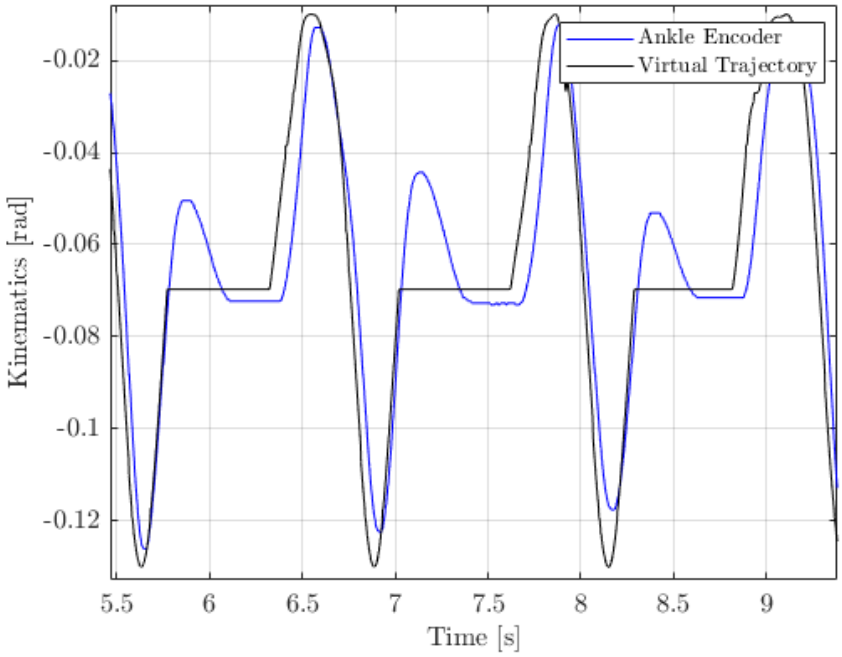


Figure 3.9: Tracking performance of the controller during the benchtop test based on the phase variable, manually tuned impedance-inspired formula, and inverse actuator model. Note that the ILC algorithm was not tested for this experiment.

PAFP, similar to the system integration benchtop tests discussed previously. The goal of the ILC benchtop tests was to implement the ILC method to control the table-mounted PAFP in a way that mimics the simulated biological ankle angles while walking. Note that $\theta_{b,k}$ in Equation 3.9 was a fixed trajectory. In practice, this signal would change across iterations as the control signal is learned and the user adapts. The bias current was set to 2 amps and the fixed impedance gain K found in Equation 3.20 was preset to 400. Results from the first ILC benchtop test are displayed in Figure 3.10.

The results from the first test show that the ILC update law was able to modify the virtual trajectory, θ_v, over training iterations k to slowly reduce the error between the prosthetic ankle angles and the simulated biological ankle angles. One interesting discovery was that the terminal stance phase errors (around $30-60\%$ gait) reduced over iterations $1-5$ but started increasing after iteration 6. The magnitude of the virtual trajectory θ_v was lowered

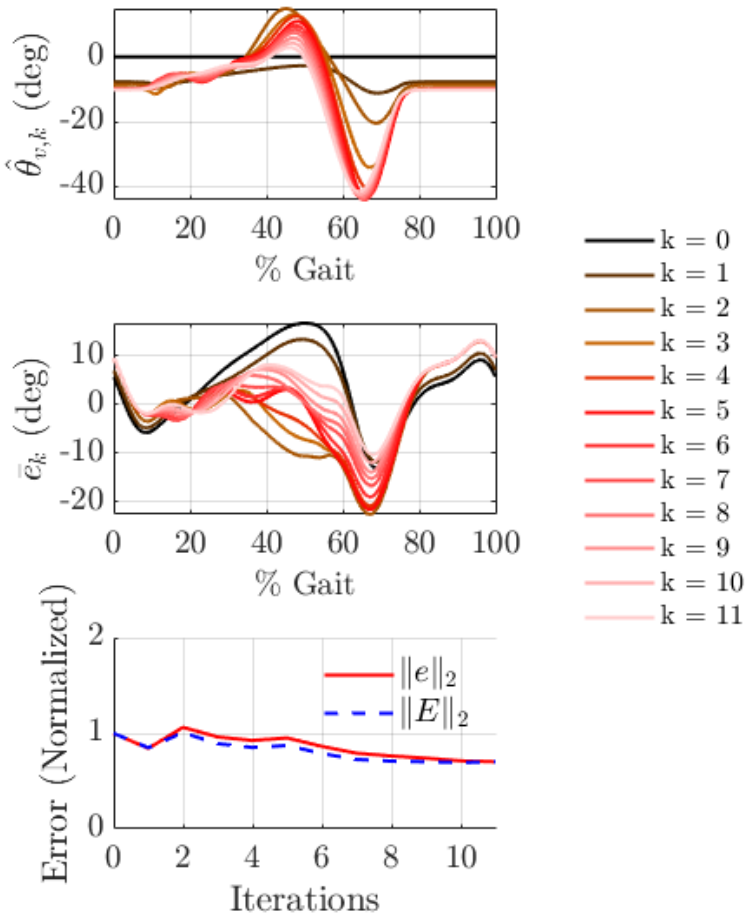


Figure 3.10: Results from the first ILC benchtop test. Time domain evolution of the virtual trajectory $\theta_{v,k}$ (top) and error $\bar{e}_k$ (middle) are shown. In addition, the 2-norm ILC error values (e for time domain and E for frequency domain) are plotted across training iterations and normalized based on the initial 2-norm error at $k = 0$.

during terminal stance as well and the negative peak around TO was shifted further to the left (backwards along the gait cycle) for iterations $6 - 11$. This was likely due to the fact that the ILC algorithm was continuously trying to adjust the virtual trajectory to correct for swing phase errors. However, since the real-time embedded controller sets the reference ankle torque to zero at swing phase (Equation 3.20), the error during swing can never be actively reduced. As a result, the stance phase errors increase as the ILC update law tries to reduce the swing phase errors by lowering the magnitude of θ_v during terminal stance and

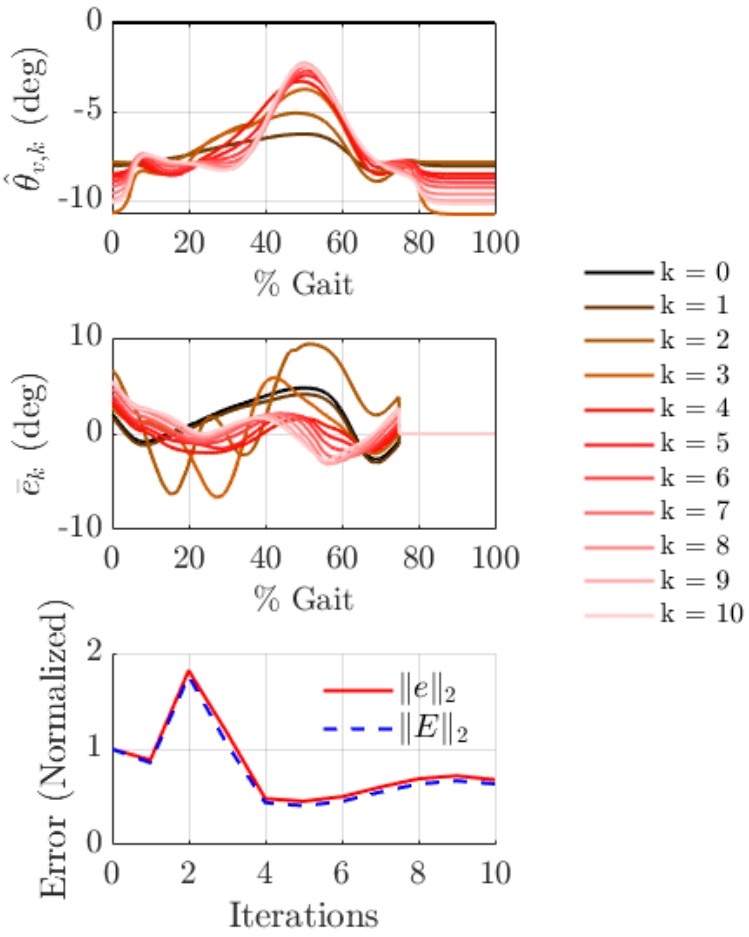


Figure 3.11: Results from the second ILC benchtop test. Time domain evolution of the virtual trajectory $\theta_{v,k}$ (top) and error $\bar{e}_k$ (middle) are shown. In addition, the 2-norm ILC error values (e for time domain and E for frequency domain) are plotted across training iterations and normalized based on the initial 2-norm error at $k = 0$.

phase-shifting the signal to the left. This explains why the 2-norm errors across the full gait cycle did not significantly reduce over 11 total iterations.

Adjustments were made for the second ILC benchtop test. First, the magnitude of the simulated biological signal was multiplied by 0.25 to lower the motor command requirements. This is justified since the PAFP's spring does not assist the motor during the benchtop tests like it would during an experimental walking trial. Another change that was made was that the ILC algorithm was modified to only learn during the stance and initial swing phases,

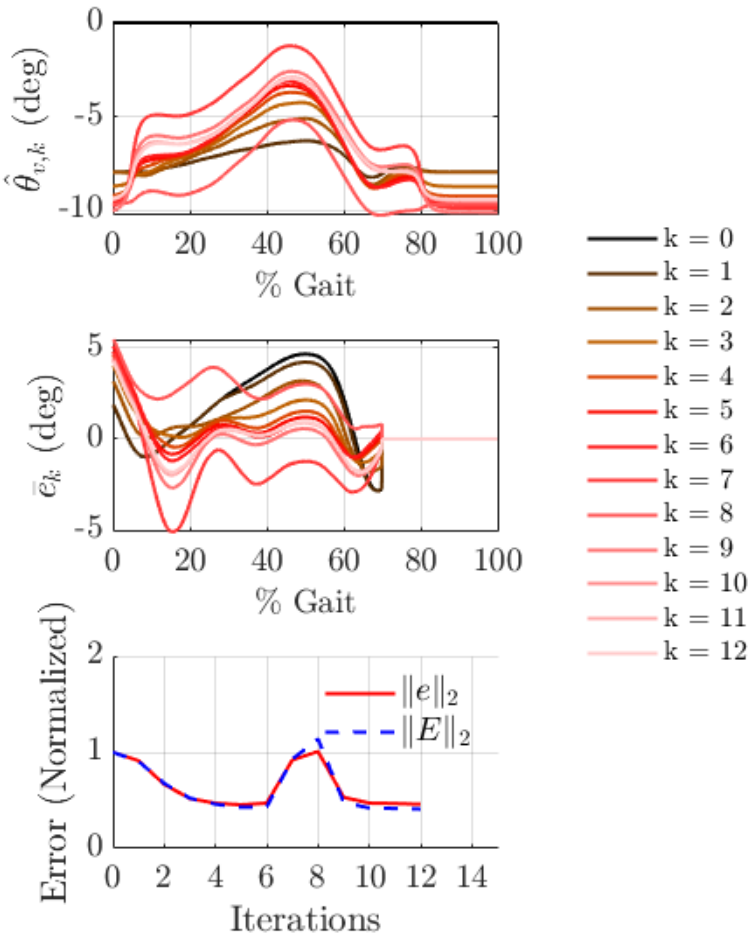


Figure 3.12: Results from the third ILC benchtop test. Time domain evolution of the virtual trajectory $\theta_{v,k}$ (top) and error $\bar{e}_k$ (middle) are shown. In addition, the 2-norm ILC error values (e for time domain and E for frequency domain) are plotted across training iterations and normalized based on the initial 2-norm error at $k = 0$.

which was accomplished by setting the ILC error to zero from $70 - 100\%$ gait.

Results from the second ILC benchtop test are displayed in Figure 3.11. The results from this test show that the normalized error reduced by a larger degree, however, it increased during earlier iterations. This behavior is undesirable as it could risk the safety of the subject. The most likely factor that caused this instability was the data-driven inverse model used in the ILC update law (see Section 3.3.2). Since only a small amount of data is used to learn the inverse model during the early training iterations, the quality of the model is likely

very low. Additionally, the ILC algorithm does not have a chance to explore different input spaces during early iterations, so the inverse model should not be expected to generalize very accurately at this stage.

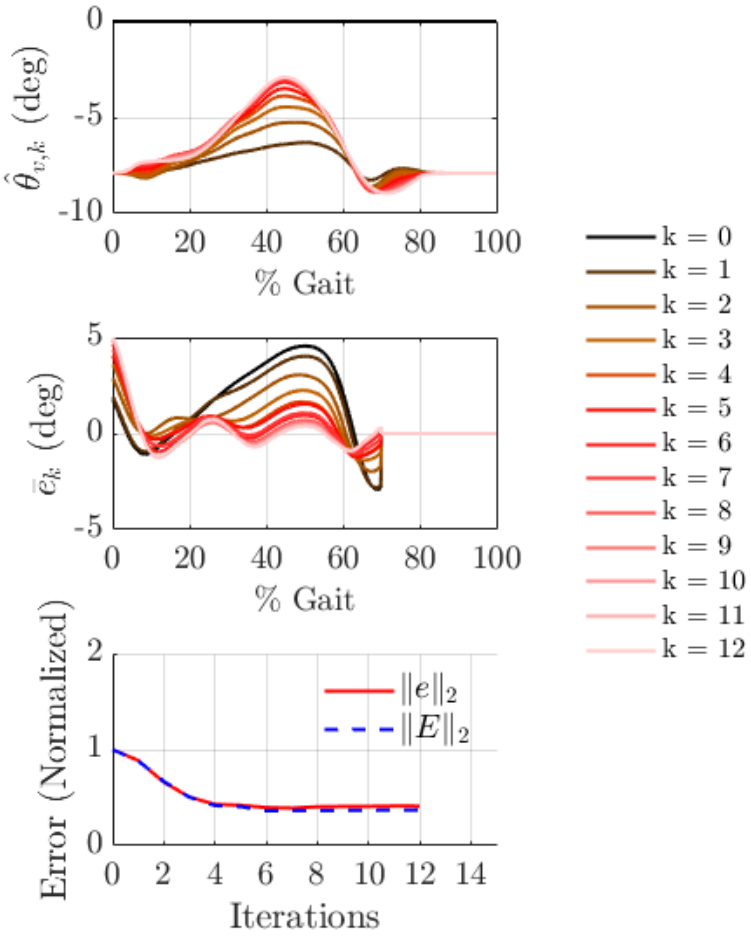


Figure 3.13: Results from the fourth ILC benchtop test. Time domain evolution of the virtual trajectory $\theta_{v,k}$ (top) and error $\bar{e}_k$ (middle) are shown. In addition, the 2-norm ILC error values (e for time domain and E for frequency domain) are plotted across training iterations and normalized based on the initial 2-norm error at $k = 0$.

To analyze the effects of including the inverse model in the ILC update law, a third ILC benchtop test was conducted. For this test, only the adaptive gain and error signals were used for the first 6 iterations to update the learned virtual trajectory. For iterations $7 - 13$, the inverse model was included in the ILC update law as well. The results shown in Figure

3.12 show that the error steadily decreased and started to converge during iterations $1-6$. However, once the inverse model was added to the ILC update law, the error signal increased by a large margin. The increased error also corresponds to sharp vertical shifts in the virtual trajectory signal during iterations 7 and 8 as seen in the top plot. After iteration 8, the error signal started to converge again but, as stated before, the sharp changes between iterations 7 and 8 would put the safety of the user at risk during an actual walking trial. Therefore, it was concluded that the inverse model should be excluded from the ILC update law (i.e., $G_k^{-1}(\omega) = 1$) in order to promote stability. The ILC update law that was implemented is as follows:

$$\theta_{v,k+1}(\omega) = \theta_{v,k}(\omega) \; + \; \rho_k(\omega)\, e_k(\omega) \tag{3.38}$$

A final ILC benchtop test was conducted to analyze the performance of the algorithm using 3.38. The learned signal θ_v eventual converged and the learning gains reduced. This can be seen in the time domain (Figure 3.13) as the learned signal and errors during iterations $k = 6-12$ had very small differences. These results suggest that the adaptive gain ILC algorithm does work as anticipated and produces a bounded signal, even without the assistance of the PAFP's passive mechanics. Note that a limitation of the ILC benchtop tests is that the reference signal remained fixed, which will not be the case during experimental walking trials.

Chapter 4

SYSTEM INTEGRATION

4.1 Overview

The controller architecture is illustrated as a block diagram in Figure 4.1 and consists of two main signal paths: real-time (solid lines) and offline (dashed lines). Additionally, sensor device connections to the human-robot system are displayed (dotted lines) which include: a force plate that measures the PAFP-side vertical ground reaction force F_p, thigh-mounted IMU which is used to compute the thigh angle θ_h, and ankle-mounted encoder which measures the prosthetic ankle angle $\theta_{p,e}$. The real-time signals are handled by an embedded system which rests on a table near the user and treadmill. The embedded system consists of: (1) a HSD algorithm that outputs a Boolean variable H which indicates when a HS has occurred, (2) the phase calculator outlined in Section 3.4, (3) lookup tables for modulating the virtual trajectory θ_v and mapping the phase variable to gait percent location Ω, (4) the impedance-inspired model which outputs the desired active torque τ_{ref} (Equation 3.20), (5) the inverse actuator model discussed in Section 3.5, and (6) a low-level proportional integral (PI) controller for tracking the desired motor current i_d.

Offline, after each walking trial (i.e., learning iteration k), inverse dynamics are computed on the motion capture (MoCap) data by the host computer resulting in the mean prosthetic ankle kinematics $\bar{\theta}_{p,m}$ and mean biological ankle kinematics $\bar{\theta}_b$. These kinematic signals are used to compute the error signal e_k defined in Equation 3.7. Using the ILC algorithm discussed in Section 3.3, a new virtual trajectory $\theta_{v,k+1}$, defined in Equation 3.38, is computed. Using measured prosthetic ankle kinematics from the MoCap system $\theta_{p,m}$ as features and ankle outputs $\theta_{p,e}$ as targets, a linear regression model G_a is trained and used to map the virtual trajectory from the MoCap subspace to the encoder subspace. This mapping allows learned signals from MoCap data to be used in real time with the encoder sensor. Finally, using the mapped virtual trajectory and the mean thigh kinematics $\bar{\theta}_h$, pattern generators are used to update the gait percentage and virtual trajectory lookup tables. All these features

are discussed in detail next.

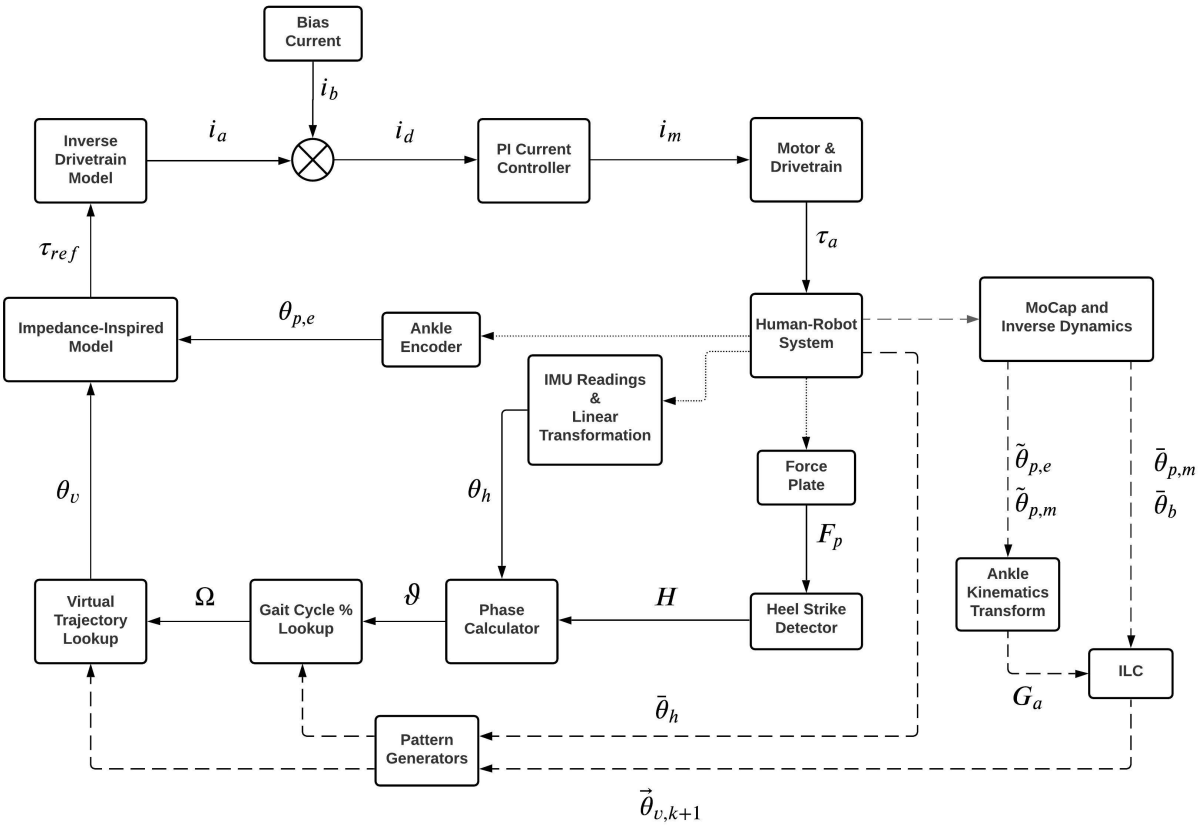


Figure 4.1: Block diagram of the proposed controller architecture. Solid lines represent real-time signals and dashed lines represent offline signals. Dotted lines represent sensors integrated into the human-robot system.

4.2 Motion Capture System & Force Plates

Gait kinematics were collected using a MoCap protocol with 16 Vantage V8 Cameras and a 63-marker full body model (Vicon). Additionally, the split-belt treadmill used for experiments is instrumented with a four-degree-of-freedom force plate (AMTI) at each of the two belts (left and right). GRFs are measured and used to compute gait kinetics via inverse dynamics [191]. In addition, the vertical ground reaction force (vGRF) signal is sampled

Table 4.1: PAFP Control Variables

Name	Symbol
Active Motor Current	i_a
Bias Motor Current	i_b
Desired Motor Current	i_d
Measured Motor Current	i_m
Desired Active Torque	$\boldsymbol{\tau}_{ref}$
Active Ankle Torque	$\boldsymbol{\tau}_a$
Prosthetic-Side Vertical Ground Reaction Force	F_p
Heel Strike Indicator	H
Thigh Angle	θ_h
Time-Normalized Mean Thigh Angles	$\bar{\theta}_h$
Prosthetic Ankle Encoder Angles from Trial k	$\tilde{\theta}_{p,e}$
Time-Normalized Mean Prosthetic Ankle Encoder Angles	$\bar{\theta}_{p,e}$
MoCap Prosthetic Ankle Angles from Trial k	$\tilde{\theta}_{p,m}$
Time-Normalized Mean MoCap Prosthetic Ankle Angles	$\bar{\theta}_{p,m}$
Time-Normalized Mean MoCap Biological Ankle Angles	$\bar{\theta}_b$
Linear Map from MoCap to Encoder Subspace	G_a
Phase Variable	ϑ
Gait Percent Location	Ω
Instantaneous Virtual Trajectory Value	θ_v
Gait Cycle-Indexed Virtual Trajectory for Next Trial	$\vec{\theta}_{v,k+1}$

in real time by the embedded system as an analog input voltage that helps determine HS instances.

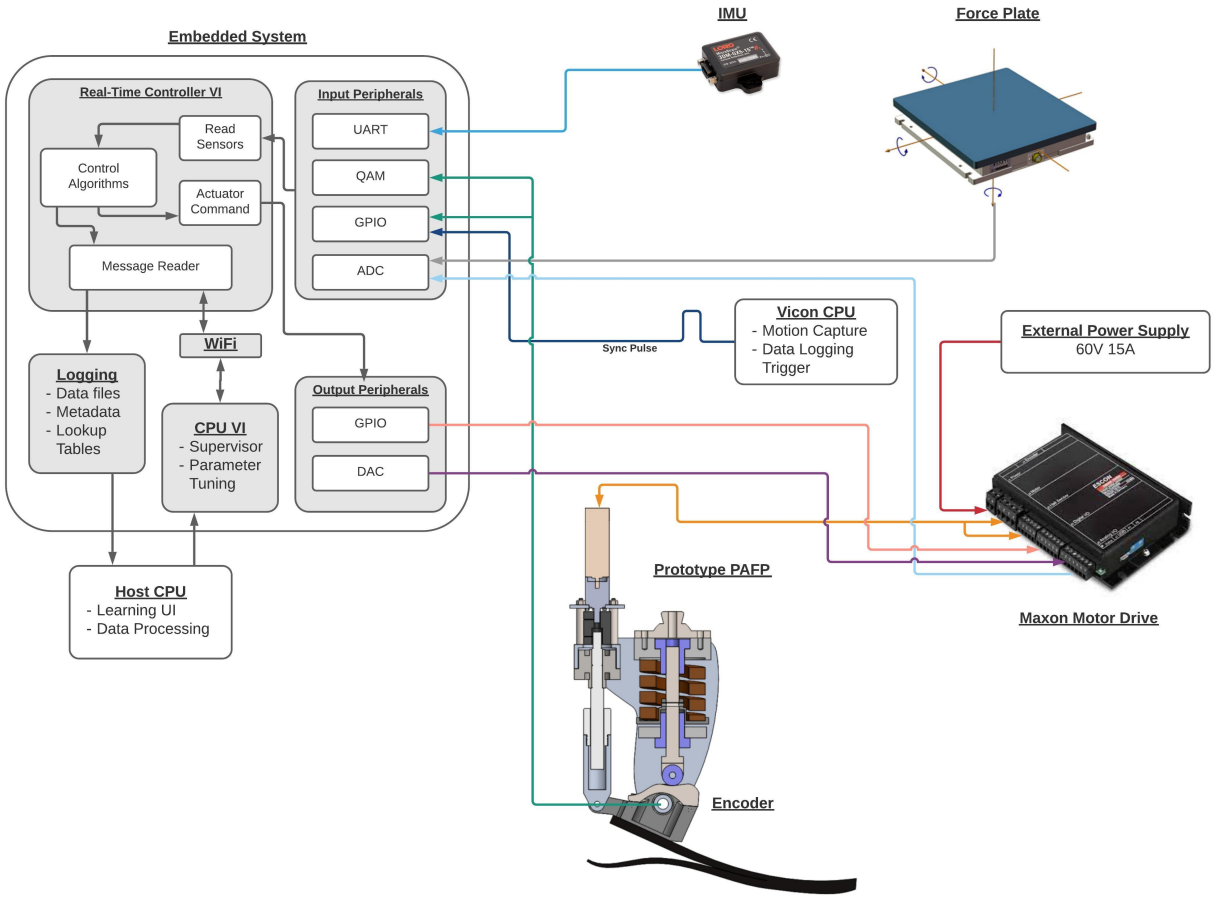


Figure 4.2: Embedded system integration and wiring.

4.3 Heel Strike Detection

The HSD algorithm determines the instance when the most recent HS occurs. It utilizes the two vGRF signals from the each force plate to measure the ground interactions at each foot. Note that even though HS instances were logged for both feet, only the prosthesis HS instance is used in the control software. For each control loop n, the raw force signal $F_p[n]$ is digitally filtered using two digital low-pass filters in series as seen in Figure 4.3. The

filter coefficients for each stage of the filter structure are derived from a continuous low-pass first-order Butterworth filter with cutoff frequency of 50 Hz and a control loop sampling frequency of 500 Hz. This produces sufficiently smooth signals at each filter output with minimal phase delay. An estimate of the force rate of change $\delta F_p[n]$ is calculated as the difference between the first low-pass filter's output and the final filtered output.

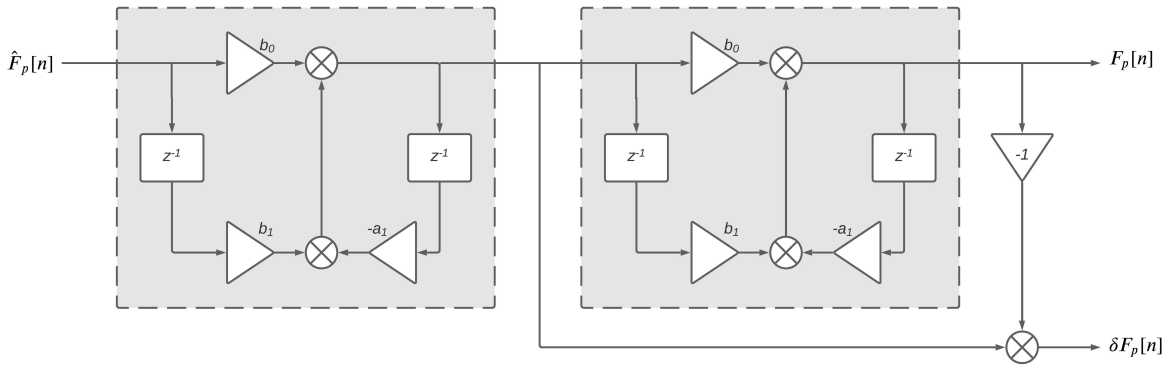


Figure 4.3: The cascaded filter structure, where $\hat{F}_p$ is the raw force plate signal on the prosthetic side.

The FSM for HSD, depicted in Figure 4.4, consists of two states: (1) state 1, denoting no foot-ground contact, and (2) state 2, denoting foot-ground contact. The transition between states is determined by thresholding the force signal $F_p[n]$ and the force rate of change $\delta F_p[n]$. If the force rate of change $\delta F_p[n]$ is increasing (i.e., $\delta F_p[n] > \delta F_{th}$) and the force $F_p[n]$ is below the threshold value (e.g., $F_p[n] < F_{th}$), then a transition from state 1 to state 2 occurs. If the force rate of change $\delta F_p[n]$ is decreasing (e.g., $\delta F_p[n] < 0$) and the force $F_p[n]$ is below the swing threshold value (i.e., $F_p[n] < F_{swing}$), then a transition from state 2 to state 1 occurs.

During stance phase, the force signal includes two peaks, one occurring at HS and the other occurring at FF. These peaks are approximately equal in magnitude so undesirable changes in state can occur, e.g., the second force peak could cause the controller to measure a false HS within a single gait cycle. Tuning the HSD is difficult without aggressive filtering (which can increase signal latency) due to the double peak behavior of the force signal. To solve this issue, $F_p[n]$ is hard clipped so it can never be above F_{th}. The saturated signal

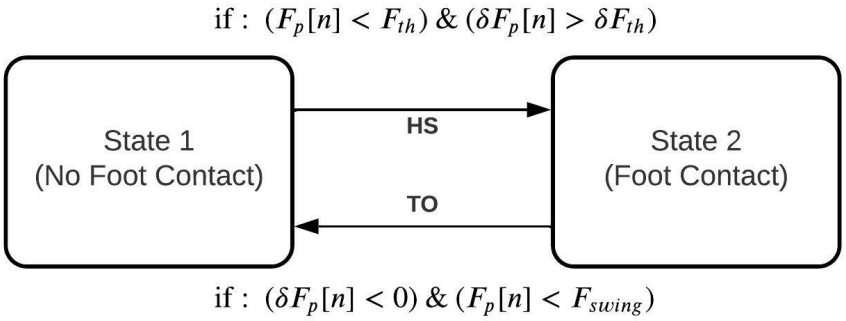


Figure 4.4: Finite state machine (FSM) for HSD, where F_p is the filtered force signal from the PAFP-side force plate, δF_p is an estimate of the rate of change of the force signal, F_{th} is the force saturation threshold, δF_{th} is the rate of change threshold for HSD, F_{swing} is the force threshold to indicate when TO occurs, and n is the current timestamp.

$\tilde{F}_p[n]$ is defined as

$$\tilde{F}_p[n] = \begin{cases} F_{th}, & \text{if } F_p[n] > F_{th} \\ F_p[n], & \text{otherwise} \end{cases} \tag{4.1}$$

This produces a smooth $\delta F_p[n]$ signal that has two distinct peaks (positive and negative) which is beneficial when tuning the force rate threshold δF_{th}. This algorithm works as long as F_{th} is set below the local minima found in the raw force signal. Figure 4.5 shows the raw and saturated force signals as well as the resultant force rate signal which has two peaks indicating the loading and unloading rates of the foot.

When a transition from state 1 to state 2 is identified, the algorithm senses that a HS has occurred. A flag H (Boolean variable) is then raised which is sent to the phase calculator, triggering the thigh angle integral to reset. Additionally, the scale and shift variables z and γ are updated based on the thigh angles of the previous gait cycle (i.e., previous HS to most recent HS). When a transition from state 2 to state 1 is identified, the algorithm senses that a TO has occurred, however, this event is not used in the phase calculator. Rather, the TO indicator is used with the phase calculator to set the control signal to a preset bias current that helps prevent toe drag and prepares the subject for the next HS event. Figure 4.5 shows

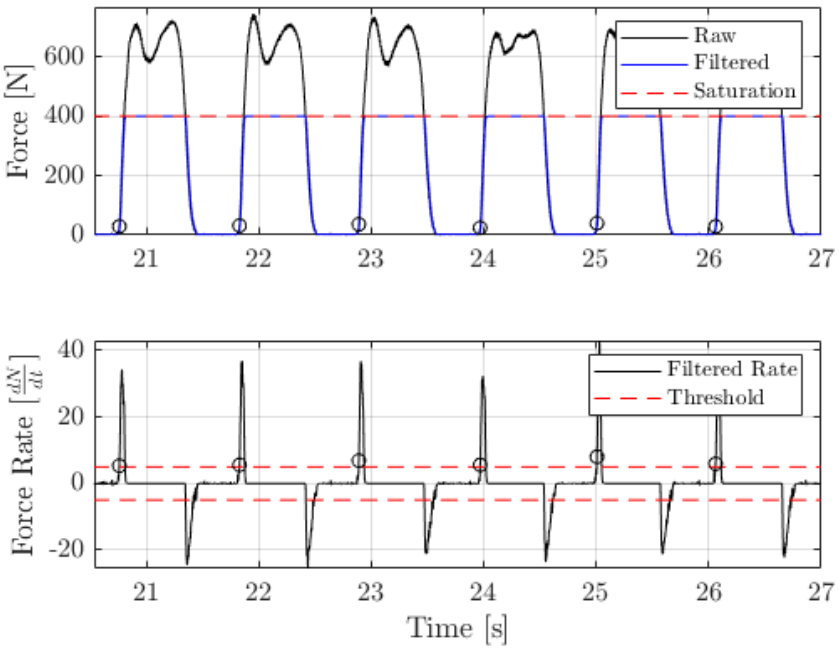


Figure 4.5: HSD during a preliminary control subject experiment. The top plot shows the raw vertical ground reaction force signal from the force plate as well as the filtered and saturated signal. The saturation threshold is represented by a red dashed line. The bottom plot shows the rate of change in the ground reaction force. The thresholds that trigger state changes in the FSM are shown as horizontal red dashed lines. For both plots, the HS events are represented as circular markers.

the results of the HSD during a preliminary human subject experiment.

4.4 Ankle Encoder

A 12-bit capacitive encoder (CUI Devices, AMT113S-V) is used to sense the angular position of the prosthetic ankle joint $\theta_{p,e}$. The encoder interfaces with the embedded system via digital square waves (signals A and B) in quadrature. The waveform frequency indicates the speed of shaft rotation and the number of pulses indicates the distance moved, whereas the A-B phase relationship indicates the direction of rotation. Additionally, an index pulse (signal Z) from the encoder is read by the embedded system via a digital interrupt pin which resets

the encoder count whenever this signal is high.

The encoder housing is mounted on the surface of the shank link. The sleeve and shaft adapter of the encoder contact and measure the position of the ankle joint shaft. This shaft is then mechanically coupled to the ankle link via a pin connection which allows the encoder to measure the angular kinematics of the PAFP. However, unlike absolute encoders, an incremental quadrature-based encoder does not keep track of the absolute position of the shaft it is attached to. To overcome this limitation, custom software from the manufacturer can set the digital index at the current angular position of the system. This position is stored in non-volatile memory and will remain present until a zero command is set again or encoder is reprogrammed. For this research, the index pulse was manually set to the PAFP's unloaded resting position where the ankle angle is -5 degrees. Note that all subsequent calculations take this 5 degree offset into account.

4.5 Thigh-Mounted Inertial Measurement Unit

In order to compute the phase variable, an IMU (LORD MicroStrain, 3DM-GX4-15) is mounted on the user's thigh within the sagittal plane in order to measure its angular position θ_h. The IMU is compact and contains a triaxial accelerometer, gyroscope, and magnetometer. Dual onboard processors run an auto-adaptive EKF based on Newton's and Euler's equations of motion to compute real-time Euler Angles (roll, pitch, and yaw) in the IMU coordinate frame. The auto-adaptive EKF dramatically improves mean and peak errors for both pitch and roll estimates in dynamic conditions (non-constant velocity) by continuously monitoring and adapting to the acceleration conditions. The acceleration vector is only used as a gravity reference when it appears that the lateral accelerations are zero and the magnitude of the gravity vector is within the default bounds.

The Euler angle outputs are sampled at 500 Hz and are read by the embedded system via a universal asynchronous receiver-transmitter (UART) interface. Within the embedded program, a producer loop runs, which runs in parallel to the control loop, constantly streams the sampled Euler angles and updates a shared variable that contains the read bytes. This was done to avoid missing data and overflowing the buffer. The control loop then accesses the shared byte array variable at each control sample n and converts the bytes to interpretable Euler angle values. Once the roll and pitch angles have been decoupled via a linear transformation, the thigh angle θ_h is computed and sent to the phase variable calculator.

4.6 IMU Linear Transformation for Computing Thigh Angle

Once the user starts walking and several strides of IMU data have been logged, a principal components analysis (PCA) was conducted to compute a linear transformation that decouples the roll and pitch angles [192]. Since it is difficult to place a device perfectly parallel to an anatomical plane (sagittal in this case), the PCA weights can be used as a transformation matrix that, when multiplied by the roll and pitch IMU angle vectors, result in more accurate frontal and sagittal plane thigh angles. These corrected angles are uncorrelated and the first principal component is used as an improved estimate of the thigh kinematics θ_h within the sagittal plane. The first principal component is chosen since it is expected that the sagittal plane has the largest motion range for steady-state walking.

4.7 Pattern Generators

The pattern generators are used to update lookup tables after each iteration k (i.e., each learning trial). The first lookup table is populated with the gait percent locations Ω based on the time normalized mean phase variable $\bar{\vartheta}$. The second lookup table is populated with the virtual trajectory θ_v along the gait cycle. In practice, it is useful to eliminate this setpoint signal during the end points of the gait cycle in order to avoid discontinuities that can risk the safety of the user. Therefore, the learned trajectory θ_v at each iteration k was eliminated from $0-5\%$ $80-100\%$ gait. To achieve this, the virtual trajectory lookup table was set to the encoder output at swing during these two gait sub phases. The signal is then filtered using a zero-phase low-pass filter in order to generate a smooth signal transition from the end of gait to the start of the next gait cycle.

Each lookup table is stored as a 1000 element array of 64-bit double type variables. The first element corresponds to 0.1% of gait and the last element corresponds to 100% gait. During the walking trial, these lookup tables are deployed and linear interpolation was performed between the closest precalculated adjacent elements to the value inputted into each table (precomputation with interpolation). Note that no extrapolation strategy is used when deploying the lookup tables, rather, the nearest bound of the dependent variable is used if the independent variable input is outside the expected range. For example, the gait cycle lookup table outputs 100% gait if the input value (normalized phase variable) is greater than 1.

4.8 Bias Current

The cam-based parallel spring results in the prototype PAFP having a plantarflexed unloaded neutral position caused from the preload in the elastic follower. Therefore, to prevent toe drag during swing, a bias current term i_b offsets the preload. The procedure for choosing the bias term i_b is described in Section 5.2.4.

4.9 Low-Level PI Motor Controller

The low-level current tracking is implemented with a four-quadrant servo controller (Maxon, ESCON 70/10). The control command u at the n^{th} sample is defined as

$$u[n] = \kappa \, i_a[n] \tag{4.2}$$

where $\kappa \in [0, 1]$ is a control gain that allows scaling of the desired active current i_a (Equation 3.35). The control gain is manually scaled over the acclimation period to allow the user to gradually adapt to the active assistance without the safety risk of a bang-bang control scheme. During an experimental walking trial, the researcher increases κ by increments of 0.1 during swing phases on the prosthetic side. The desired motor current at the n^{th} control loop,

$$i_d[n] = u[n] + i_b[n] \tag{4.3}$$

is encoded by the embedded system as an analog voltage output. The servo controller takes this analog voltage setpoint as an input and provides the required motor communication, using hall-effect feedback from the brushless EC motor (Maxon EC-22, 120 W, 18 V). The actual motor current i_m is measured from the Maxon motor drive and sampled by the embedded system as an analog input voltage. Electrical current tracking during a preliminary experiment is shown in Figure 4.6.

4.10 Linear Transformation to Map Ankle Kinematics

The virtual kinematic trajectory signal from the ILC algorithm is computed using the error signal between the biological and prosthetic kinematics measured by the MoCap system. However, reading MoCap data and computing inverse dynamics cannot be done easily or

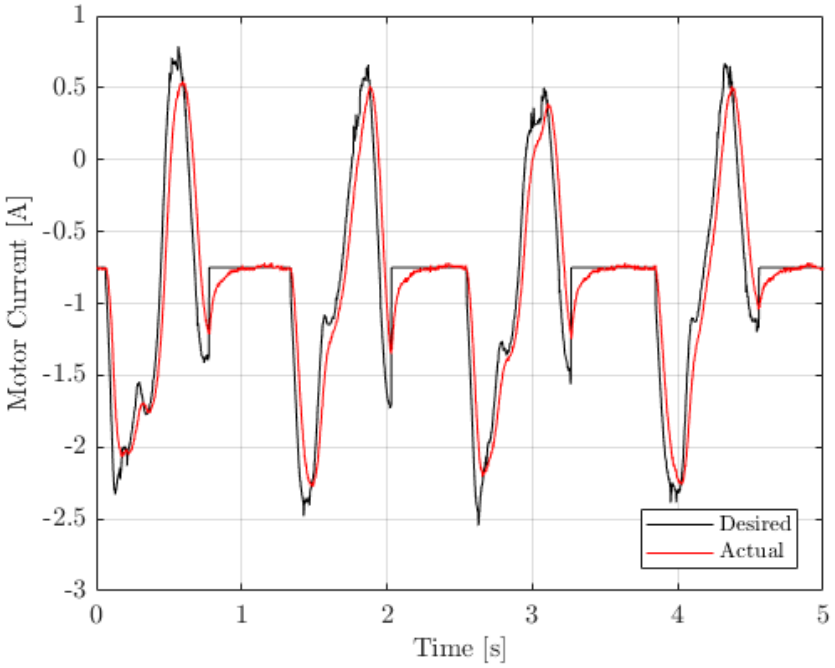


Figure 4.6: Electrical current tracking performance of the servo controller during benchtop testing. The desired current consists of both a constant bias current as well as the active current command. The actual motor current command achieved from the low-level PI controller is sampled from the ESCON 70-10 drive.

quickly in real time. On the other hand, encoder measurements are precise, reliable, and quick for real-time control. Therefore, the subspace of the learned virtual trajectory using MoCap data $\theta_{v,m}$ must be transformed to the subspace of the encoder before conducting the next experimental walking trial. To achieve the learned trajectory in the encoder domain $\theta_{v,e}$, a simple model is trained offline between training iterations using linear regression [193, 194] of the form

$$\theta_{v,e} = p_0 + p_1 \theta_{v,m} \tag{4.4}$$

where p_0 and p_1 are fitted parameters from the measured data. The use of a simple linear model is justified since the difference between MoCap and encoder signals can likely be described by a single change in magnitude and bias shift. This is demonstrated by the signals having relatively similar shapes as seen in Figure 4.7. Additionally, a linear model

is more likely to generalize better to kinematics not captured by the training data, which is important since the virtual trajectory may span outside the training space [195]. The MoCap-measured ankle kinematics $\theta_{p,m}$ from the most recently collected trial are used as the feature vector while the encoder-measured ankle kinematics $\theta_{p,e}$ from that same trial are used as the target vector [196]. The model is trained using MATLAB's fitlm function and example results are seen in Figure 4.7.

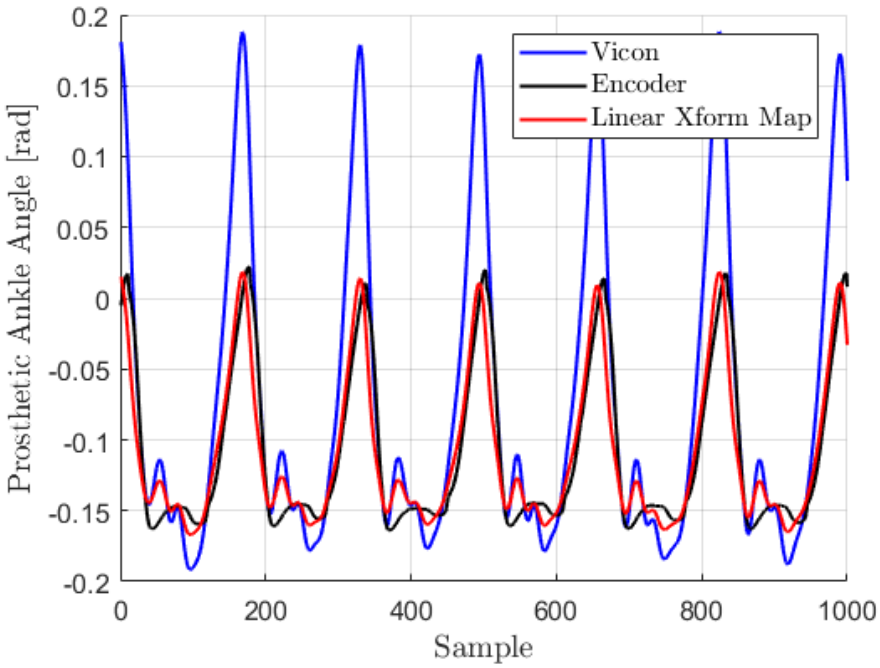


Figure 4.7: Comparison of prosthetic ankle angles from Vicon measurements, ankle encoder sensor outputs, and linear map which transforms Vicon measurements to the encoder subspace for real-time feedback control.

4.11 Embedded System Implementation

The embedded system integrates a 32-bit CPU, portable real-time evaluation board (myRIO, National Instruments), circuitry, motor servo controller, and sensors described above. The shared memory space between the CPU and myRIO integrates multi-level control, tuned parameters, the state of the system (e.g., sensor and control values at each control loop

n), lookup tables, and data logging. The system is programmed using National Instruments LabVIEW software. In addition, the system makes use of several standard electrical communication peripherals such as analog-to-digital conversion (ADC), digital-to-analog conversion (DAC), UART, general purpose input output (GPIO), digital interrupts, and quadrature amplitude modulation (QAM). The real-time system provides a control loop sampling frequency of 500 Hz and is externally powered using a tether. An additional tether to an external configurable power supply provides power to the servo controller. Data logging between the MoCap measurements and embedded system sampling is synced via a pulse generated by the Vicon system.

4.11.1 CPU Program

The main LabVIEW virtual instrument (VI) running on the host CPU is a supervisory user interface used for monitoring signals and making adjustments to the controller. The researcher is able to change configurable parameters, input settings, load new lookup tables, monitor the various signals, activate the controller, adjust the control gain during acclimation, and execute an emergency stop if necessary. The host VI can access signals from the embedded system through the myRIO's built-in WiFi functionality (wireless 802.11b,g,n). In order to reduce data size for storing, handling, and transmitting content, a lossy data compression protocol is used to pass data from the embedded system to the host CPU.

4.11.2 Real-Time Software

The real-time programs running on the myRIO are responsible for sampling sensor measurements, executing standard signal processing peripherals, implementing the control algorithms, storing data for analysis, sending lossy data to the host CPU, and receiving messages from the host CPU. Even though most myRIO applications are for academic purposes due to the ease of interfacing with instrumentation, it is well-suited for portable real-time control systems that require many input-output (I/O) options and high performance.

The myRIO runs a Linux-based real-time operating system which allows for multithreading and parallel operations without programming a large amount of interrupt handlers. Along with the microprocessor (Xilinx Z-7010, Dual-Core ARM Cortex-A9), the myRIO includes a field-programmable gate array (FPGA) which is a reprogrammable silicon chip. In contrast

to the microprocessor, programming a FPGA rewires the chip itself rather than run a software application. This allows for high-speed data acquisition, parallel processing, filtering, logic, and digital I/O on the order of 40 MHz, which is much faster than most microcontroller devices. The FPGA of the myRIO has predefined functionality which removes the barrier of learning and programming VHDL or Verilog code. This abstracts the details away from the low-level electronic support structures and allows for greater focus to be placed on the real-time algorithms, however, FPGA customization can still be achieved through the LabVIEW FPGA Module. Additionally, the myRIO has built-in WiFi without the need of Ethernet or a USB adapter. Finally, the myRIO includes onboard DAC and ADC functionality, designated I/O for serial communication interfaces (SPI, UART, I2C), and high-resolution data acquisition protocols. The three main programs running on the real-time myRIO device are described next.

Message Reader

Using a network stream, a buffered and lossless communication strategy that ensures data written to the stream is never lost, the real-time system is able to read messages from the host CPU. This program is responsible for interpreting the CPU message and executing the correct response within the real-time environment. The various protocols initialized by the host CPU and carried out by the real-time program are as follows:

- Stopping the program
- Activating the controller by sending a digital high signal to the motor
- Initializing data logging to a file
- Adjusting the scaling gain of the motor command
- Adjusting the force threshold for the HSD algorithm
- Adjusting the force rate threshold for the HSD algorithm
- Changing the bias current value
- Changing the data logging file name

- Changing the data logging time period
- Changing the phase normalization parameter values
- Choosing whether or not to unwrap the phase signal
- Changing the cutoff frequency for the HSD filter
- Changing the number of samples in the moving average computation for the phase parameters
- Initialize PCA training for the IMU signals
- Wireless network reader timeout

All these protocols read the lossless message from the CPU once using a network shared (global) variable and update a corresponding single-process (real-time FIFO) variable. This is done to reduce latency in the control loop for deterministic and time-critical algorithms.

Deterministic Tasks

The deterministic tasks real-time program, shown in Figure 4.8, is responsible for executing time-critical control actions. These tasks include sampling sensor measurements, implementing the controller algorithms, sending electrical current commands to the motor, and sending lossy data to the CPU and data logger program.

Data Logging

The data logging real-time program, shown in Figure 4.9, runs in parallel to the deterministic tasks and message reader programs. The program runs in an idle state until a data logging flag is raised. During walking experiments, this flag is raised using an external pulse signal generated from the Vicon system in order to make sure the timesteps logged by the embedded and MoCap systems are synced to each other. The flag can also be set to be raised using a button on the CPU user interface whenever the MoCap system is not being used (e.g., during validation testing).

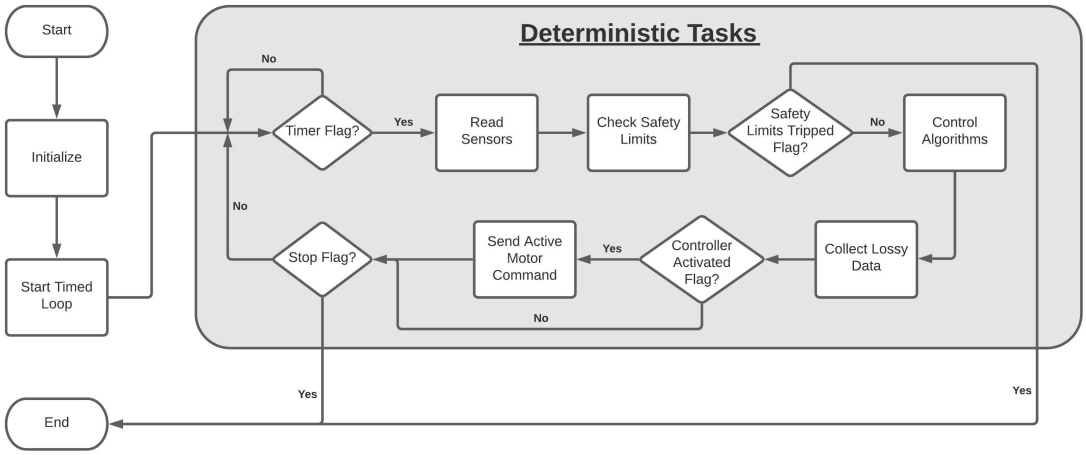


Figure 4.8: Flowchart illustrating the software program (LabVIEW VI) that executes the deterministic tasks of the real-time device.

Once the data logging flag has been raised and read by this program, the program starts to build an array of sampled sensor and control values. This data is sent from the deterministic tasks program to the data logging program as compressed lossy data. Once the logging time has been met, the array is written to a Technical Data Management Streaming (TDMS) file and saved on an external USB flash drive that is connected via the myRIO's USB port. Subsequently, metadata (parameters, subject information, experiment information, settings, etc.) are saved to a separate TDMS file. The data on the USB is then transferred to the host CPU for offline learning, lookup table updates, and post-hoc data analysis. A state manager is used to switch between idle, accumulating data, writing data to file, and writing metadata to file actions. Switch conditions are based on the data logging flag and logging time (see Figure 4.9).

4.11.3 Embedded System Hardware Design

The embedded system hardware is placed on a table near the user and treadmill during walking experiments. The system is packaged in a small 3D-printed casing (shown in Appendix A) and contains the myRIO, custom breakout circuitry, and motor servo controller. A lip is included around the house in order to clamp it to the table. Strain relief wire guides are included in the 3D-printed design as well. The high-current power supply and motor cables

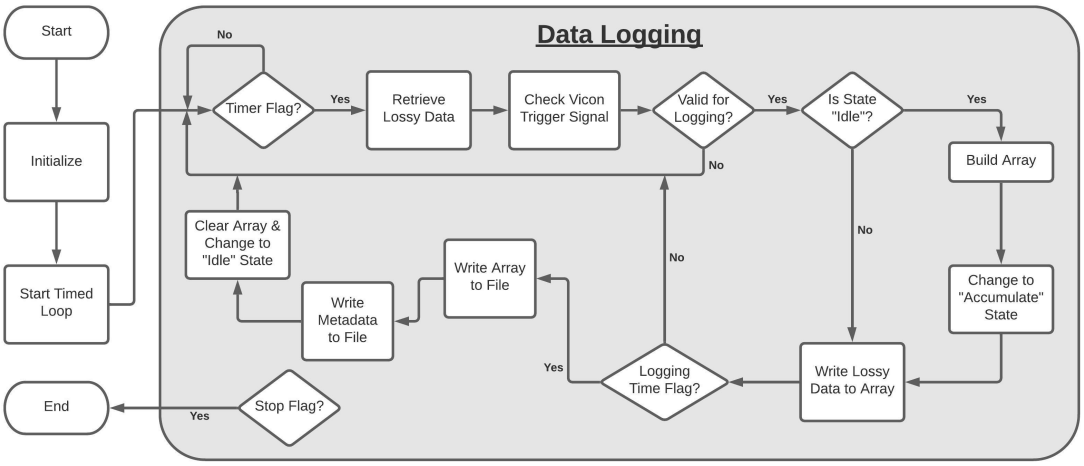


Figure 4.9: Flowchart illustrating the software program (LabVIEW VI) that executes the data logging tasks for the real-time device.

are supplied together in a tether while a second tether runs perpendicular to the first tether and contains the lower-current sensor cables. The separation of these cables was done to avoid any interference or noise caused by magnetic fields which could affect the quality of the lower-current signals. Note that the embedded system hardware can be easily redesigned into a compact and portable design. However, this is outside the scope of this research, which focuses more on evaluating the proposed control scheme.

Chapter 5

HUMAN SUBJECT EXPERIMENTAL EVALUATION

5.1 Introduction

The purpose of this study was to investigate whether the proposed control law improves gait symmetry for individuals with below-knee amputations. This chapter outlines the setup for data collection, experimental conditions, experimental protocol, research participants, and analysis pipeline. Results are then presented and summarized based on the study's hypotheses. The control law's performance toward proper PAFP functionality is then discussed.

5.2 Experimental Methods

The outcomes of this experiment are designed to conclude whether the novel PAFP control law: (1) promotes symmetrical biomechanics about the ankle and (2) reduces peak intact KAM and peak intact HAM. Three prosthesis conditions are assessed. The first condition consists of the human subject ambulating with their clinically prescribed prosthetic ankle-foot. The next condition consists of the subject ambulating with the prototype PAFP in passive mode, i.e., when the active current command is zero. The third conditions consists of the subject ambulating with the prototype PAFP in active mode, i.e., when the active current command is non-zero. The hypotheses are as follows:

- **H1.1** The control law will reduce ankle angle asymmetry, when compared with the passive condition.

- **H1.2** The control law will reduce ankle moment asymmetry, when compared with the passive condition.

- **H1.3** The control law will reduce ankle power asymmetry, when compared with the passive condition.

- **H2.1** The control law will reduce the intact peak knee adduction moment, when compared with the passive and prescribed conditions.

- **H2.2** The control law will reduce the intact peak hip adduction moment, when compared with the passive and prescribed conditions.

5.2.1 Experimental Setup

The experimental setup, shown in Figure 5.1, consists of an AMTI split-belt force-sensing treadmill, Vicon MoCap system with 16 vantage V8 cameras, and a human subject wearing either their prescribed prosthesis or the prototype PAFP connected to the custom embedded system and tethered power supply. The MoCap marker set consists of 63 reflective markers arranged in a modified version of Vicon's full body plug-in gait model (as shown in Figures 5.2 and 5.4). The prosthetic devices were fitted to the amputated limb of each subject by a certified prosthetist.

5.2.2 Subjects

The protocol was performed with two subjects who provided written informed consent to participate in the experimental protocol, approved by the VA and UW Institutional Review Boards, a copy of which is provided in Appendix E. The subjects were healthy and active individuals with unilateral transtibial amputation. Table 5.1 provides details of the subject-specific parameters.

5.2.3 Learning Algorithm and Controller Parameters

The control stiffness parameter K, defined in Equation 3.3 and used to convert kinematic errors into appropriate torque reference values via an impedance control law, was chosen as $K = 200$. The initial learning gain ρ_0 used in the ILC update law (see Equations 3.14 and 3.19) was initialized to a value of 0.5 and the maximum learning harmonic $\omega_{max} = 10 \cdot \omega_0$ such that,

$$\rho_0(\omega) = \begin{cases} \rho, & \text{if } \ \omega \leq \omega_{max} \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

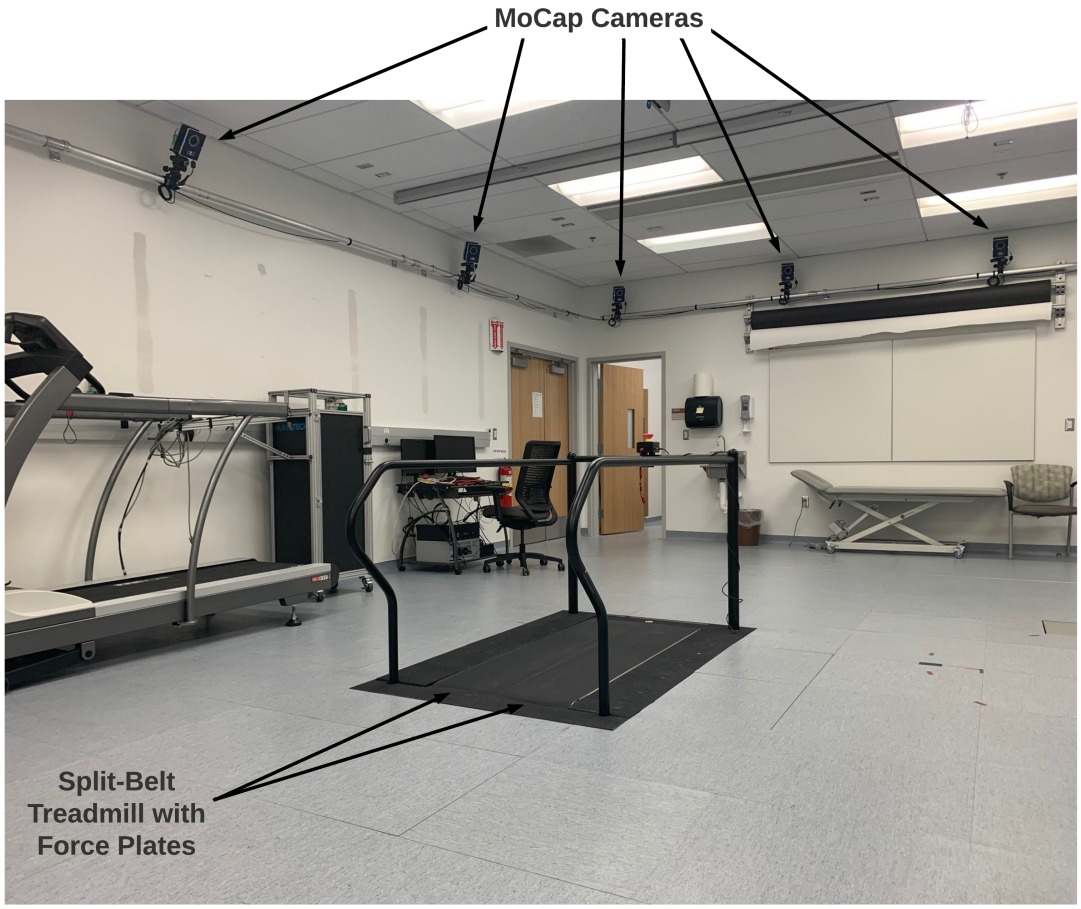


Figure 5.1: Motion Analysis Lab setup including split-belt treadmill with force plates and high-speed motion capture cameras.

The filter cutoff frequency, described in Section 4.7, was chosen to be equivalent to the maximum learning harmonic $\omega_{max} = 10 \cdot \omega_0$. Prior to filtering, the learned virtual trajectory was eliminated (i.e., set to the mean encoder output during swing phase) during the early portion of stance phase and the end of swing phase. These regions are defined by t_1 and t_2 and the signal is eliminated using the following rule:

$$\hat{\theta}_{v,k+1}(t) = \begin{cases} \theta_{v,k+1}(t), & \text{if } \ (t < t_1) \text{ and } (t > t_2) \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

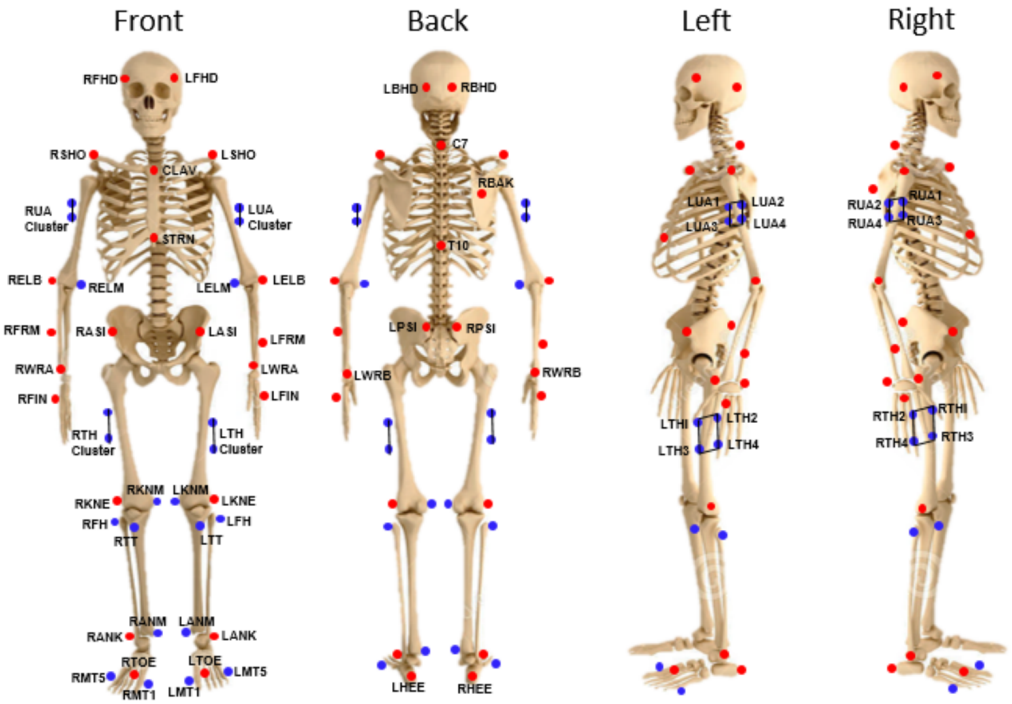


Figure 5.2: Modified plug-in gait reflective marker set arrangement. Original plug-in gait model markers are displayed as red dots. Additional markers are displayed as blue dots. Modified from notes (Krista Cyr).

The assigned values of t_1 and t_2 were 5 and 80 respectively. Therefore, the learned signal was eliminated from $0-5\%$ and $80-100\%$ gait. Additionally, the ILC error was eliminated for the majority of swing phase in order to focus the trajectory learning and improve the controller performance during stance phase. The learning cutoff time parameter t_L was used to eliminate these errors,

$$\hat{e}_k(t) = \begin{cases} e_k(t), & \text{if } \ (t < t_L) \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

For this study, the learning cutoff time was chosen as $t_L = 70$. The maximum allowable growth factor α and the decay factor ζ, defined in Equation 3.14, were chosen as 0.5 and 1

Table 5.1: Subject-Specific Parameters

Parameter	**A01**	**A02**
Gender	Male	Male
Age (years)	57	29
Height (cm)	178.5	164.5
Amputated Side	Right	Left
Intact Leg Length (mm)	807.5	801.5
Prosthetic Leg Length (mm)	809.5	802
Mass with Prescribed (kg)	85.3	81.9
Mass with Prototype (kg)	89.9	84.1
Prescribed Prosthesis	Ossur Cheetah Xplore	Ossur Pro-Flex XC Torsion
Treadmill Speed (m/s)	1.0	0.7

respectively. The frequency-dependent padding $\varepsilon(\omega)$ in Equation 3.12 was chosen as three times the standard deviation of the magnitude of the prosthetic ankle angles during the first learning iteration (i.e., passive trial),

$$\varepsilon(\omega) = 3\sigma_M(|\theta_{p,0}(\omega)|) \tag{5.4}$$

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

$$\sigma_M(|\theta_{p,0}(\omega)|) = \frac{1}{M}\sum_{m=1}^{M}(|\theta_{p,0,m}(\omega)| - \mu_M(|\theta_{p,0}(\omega)|))^2 \tag{5.5}$$

$$\mu_M(|\theta_{p,0}(\omega)|) = \frac{1}{M}\sum_{m=1}^{M}|\theta_{p,0,m}(\omega)| \tag{5.6}$$

Note that $\theta_{p,0,m}$ represents the prosthetic ankle angle at the m^{th} step during iteration $k = 0$ (i.e., passive trial). Table 5.2 summarizes the learning algorithm and controller parameters used in this study.

5.2.4 Data Collection Protocol

The experimental protocol, which consisted of two visits, is illustrated in Figure 5.3 and discussed next.

Setup

At the start of their first visit, each subject provided written informed consent to participate in the experimental protocol, as approved by the VA IRB. After the consent process, the subject walked three times along a 20-meter hallway using their prescribed prosthesis. The time to complete these trials was averaged and used to compute the subject's preferred walking speed. Next, a certified prosthetist fitted the prototype PAFP to the subject. After this, anthropometric measurements were taken and the Vicon reflective markers were placed. The total setup time took approximately $1 - 1.25$ hours to complete.

Prescribed Walking Trials & Prototype Pretest

The subject was asked to walk on the instrumented treadmill using their prescribed prosthesis. Prescribed device experimental data were collected, which consisted of 30 seconds of steady state walking for each trial. For next part of the first visit, the Vicon markers were removed and the subject was re-fitted with the prototype PAFP. The subject was asked to choose a bias current i_b for the device, through manual tuning, which felt comfortable during static standing. Due to the preload of the PAFP's nonlinear spring, which causes the prosthetic foot's resting position to be plantarflexed, this bias current was necessary in order to overcome the preload and adjust the neutral position of the prosthetic foot to avoid toe drag during swing phases.

Once a comfortable bias current was discovered, the subject acclimated to the prototype PAFP by walking on the treadmill for approximately 5 minutes. The bias current

Table 5.2: Algorithm and Controller Parameters

Control Stiffness	K	200
Initial Learning Gain	ρ_0	0.5
Filter Cutoff	ω_c	$10 \cdot \omega_0$
Pre-Smooth Time (%)	t_1	5
Post-Smooth Time (%)	t_2	80
Learning Cutoff Time (%)	t_L	70
Growth Factor	α	0.5
Decay Factor	ζ	1
Padding Factor	ε	3

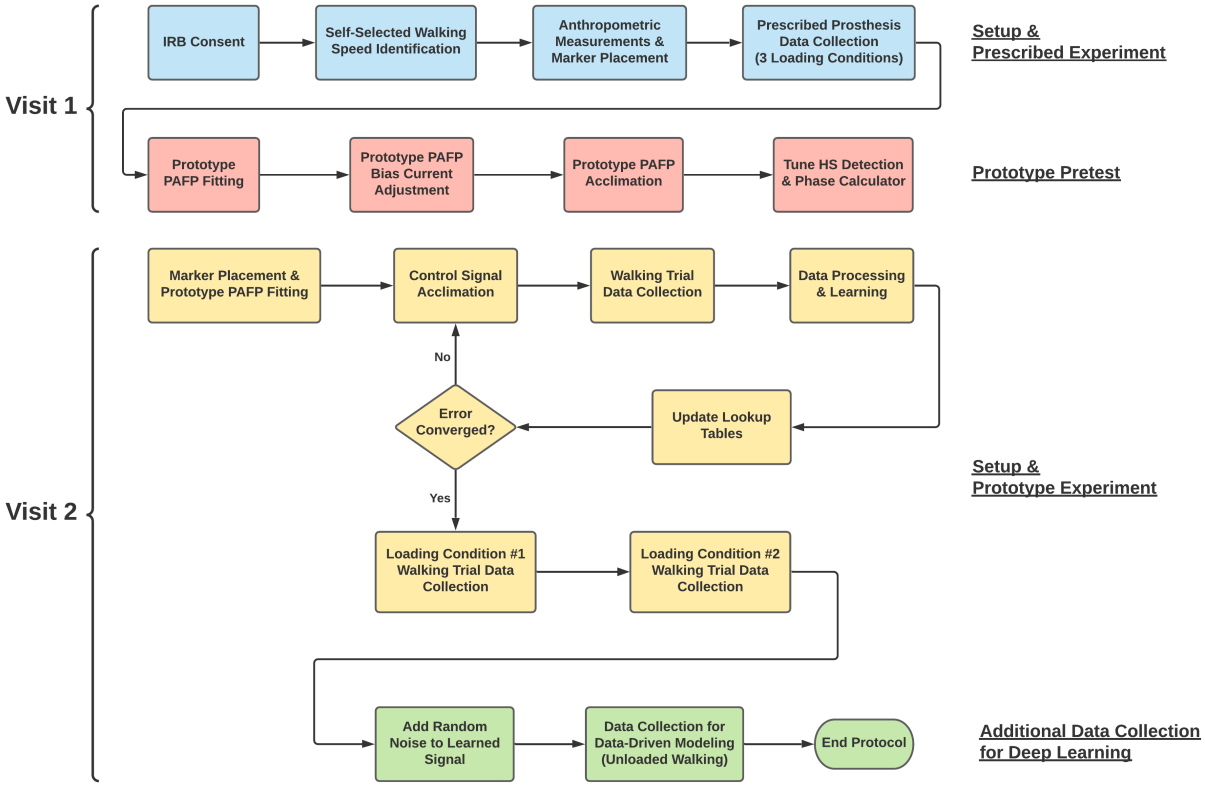


Figure 5.3: Experimental protocol divided into two visits.

was then verified and adjusted further if necessary during this period. Additionally, phase calculator normalization parameters (Equation 3.21) and HSD thresholds (illustrated in Figure 4.4) were tuned by the researcher during this period. Total pretest time was approximately 10 minutes. Once the user acclimated to the prototype device, visit 1 was concluded.

Prototype PAFP Walking Trials

For visit 2, Vicon reflective markers were placed and experimental walking trials with the prototype PAFP were collected. Before data were logged for each walking trial k (symmetry learning iteration), PCA was performed on the IMU roll and pitch angle signals in order to learn the linear transformation that provides a sagittal plane thigh angle estimate (see Section 4.6). After this, excluding iteration $k = 0$ where the PAFP was set

Figure 5.4: Modified plug-in gait reflective marker set placed on subject A01.

to passive mode ($u = 0$), the control scaling factor was increased incrementally over a span of $30 - 60$ seconds (Equation 4.2). After the subject had acclimated to the control signal, data were collected for 30 seconds. Once data were collected, the control scaling factor and treadmill speed ramped down to zero, thus completing the walking trial. After each walking trial, the subject was allowed to rest while the data were processed, the ILC update law was executed, and the new virtual trajectory was uploaded to the embedded system. Each walking trial, including control signal acclimation, data collection, processing and learning took approximately $10 - 15$ minutes. This process was repeated until the measured ILC error signal could no longer be reduced, i.e., the learning signal converged.

Data Processing & Learning

The data processing pipeline is outlined in Figure 5.5 and the MATLAB scripts can be found in Appendix B. Motion and GRF data were captured by the Vicon MoCap system

at 120 Hz and 1200 Hz respectively. Sensor and control signals were sampled by the embedded system at 500 Hz and synchronized with the Vicon data through the use of a sync signal from the Vicon system. The MoCap and GRF signals were processed with a standard inverse dynamic gait model (plug-in gait model, Vicon), which also used the anthropometric data of the subject, resulting in joint mechanics. Then, joint mechanics and GRFs were low-pass filtered with a zero-phase fourth order Butterworth filter with cutoff frequency of 6 and 50 Hz, respectively. Additionally, the sampled true motor current and motor velocity signals were low-pass filtered with a cutoff frequency of 50 Hz.

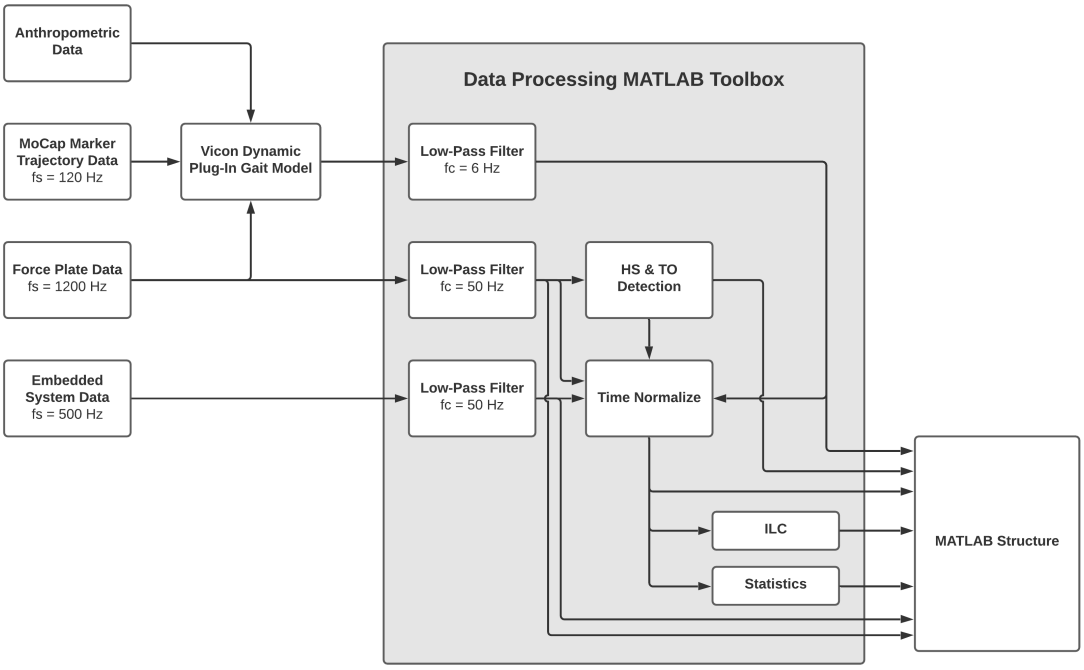


Figure 5.5: Data processing pipeline. The outputs are time series vectors of joint kinematics, joint kinetics, marker trajectories, GRF, sensor, and control data. In addition, time series vectors are also time-normalized for analysis.

HS and TO events were identified post-hoc from the filtered GRF data as follows: HSs were defined as the frames that transitioned to a GRF magnitude greater than 50 N; TOs were defined as the frames that transitioned to GRF magnitude of less than 50 N. All data were then time-normalized, via interpolation, to both 100% of gait and 100% of

stance for statistical analysis. The mean time-normalized intact ankle angles and mean time-normalized prosthetic ankle angles were used to compute the error signal for the ILC algorithm described in Chapter 3. Note that the error signal was set to zero during 70-100% gait so the learning algorithm could better adapt and make improvements to the virtual trajectory during phases where the controller is active (i.e., stance phase). By using this technique, the swing phase errors did not directly influence the learned signal.

5.2.5 Data Analysis

Note that data were collected for two load carriage conditions (backpack and frontpack), however, these results will not be analyzed in this study. The Vicon marker set does not directly apply to the load carriage conditions since the load carrier obstructs the MoCap system's view of crucial markers. Further work must be conducted to compute lower-body joint kinematics and inverse dynamics using custom biomechanical models developed in Visual 3D software (C-Motion, Inc.) for the load-carrying conditions. For this study, lower-limb joint kinematics and kinetics were processed in Visual 3D for each of the subject's unloaded conditions. Visual 3D software is able to utilize custom biomechanics models and virtual marker labeling, which results in more accurate estimates of joint mechanics when compared to using Vicon software alone. Each unloaded condition contained data from three different trials: walking with the (i) prescribed device, (ii) prototype device in its passive mode, and (iii) prototype device in its active mode. The processed data were then exported to MATLAB for further analysis and figure generation. The results from these three experimental conditions are presented in Section 5.3 and discussed in Section 5.4. Mean peak biomechanical outcomes and temporal information across subjects are compiled in Table F.1 and mean asymmetry outcomes across subjects are recorded in Table F.2. Statistical information is also included within these tables and relevant statistics are summarized in Section 5.3.

Quantifying Asymmetry

In this work, for each gait variable (e.g., ankle kinematics, torque, and power), each individual step is assigned an asymmetry measurement (AM) by computing the average norm difference of the step of interest and the steps of the opposite limb occurring immediately before and after the step of interest. The AM for step s of some gait variable X can be

computed as

$$AM(X_s) = \frac{1}{2}\|X_s - Y^-\|_2 + \frac{1}{2}\|X_s - Y^+\|_2 \tag{5.7}$$

where X is the time series data of the gait variable of interest (e.g., prosthetic ankle angles) over step s and Y is the time series data of the opposite limb (e.g., intact ankle angles). The "$-$" and "$+$" superscripts of variable Y indicate the vectors of data for the steps before and after step s respectively. The 2-norm is defined as $\|\cdot\|_2 = \sqrt{(\cdot)^2}$. Note that this process is applied to each step within each walking trial and the result is a distribution of AMs for each trial. Therefore, it is possible to conduct statistical analysis comparing the AM distributions between learning trials and experimental conditions.

Statistical Methods

Linear regression was used to assess the association between outcomes and condition averaged over study participant. Each outcome (e.g., peaks and temporal information) was the dependent variable while the conditions (e.g., prescribed, passive, and active), limbs (e.g., prosthetic vs intact limb), study subject ID (e.g., A01 and A02), and condition by foot interaction were the independent variables. Condition was modeled using two dummy variables. The interaction terms allowed for separate estimates of the association between outcome and condition by limb in the same model. The presence of the subject ID variable allows separate intercepts to be estimated for each participant. Similarly, linear regression was used to assess the association between AM outcomes and condition averaged over limb and study subject ID, using the model described above, but excluding the interaction term.

Hypothesis tests to detect an overall association between outcome and condition were carried out using F-tests. The Benjamini-Hochberg correction was applied to the p-values to maintain a false discovery rate of 5%. Hypothesis testing to detect pair-wise differences among the three condition categories were carried out using t-tests, with Tukey's method for comparing three sets of differences. The statistical significance criterion was $p < 0.05$. Results are presented as means and standard errors for each outcome by condition and mean pairwise differences among conditions with standard errors (SEs) and 95% confidence intervals. Statistical analysis was carried out using R 4.0.3 [197], and packages tidyverse

[198], emmeans [199], and kableExtra [200].

5.3 Results

5.3.1 Controller and Learning Algorithm Performance

The learning algorithm produced bounded control signals for both subjects and all iterations k despite human adaptations. This can be seen in Figures 5.6 and 5.7, which show time domain signal traces at each learning iteration k for the learned virtual trajectory $\hat{\theta}_{v,k}$ (top plot), ILC error $\bar{e}_k$ (middle plot), and the change in reference signal, i.e, $\bar{\theta}_{b,k} - \bar{\theta}_{b,0}$ (bottom plot). Each time domain signal is time-normalized with respect to the gait cycle and averaged across gait cycles for each iteration k. Note the color of the traces corresponds to each iteration is a gradient from black (passive trial) to pink (final learning iteration). The signal of the best-performing active trial is highlighted with a greater line thickness.

The differences in the virtual trajectory across the two subjects is very apparent, which supports the concept that personalization is important for powered prosthesis control. This is further supported when examining the differences in the error signals across the two subjects. Also, each subject adapted to the controller very differently. Subject A02's reference signal noticeably changed only at the first learning iteration (i.e., $k = 1$). Changes past the first learning iteration were very small. Comparatively, subject A01 adapted their reference signal at the first learning iteration and also much later into the experiment. This can be understood from looking at the bottom plot of Figure 5.6. The reference signal appears to be converging up to $k = 3$, however, it starts to shift again at $k = 4$. In addition, note the growth in the error signal e_k at late stance, i.e., near 60% gait. The levels of adaptation across the two subjects further highlight the need for control system personalization and properly adapting the active assistance based on changes in the user's behavior.

Despite the variations in reference signal and increases in the ILC error during some of the later iterations, the learning algorithm was able to avoid divergence of the virtual trajectory and improve overall performance when compared to the passive trial. These results suggest that the adaptive learning gain ρ_k reduced in response to increasing ILC error $\bar{e}_k$ or increasing changes in reference signal $\bar{\theta}_{b,k}$. The lowest achieved mean error $\bar{e}_k$ occurred at $k = 3$ for subject A01 and $k = 2$ for subject A02. Therefore, these iterations will be used in subsequent analysis and will be denoted as the active trials.

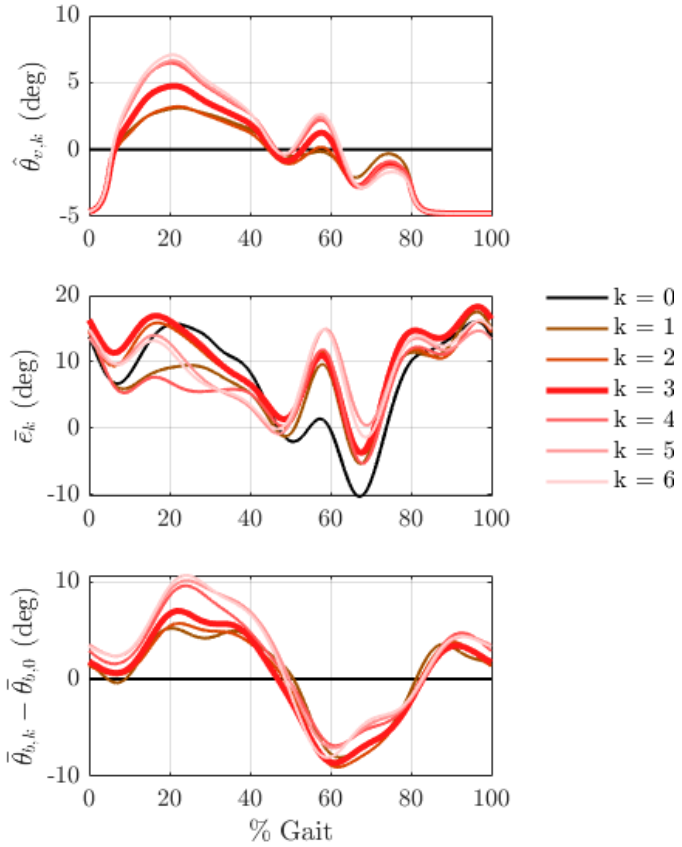


Figure 5.6: Time domain evolution of the learned virtual trajectory $\hat{\theta}_{v,k}$, ILC error $\bar{e}_k$, and the change in reference signal, i.e, $\bar{\theta}_{b,k} - \bar{\theta}_{b,0}$ during each iteration k for subject A01.

5.3.2 Ankle Mechanics

Both the passive and active conditions had a significant effect on the peak ankle plantarflexion and dorsiflexion angles of the prosthetic limb when compared to the prescribed condition. The mean ankle angles for each condition are shown in the top row of Figures 5.8 (subject A01) and 5.9 (subject A02). Additionally, the active condition significantly reduced the peak prosthetic dorsiflexion angle when compared to the passive condition ($p < 0.000$). This suggest that the algorithm learned to support the users throughout mid-stance. There was no significant change in peak prosthetic plantarflexion angle between the passive and active conditions but this could be attributed to the different trends in each subject. For subject A01, the active condition reduced the peak prosthetic plantarflexion angle. Qualitatively, it

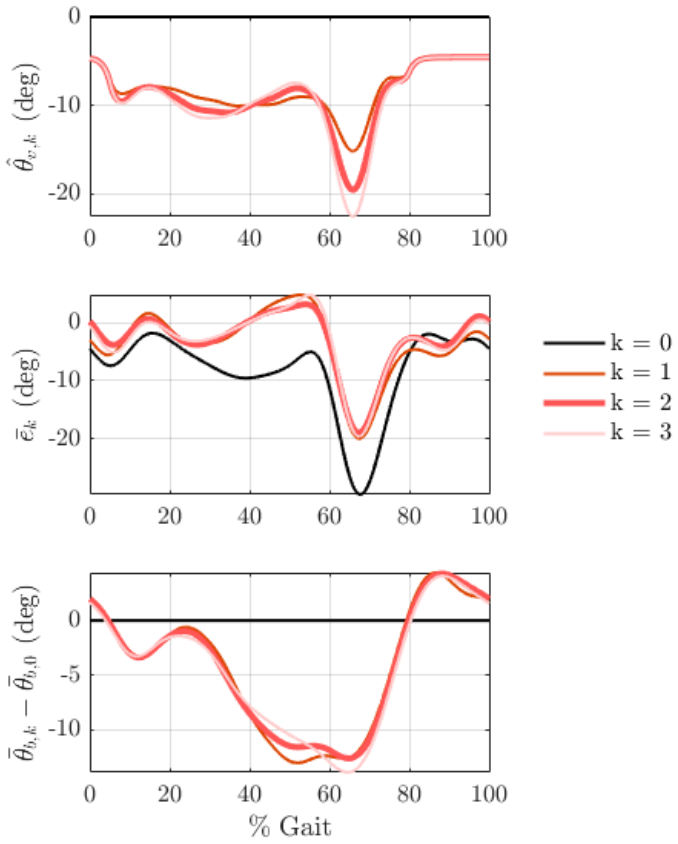


Figure 5.7: Time domain evolution of the learned virtual trajectory $\hat{\theta}_{v,k}$, ILC error $\bar{e}_k$, and the change in reference signal, i.e, $\bar{\theta}_{b,k} - \bar{\theta}_{b,0}$ during each iteration k for subject A02.

is evident that the prosthetic ankle is actively plantarflexed near 70% gait during the active condition for subject A02 (see Figure 5.9). A possible reason for the discrepancy between the subjects' peak plantarflexion outcomes is that subject A02 received a much higher peak motor torque near push-off. Another reason could be the inconsistent angular kinematics across limbs and subjects, which is discussed further in Section 5.4.4. Peak dorsiflexion on the intact side was only significantly different between the prescribed and active conditions ($p = 0.031$). However, this significance is likely motivated in large part by subject A02, who had a relatively large difference between these conditions. Note that increased dorsiflexion of the intact ankle is common for individuals with below-knee amputation [6]. However, subject A02 achieved their lowest peak intact-side dorsiflexion during the active condition. Interest-

ingly, the active condition also significantly increased the intact limb's peak plantarflexion angle when compared to the passive condition ($p = 0.026$) and this trend was consistent across both subjects. This finding supports the notion that the behavior of the limbs are coupled, therefore, modifying and improving the behavior of the prosthetic limb can benefit the biomechanics of the intact limb. The ankle angle asymmetry was significantly different across all conditions, with the active condition having lowest ankle angle AM value (i.e., the active condition achieved the best ankle angle symmetry). All peak ankle angle outcomes are shown in Figures G.3 and G.4. Additionally, all ankle ankle asymmetry measurement outcomes are shown in Figure G.22.

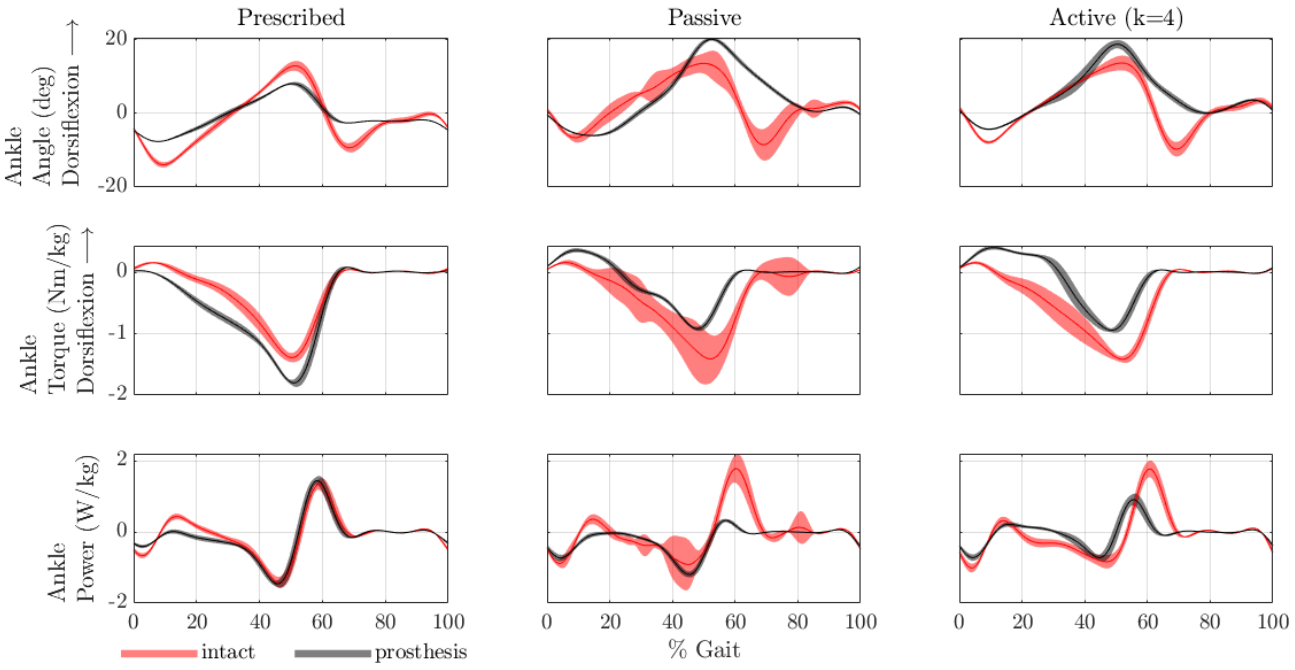


Figure 5.8: Sagittal plane ankle mechanics for subject A01 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

The peak prosthetic ankle moment was significantly different across all conditions. The mean ankle moments for each condition are shown in the middle row of Figures 5.8 (subject A01) and 5.9 (subject A02). The prescribed condition had the highest peak prosthetic ankle moment with a mean of 1.55 Nm/kg. The peak prosthetic ankle moment significantly

increased from the passive condition to the active condition for both subjects ($p = 0.024$). Even though the active condition increased peak prosthetic ankle moment, the peak moment occurred earlier in the gait cycle when compared to the intact side for both subjects (see Figures 5.8 and 5.9). It is possible that the subjects were not receiving the peak active torque at push-off and were, therefore, not receiving the full benefit of the active assistance. With additional learning trials, the learning algorithm may have been able to phase-shift the virtual trajectory and accommodate for this timing discrepancy. For the intact limb, the ankle moment was only significantly different between the prescribed and passive conditions ($p = 0.021$). However, this trend is likely dominated by subject A01, who had a larger increase in peak intact ankle moments from the prescribed to passive conditions. Also note that the subjects had opposing trends when comparing the peak intact ankle moments from the passive to active conditions. Subject A02's peak intact ankle moment increased from passive to active conditions while subject A01's peak intact ankle moment reduced. Similar to what was observed with peak intact-side ankle plantarflexion, subject A02 was likely able to utilize the active assistance to a greater degree. Ankle moment asymmetry significantly increased from the prescribed condition to both the passive and active conditions ($p < 0.000$ for both). The difference in asymmetry between the passive and active conditions was not statistically significant, however, it should be noted that each subject displayed opposing trends. Subject A02's ankle moment asymmetry decreased from passive to active conditions while subject A01's increased. Subject A02 achieved very similar ankle moment symmetry between their prescribed and active conditions. These opposing trends between the subject's passive and active conditions are consistent when looking at support moment asymmetry, i.e., the sum of the sagittal plane extensor moments of the three joints of the lower limb. However, support moment asymmetry was significantly reduced during these two conditions when compared to the prescribed condition ($p < 0.000$). All peak ankle moment outcomes are shown in Figure G.7. Additionally, all ankle moment and support moment asymmetry measurement outcomes are shown in Figures G.23 and G.21 respectively.

Peak positive and negative prosthetic ankle power was significantly altered across all three conditions. The mean ankle powers for each condition are shown in the bottom row of Figures 5.8 (subject A01) and 5.9 (subject A02). Peak positive prosthetic ankle power increased by 0.52 W/kg on average from the passive to active condition ($p < 0.000$). Negative prosthetic ankle power also significantly decreased by an average of 0.26 W/kg between these

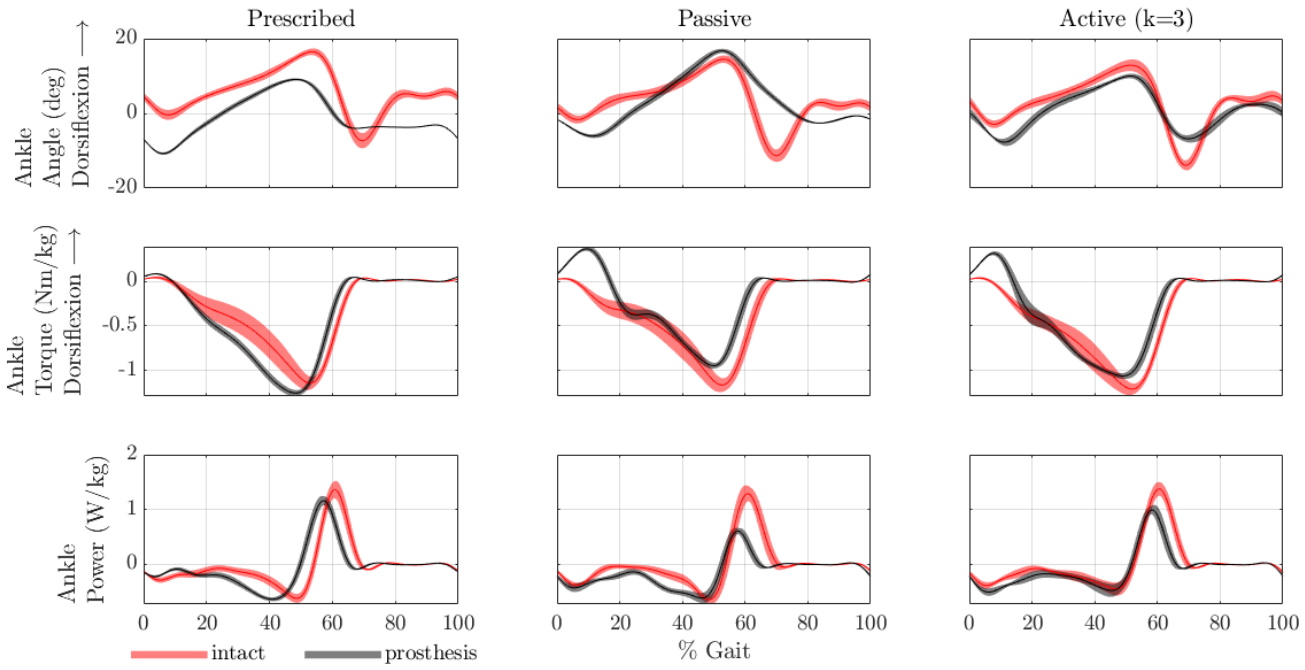


Figure 5.9: Sagittal plane ankle mechanics for subject A02 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

two conditions ($p < 0.000$). Peak positive and negative intact ankle power was significantly different for the prescribed condition when compared to the passive and active conditions. However, this trend is likely dominated by subject A01. Subject A02's peak positive and negative intact ankle power did not differ greatly across all three conditions. Ankle power asymmetry was significantly lower for the prescribed condition when compared to the passive condition ($p < 0.000$) but not when compared to the active condition ($p = 0.080$). The active condition significantly reduced ankle power asymmetry when compared to the passive condition ($p = 0.033$). Subject A02's active condition achieved the lowest mean ankle power AM while subject A01's prescribed condition had the lowest AM. These results indicate that powered assistance has the potential to improve prosthetic ankle power (i.e., increases positive power and decreases negative power) and ankle power symmetry. All peak ankle power outcomes are shown in Figures G.13 and G.14. Additionally, all ankle power asymmetry measurement outcomes are shown in Figure G.24.

5.3.3 Knee Mechanics

The mean knee angles of each subject for each condition are shown in the top row of Figures 5.10 and 5.11. Peak prosthetic-side knee flexion was significantly different across all three experimental conditions. With each subsequent condition, prosthetic-side knee flexion increased significantly ($p < 0.000$), with the lowest outcome observed during the prescribed condition and the highest during the active condition. A similar trend was observed for subject A01's peak intact-side knee flexion angles, however, these peaks did not differ greatly across conditions for subject A02. Prior research has shown that individuals with below-knee amputation have increased knee flexion about both limbs [6]. An interesting observation in this study was that knee angle asymmetry during the active condition was significantly reduced when compared to both the prescribed and passive conditions ($p = 0.045$ and $p = 0.039$ respectively). However, this trend is likely dominated by subject A02, who greatly reduced their knee angle asymmetry during the active condition. Knee angle asymmetry was not significantly different between the prescribed and passive conditions. Qualitatively, it can be seen that the peak knee flexion angles between the two limbs for both subjects appear closer in magnitude during the active condition (top right plot of Figures 5.10 and 5.11). Additionally, subject A02's prosthetic-side knee angles overlap with the intact-side knee angles during the the early and mid-stance phases of the active condition, which is not observed in the other two conditions. On the other hand, peak knee flexion timing appears noticeably earlier in gait for each of the subject's passive and active conditions when compared to their prescribed conditions. All peak knee flexion angle outcomes are shown in Figure G.5. Additionally, all knee angle asymmetry measurement outcomes are shown in Figure G.25.

The mean knee moments of each subject for each condition are shown in the middle row of Figures 5.10 and 5.11. Peak prosthetic-side knee flexion moments were not significantly altered across experimental conditions. Peak intact-side flexion moments were only significantly different between the prescribed and active conditions ($p = 0.0058$), however, this trend is likely dominated by subject A01. Peak prosthetic-side knee extension moments were significantly different across experimental conditions but this was likely dominated by subject A01. For subject A01, peak prosthetic-side knee extension moment was reduced for the passive condition when compared to the prescribed condition. The peak prosthetic-side knee extension moment increased for the active condition when compared to the passive

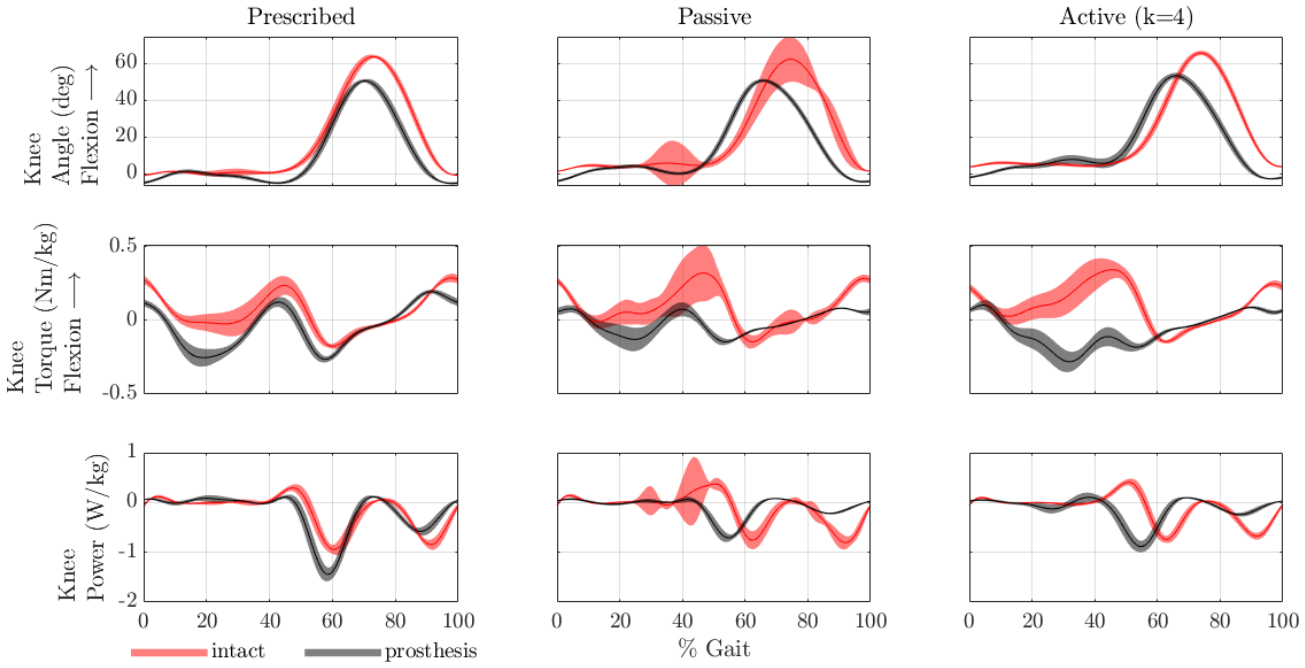


Figure 5.10: Sagittal plane knee mechanics for subject A01 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

condition. The differences between conditions were less pronounced for subject A02's experiment, with the lowest average peak prosthetic-side knee extension moment observed during the active condition. The peak intact-side knee extension moment was significantly reduced from the passive to the active condition ($p < 0.000$). This trend was consistent across both subjects. The prescribed condition was also significantly different when compared to the passive condition ($p < 0.000$), but this trend was likely dominated by subject A02, who had a peak intact-side knee extension moment nearly twice the magnitude compared to the peak intact-side knee extension moment of the prescribed condition. Knee moment asymmetry was significantly altered between experimental conditions, however, opposite trends are observed between each subject. For subject A02, knee moment asymmetry was reduced from the prescribed to passive condition and reduced further to the active condition. An increase in knee moment asymmetry is observed from the prescribed to passive condition and the passive to active condition for subject A01. However, on average, the differences in peak

knee extension moments across limbs were reduced by 0.15 Nm/kg during the active condition when compared to the passive condition. All peak knee moment outcomes are shown in Figures G.8 and G.9. Additionally, all knee moment asymmetry measurement outcomes are shown in Figure G.26.

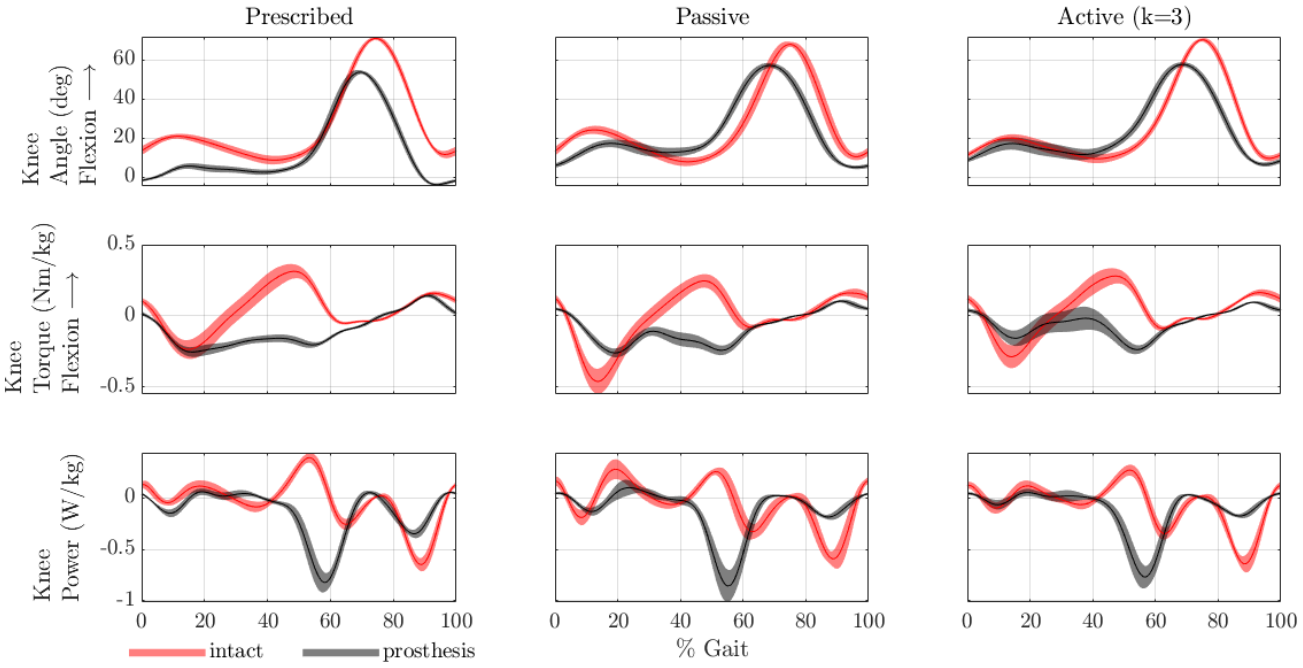


Figure 5.11: Sagittal plane knee mechanics for subject A02 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

The mean knee powers of each subject for each condition are shown in the bottom row of Figures 5.10 and 5.11. Peak positive knee power was not significantly altered across experiment conditions for both limbs. Peak prosthetic-side negative knee power was significantly different when comparing the prescribed condition to the passive and active conditions ($p < 0.000$ for both). However, this trend is likely dominated by subject A01 since their peak prosthetic-side negative knee power was much larger in magnitude for the prescribed condition when compared to the other two conditions. Peak negative knee power was not significantly altered for the intact limb across experimental conditions. However, it can be observed that subject A01's peak intact-side negative knee power reduced from the prescribed

to the passive condition and reduced even further when compared to the active condition (see Figure 5.10). On average, knee power asymmetry significantly increased during the passive condition when compared to the prescribed condition ($p = 0.0053$). Furthermore, knee power asymmetry increased during the active condition when compared to the passive condition ($p = 0.0023$). Both trends in knee power asymmetry were likely influenced by subject A01 to a large degree. Qualitatively, it can be seen that subject A02's knee power asymmetry was not affected across experimental conditions. Furthermore, the largest knee power AM value for subject A01 is observed during the active condition but for subject A02, the active condition had the lowest AM value. All peak knee power outcomes are shown in Figures G.15 and G.16. Additionally, all knee power asymmetry measurement outcomes are shown in Figure G.27.

5.3.4 Hip Mechanics

The mean hip angles of each subject for each condition are shown in the top row of Figures 5.12 and 5.13. The peak prosthetic-side hip extension angle was significantly reduced for the passive and active conditions when compared to the prescribed condition ($p < 0.000$ for both). The differences in peak prosthetic-side hip extension for the passive and active conditions were not statistically significant. The peak intact-side hip extension angle of the passive condition increased significantly when compared to the prescribed condition ($p = 0.012$). This outcome then reduced during the active condition when compared to the passive condition ($p = 0.0042$). However, the trends between conditions for intact-side peak hip extension is likely dominated by subject A01 since subject A02's peak extension varied less across conditions. The active condition significantly increased hip angle asymmetry when compared to both the prescribed and passive conditions ($p < 0.000$ for both). These trends were more pronounced in subject A02 relative to subject A01. The peak hip angles for subject A01 are closer in magnitude during the active condition when compared to the prescribed condition (top left and right plots of Figures 5.12 and 5.13). The intact-side hip angles stayed fairly consistent across conditions for subject A02, however, their prosthetic-side hip angles during passive and active conditions increased overall toward flexion (top middle and right plots of Figure 5.13). This is likely a cause for subject A02's higher hip angle AM values during these two conditions when compared to the prescribed condition. All hip extension angle outcomes are shown in Figure G.6. Additionally, all hip angle asymmetry

measurement outcomes are shown in Figure G.28.

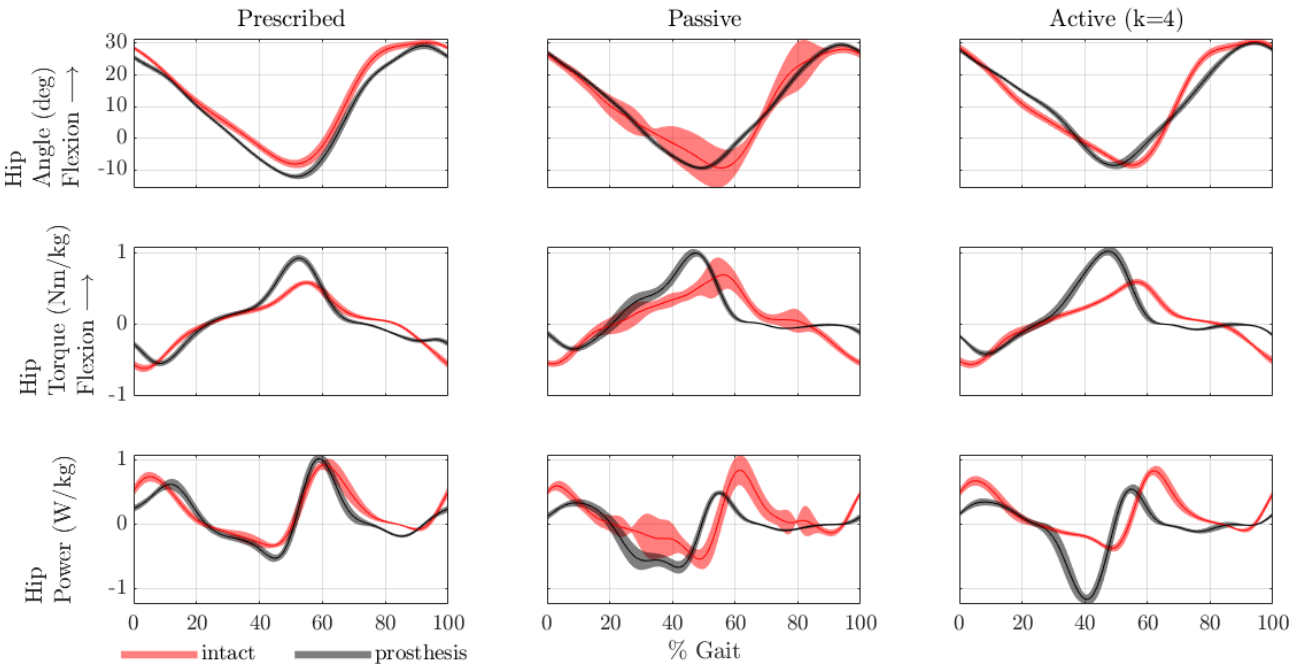


Figure 5.12: Sagittal plane hip mechanics for subject A01 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

The mean hip moments of each subject for each condition are shown in the middle row of Figures 5.12 and 5.13. The peak prosthetic-side hip flexion moment was not significantly altered across experimental conditions on average. Each subject displayed noticeable changes across conditions, however, their trends were opposite. The difference between the prescribed and passive conditions when observing the peak intact-side hip flexion moment was significant but it was much more apparent for subject A01. The peak intact-side hip flexion moment significantly increased for subject A01's passive condition when compared to the prescribed condition. For subject A02, there was very little change in peak intact-side hip flexion moment between the prescribed and passive conditions. The peak intact-side hip flexion moment was significantly reduced for the active condition when compared to the passive condition ($p < 0.000$). Differences in peak hip extensions moments were not significant across conditions and limbs. Subject A01's prosthetic-side hip extension moments decreased for

both the passive and active conditions when compared to the prescribed condition (see Figure 5.12). On the other hand, prosthetic-side hip extension moments for subject A02 increased for both the passive and active conditions when compared to the prescribed condition (see Figure 5.13). Hip moment asymmetry comparisons reached significance across all conditions ($p < 0.000$ for all). For each subject, the highest hip moment AM was observed during the active condition while the lowest hip moment AM was observed during the prescribed condition. All peak hip moment outcomes are shown in Figures G.10 and G.11. Additionally, all hip moment asymmetry measurement outcomes are shown in Figure G.29.

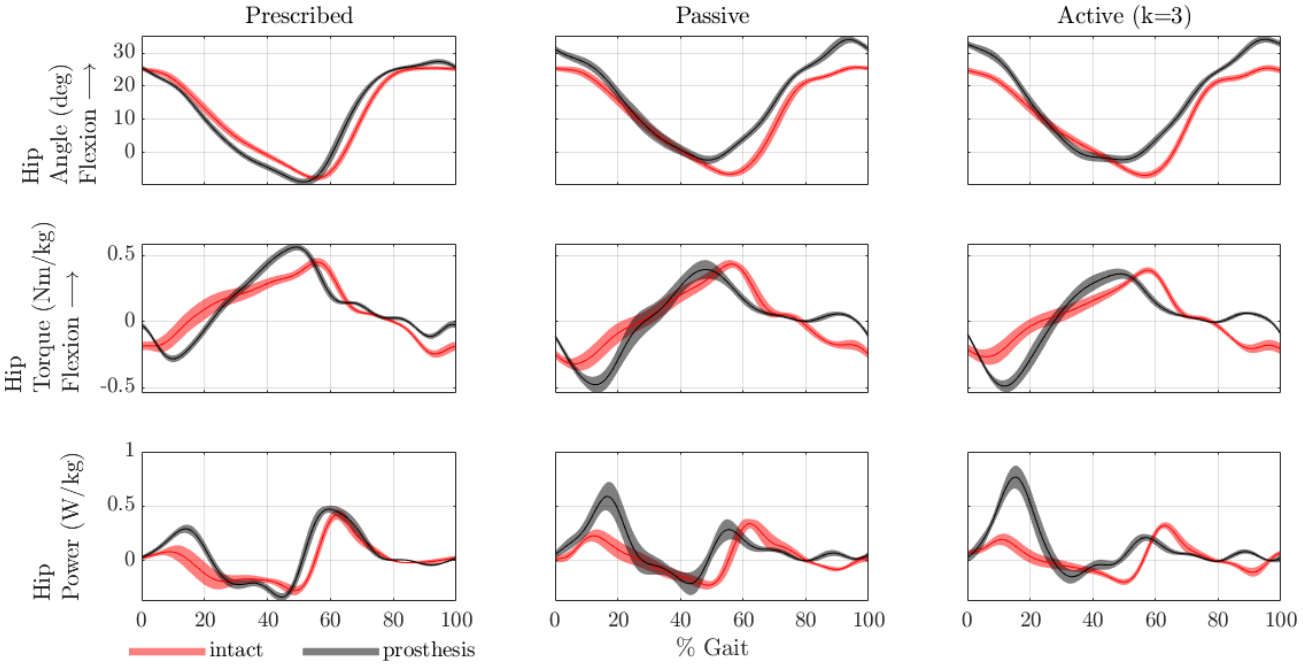


Figure 5.13: Sagittal plane hip mechanics for subject A02 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

The mean hip powers of each subject for each condition are shown in the bottom row of Figures 5.12 and 5.13. Peak prosthetic-side positive hip power was significantly different across all conditions. On average, the peak prosthetic-side positive hip power was higher for the prescribed condition, however, this was likely dominated by subject A01. Subject A02's lowest prosthetic-side positive hip power was during the prescribed condition. Qual-

itatively, it can be seen that the peak positive hip power for the prosthetic side noticeably increased during the active condition for subject A02 when compared their prescribed condition (bottom left and right plots of Figure 5.13). For both subjects, the peak prosthetic-side positive hip power during the active condition significantly increased when compared to the passive condition ($p = 0.012$). Peak intact-side positive hip power for the active condition was significantly reduced when compared to the prescribed condition ($p = 0.019$). The peak prosthetic-side negative hip power during the active condition was significantly altered when compared to the other two conditions ($p < 0.000$ for both). However, this is likely due to the large increase in negative hip power during the active condition for subject A01 when compared to their other two conditions (see Figure 5.12). Peak intact-side negative hip power significantly decreased during the active condition when compared to the passive condition ($p = 0.0013$). Similar to hip asymmetry, hip power asymmetry differences reached significance across all experimental conditions ($p < 0.000$). Furthermore, for each subject, the highest hip power AM was observed during the active condition while the lowest hip power AM was observed during the prescribed condition. All peak hip power outcomes are shown in Figures G.17 and G.18. Additionally, all hip power asymmetry measurement outcomes are shown in Figure G.30.

5.3.5 OA Loading Factors: KAM and HAM

Figures 5.14 and 5.15 show mean KAM (top row) and HAM (bottom row) for each subject during the three experimental conditions. The mean intact peak KAM was significantly increased from 0.422 Nm/kg during the prescribed condition to 0.477 Nm/kg during the passive condition ($p < 0.000$). However, the active condition was able to reduce the mean intact peak KAM down to 0.415 Nm/kg ($p < 0.000$ when compared to the passive condition). Interestingly, the mean prosthetic-side peak KAM during the active condition was also significantly reduced when compared to the prescribed condition ($p = 0.0048$). Differences in peak HAM were statistically significant across both limbs and all three conditions. The mean intact peak HAM significantly decreased from 0.623 Nm/kg during the prescribed condition to 0.537 Nm/kg during passive condition ($p < 0.000$). Despite this reduction, the active condition achieved the lowest peak intact-side HAM. The mean peak intact-side HAM was 0.448 Nm/kg during the active condition, which was a significant reduction compared to the other two conditions ($p < 0.000$ for both). Additionally, the peak prosthetic-side HAM was

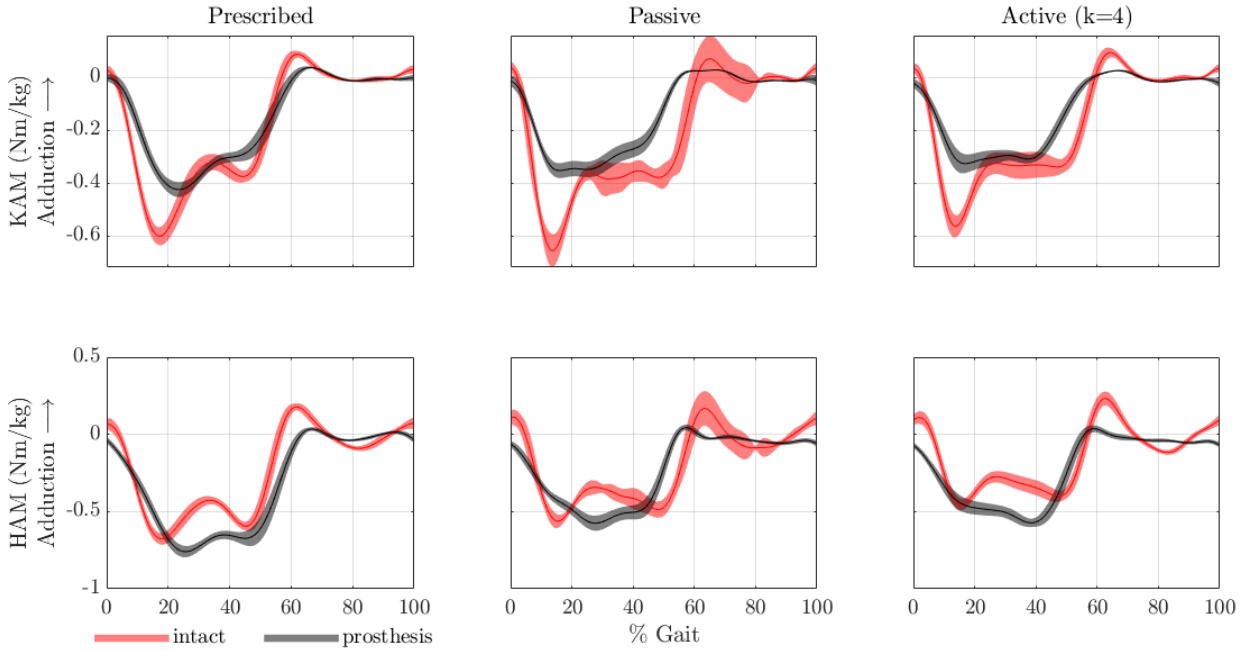


Figure 5.14: Frontal plane knee and hip abduction moments (KAM and HAM) for subject A01 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

reduced for the active and passive conditions when compared to the prescribed condition ($p < 0.000$ for both). The significant reduction in peak KAM and HAM, which are common load factors associated with osteoarthritis, illustrate the clinical benefits of PAFPs and the proposed control method. All peak KAM and HAM outcomes are shown in Figures G.1 and G.2 respectively.

5.3.6 Vertical Ground Reaction Forces

The mean vGRFs of each subject for each condition are shown in Figures 5.16 and 5.17. The mean peak intact vGRF was significantly increased from 1.019 BW for the prescribed condition to 1.171 BW for the passive condition ($p < 0.000$). However, the peak intact vGRF was significantly reduced to 1.097 BW during the active condition ($p < 0.000$ when compared to the passive condition). In addition, the active condition significantly reduced vGRF asymmetry when compared to the passive condition ($p < 0.000$). These results suggest

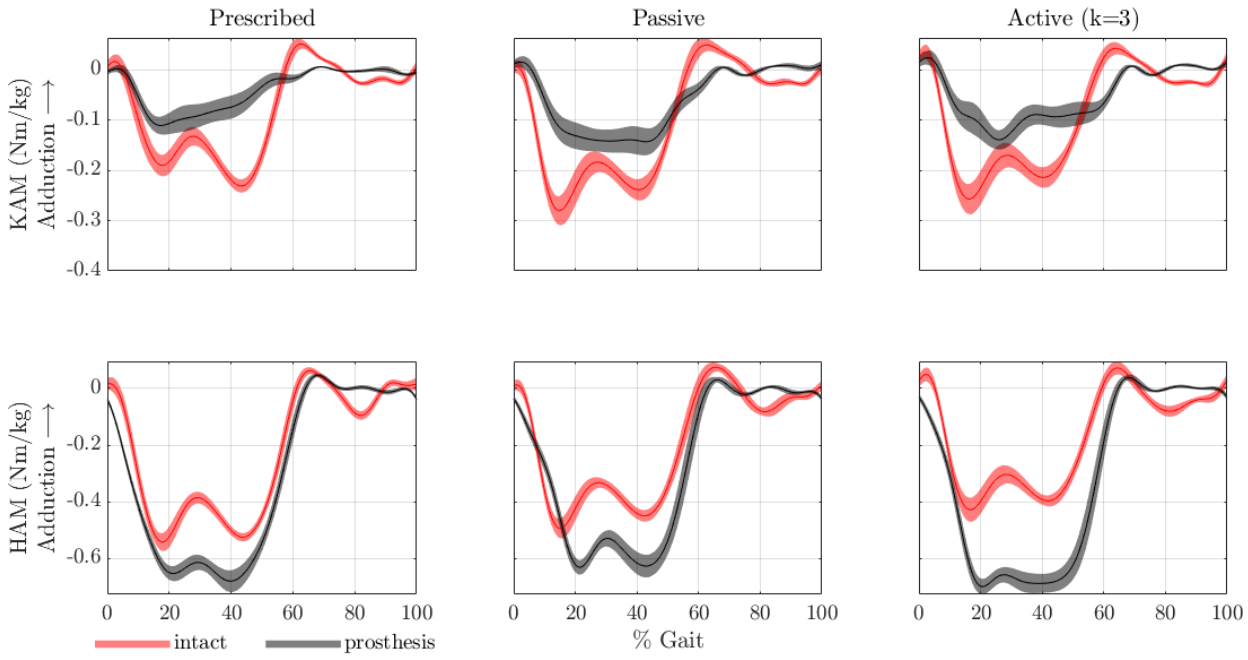


Figure 5.15: Frontal plane knee and hip abduction moments (KAM and HAM) for subject A02 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

that active assistance and the proposed controller properly are beneficial and can reduce the vertical loads on each limb while walking. Neither the passive or active condition were able to achieve the same level of vGRF symmetry as the prescribed condition ($p < 0.000$ for both). However, this is likely due to the increased time spent on the intact limb during these two conditions compared to the prescribed condition, particularly during subject A01's experiment (see Figure 5.16). AM values between the prescribed and active conditions were much more similar during subject A02's experiment. All peak vGRF outcomes are shown in Figure G.12. Additionally, all vGRF asymmetry measurement outcomes are shown in Figure G.31.

5.3.7 Temporal Symmetry

Changes in step period were only significant when comparing the prescribed condition to the passive and active conditions. This was likely because subject A01's treadmill speed needed

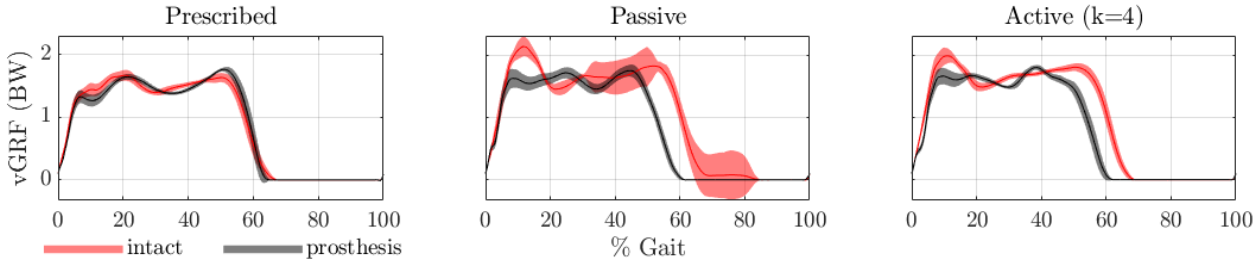


Figure 5.16: Vertical ground reaction force (vGRF), normalized by body weight, for subject A01 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

to be readjusted to 0.9 mps during the passive and active conditions for their safety and comfort. The treadmill speed during subject A01's prescribed condition was set to 1.0 mps. The prosthetic-side period of stance changed significantly when comparing the prescribed, passive, and active conditions. The period of stance on the prosthetic side decreased during the passive condition when compared to the prescribed condition ($p < 0.000$). Simultaneously, the stance phase of the intact limb increased ($p < 0.000$). The prosthetic-side period of stance increased during the active condition when compared to the passive condition ($p = 0.0096$), however, not to the levels observed during the prescribed condition. Also, the period of stance of the intact limb did not change significantly during the active condition compared to the passive condition ($p = 0.48$). All step period and stance outcomes are shown in Figures G.19 and G.20 respectively.

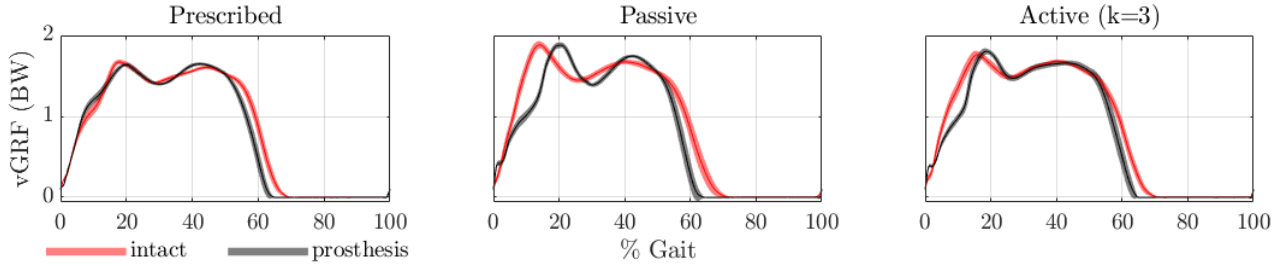


Figure 5.17: Vertical ground reaction force (vGRF), normalized by body weight, for subject A02 during the three experimental conditions. Width of the traces represent ±1 standard deviation.

5.4 Discussion

5.4.1 Biomechanics of the Lower Limbs

The hypotheses were: the proposed control method would result in (1) a reduction in ankle kinematics and kinetic asymmetries, and (2) a reduction in peak intact KAM and HAM. Additionally, knee biomechanics, hip biomechanics, vGRF, and temporal outcomes were analyzed. Hypothesis (1) was partially supported: asymmetries in ankle kinematics and power were reduced when powered assistance was introduced. Even though ankle moment asymmetry was not significantly altered when comparing the passive and active conditions, prosthetic-side peak moments significantly increased. Hypothesis (2) was supported with large reductions in KAM and HAM on the intact limb when comparing the active condition to both the passive and prescribed conditions.

Secondary outcomes also indicate some significant adaptations at the intact hip and knee during active assistance. Most notably, the controller resulted in: (i) a reduction in peak intact hip extension, (ii) a reduction in peak intact knee extension moment, (iii) a reduction in knee angle asymmetry, (iv) a reduction in peak intact hip flexion moment, and (v) a reduction in peak intact negative hip power. The controller also significantly reduced peak intact-side vGRF, significantly reduced vGRF asymmetries, and significantly increased the stance time of the prosthetic limb. This research shows that all lower-limb joint biomechanics are affected by PAFP control, even when only targeting ankle angle asymmetries. This finding suggests that lower-limb joints are strongly coupled and therefore, improving ankle function via powered assistance can reduce compensations at the hip and knee, as indicated by the results.

5.4.2 The Effect of Human Adaptions, Acclimation, and Fatigue

The intact ankle position adapted significantly during learning, particularly for subject A01's experiment. It is possible that not enough time was given to rest between walking trials. Subject A01 experienced some swelling and pain in the residual limb due to the demands of the experimental protocol, which could have affected the results at later learning iterations. Additionally, it is possible that not enough time was given to acclimate to the learned signal at each iteration before collecting data. For these reasons, the current experimental protocol may not be the most conducive to improving gait patterns via powered assistance. A

long-term protocol, where more learning iterations are collected over the course of multiple sessions, would allow more time for the user to acclimate to the controller. This would also assist in convergence of the learned signal. It could also be beneficial to have multiple learning protocols in order to tune and analyze the effects of using different controller and learning algorithm parameters.

5.4.3 Experimental Limitations

One limitation of the current ILC algorithm is that an inverse model was not able to be used since training an inverse transfer function with a limited training dataset resulted in unstable signals (see Section 3.6.3). Once more data is collected or a higher-fidelity model architecture is developed, including the inverse model in the ILC update law could help the learning algorithm adapt to large variations in the reference signal and assist in convergence. Another limitation of the proposed learning approach is the use of inverse dynamics for estimating ankle angle asymmetry. This is restrictive since learning can only be conducted in a gait laboratory setting. Additionally, current learning protocols are run offline since MoCap data needs to be manually labeled and processed. The use of rigid body biomechanics models to calculate ankle angles is also limiting [201]. Future studies should also include experiments with more than two subjects with below-knee amputation in order to analyze more meaningful biomechanical trends across conditions for this target population. Adding new experimental conditions (e.g., load carriage, stair walking, ramp walking, etc.) as well may show the benefits of the adaptive impedance-based feedback controller to a greater degree.

5.4.4 Algorithm Limitations

Offset errors between the intact and prosthetic ankle angle signals were present, especially for subject A01. This offset could have been due to the structural differences between the prosthetic and intact limbs. Another explanation is that Vicon computed the static offset ankle angles incorrectly. This could explain why the results from Visual 3D show the ankle angles having very little offset error during early stance compared to the Vicon signals used for learning (shown in Figure 5.18). The learning algorithm was not able to account for these offsets and as a result, the learned signal was heavily skewed (i.e., changes in the signal

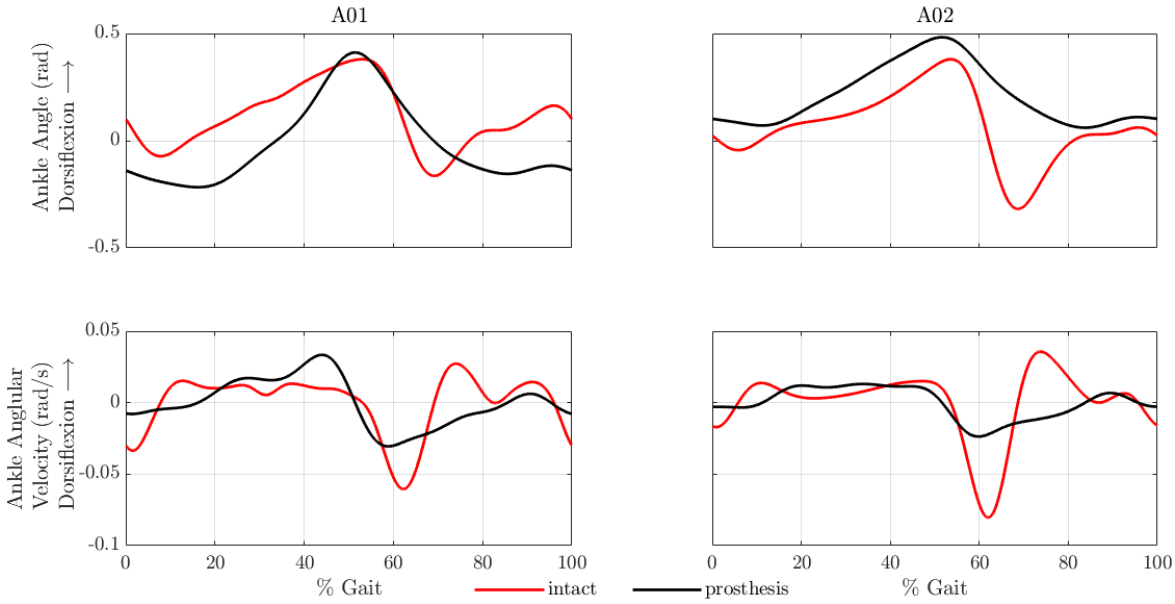


Figure 5.18: Ankle angles (top row) and angular velocities (bottom row) for the two limbs during the passive conditions ($k = 0$) for each subject.

magnitude were large) toward early stance since the magnitude of the error signal was largest during this phase. It is likely that the learner was not able to improve the performance at late stance (i.e., push-off) to the best of its abilities since it was attempting to reduce the offset errors during early stance. A change that could address this limitation would be to conduct learning using the ankle angular velocity signals $\bar{\dot{\theta}}_b$ and $\bar{\dot{\theta}}_p$ rather than ankle positions $\bar{\theta}_b$ and $\bar{\theta}_p$. The learned virtual trajectory could then be numerically integrated and encoded as an impedance (i.e., stiffness) control law just as it was in this study. Ankle angular velocities could be computed for offline learning using a zero-phase low-pass filter. However, if the ILC algorithm were to be modified to enable online learning, differentiating the angular position signals could result in unwanted signal processing issues related to noise.

Figure 5.18 displays the mean angular position and velocities for both limbs during each of the subject's passive condition ($k = 0$). Both subjects show a positional offset between the prosthetic and intact limbs during early stance. What's interesting is that subject A01's offset is plantarflexed (i.e., vertically shifted in negative direction) while subject A02's offset is dorsiflexed (i.e., vertically shifted in positive direction). This analysis suggests that the

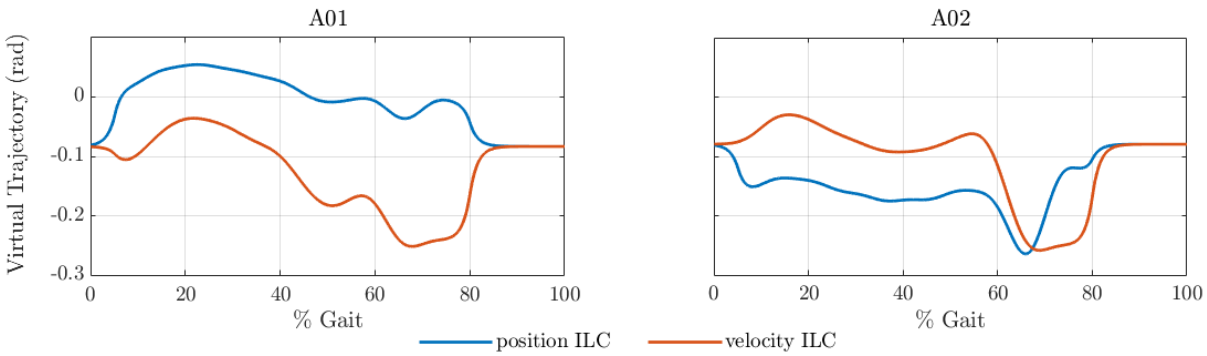


Figure 5.19: Learned virtual trajectories from the experiment's ankle position-based ILC (blue) and a modified ankle velocity-based ILC (orange). Trajectories were learned using data from each subject's passive condition ($k = 0$).

angular position is unique for each subject and limb. Therefore, position signals may not be the best choice for the ILC algorithm. It may be more appropriate to focus the learning algorithm to match the general shape of the prosthetic ankle's kinematics to that of the intact side's kinematics rather than attempt to match their magnitudes. When looking at the angular velocity signals in Figure 5.18, they appear more consistent across the two subjects. For both individuals, the intact limb has a greater velocity in the plantarflexed direction near HS and TO events compared to the prosthetic limb. Alternatively, angular velocities between limbs were more similar for both subjects during single-limb support (i.e., around $20-50\%$ gait) when compared to the ankle angle values during this phase. Therefore, an ILC law that utilises angular velocity signals may generate trajectories that better assist the user during the loading response and push-off phases of gait.

In order to compare the virtual trajectory shapes between the position-based and velocity-based methods, a single learning trial was conducted using the signals shown in Figure 5.18. The learning algorithm parameters were kept consistent between the two ILC methods. Also note that, in order to produce a virtual set-point signal for the impedance control law, the learned signal from the velocity-based ILC method is integrated with an initial condition equal to the mean encoder position at HS events. This trajectory is then filtered as described in Section 4.7. The virtual trajectories are shown in Figure 5.19. The velocity-based ILC signals across subjects have similar shapes while still having personalized peak magnitudes

and timings. For each subject's velocity-based ILC signal, the peak positive value occurs around the end of loading response while the peak negative value occurs near TO. This is consistent with how we would expect the controller to provide assistance at each of these instances. Alternatively, the span of subject A01's position ILC trajectory is entirely above (i.e., more positive than) its starting value at HS. Similarly, subject A02's position ILC trajectory is entirely below (i.e., more negative than) its starting value at HS. These trends correspond to the ankle angle offset for each subject and support the notion that these offsets have too large of an effect on the learner. The velocity ILC method could overcome these issues and allow the learner to match kinematic shapes and patterns while allowing for intrinsic offsets between ankle angles.

For subject A01, there is a noticeable difference between the virtual trajectories of the position and velocity ILC methods. The position-based ILC method attempted to correct the large positional errors during $10-50\%$ gait and as a result, the controller provided a lower motor torque during push-off when compared to subject A02. While we don't expect each peak motor torque to be consistent across individuals, subject A02 was able to achieve much better ankle power symmetry when compared to subject A01. This was likely due to the higher push-off assistance generated by subject A02's virtual impedance trajectory. There are less differences between the two ILC methods for subject A02, which could be explained by the lower offset errors between their two ankle angle signals. Overall, this initial analysis provides some insight into the potential benefits of a velocity-based ILC method. Future work should include human subject experiments to assess how this method could better assist users and adapt to changes in the reference signal.

Chapter 6

DATA-DRIVEN MODELS TO PREDICT PROSTHESIS ANKLE TORQUES FROM INVERSE DYNAMICS USING WEARABLE SENSORS

6.1 Introduction

What would it take to use model-based control to optimize prosthetic torque output? A practical model that maps from easily observed states, such as inertial sensor and load cell outputs, to the desired control commands, such as torque about the prosthetic ankle, would be needed. Unfortunately, it is challenging to observe joint torques from full-body inverse dynamics in a deployed device using marker-based motion capture technology. The large amount of sensors and complex computations from multi-body dynamics do not translate well to the real-time embedded control systems found in robotic prosthetic limbs. This chapter outlines a proposed method that can overcome this limitation. State-of-the-art data-driven machine learning models, which are richer and more powerful than those that have been shown before for this domain, are trained and evaluated based on how well they can predict total prosthetic ankle torque. Additionally, minimal instrumentation is used to provide inputs into the model, meaning that this proposed method presents a benefit over previous proposals by eliminating the need for full-body motion capture suits or a large array of fragile sensors (e.g., FSRs or EMG). Experimental data in the form of total prosthetic ankle torques from optical motion capture (target outputs) and wearable sensor data (predictors) are used to build and demonstrate the capabilities of high-fidelity predictive models for future prosthetic control design.

In robotics applications, modeling the system dynamics can be useful for both (i) planning control actions and (ii) understanding the system better which can reveal more possible future actions and tasks to the robot. A system model can also allow for model-based control methods to be used, which are much more sample-efficient (i.e., narrowing the control input search space, thus lowering the computational cost and the number of necessary evalu-

ations) and converge much quicker when compared to model-free techniques [154, 202–209]. Furthermore, once a model-based control algorithm quickly achieves proficient performance, the controller can switch to a model-free learner such as policy gradients [208, 210] using the model-based law as an initialization. The model-free learner can then fine-tune the policy, overcome model uncertainty, and achieve near-optimal behavior without the expense of higher sample complexity [211, 212].

Model-based control systems use forward predictions to generate actions that drive the system toward a beneficial state. Despite their promise, these model-based control systems have not been implemented in robotic prosthesis control since modeling human-prosthesis dynamical behavior remains a challenge. While any physical system can in theory be represented using Newtonian mechanics (e.g., mass-spring damper models), the models would be too complex to be rendered efficiently on hardware and very expensive to build. For example, these models may include systems of ordinary or partial differential equations which are complicated to solve for high-bandwidth control loops. Also, errors may be introduced through uncertain initial or boundary conditions. Additionally for this application, the human-prosthesis system dynamics are highly stochastic and nonlinear. Therefore, an accurate forward prediction cannot be approximated easily using these first principles models.

Deep neural networks (DNNs) provide a promising alternative and are state-of-the-art in other challenging learning tasks such as natural language processing [213] and computer vision [162] that also involve high-dimensional nonlinear relationships. Neural networks are trained input-output models, capable of fitting any nonlinear function to sufficient accuracy [214]. Rather than relying on a set of manually-derived equations, DNNs are represented as a network of neurons inspired by the function of the human brain. The input-output relationship of a dynamical system is described by a network of weighting factors which can be optimized given a sufficient training dataset. This type of problem is classified as a time series forecasting problem in supervised machine learning [215, 216] and the dynamical system of interest takes the the following form:

$$y_{t+1} = f(x_t, u_t) + \omega(x_t) \tag{6.1}$$

where y is the system response of interest, x is the state of the system (e.g., observable position or forces), u is the control action (e.g., the motor command), ω is the disturbance

(e.g., perturbation, randomness, or unmodeled behavior of the system), and f is the forward model of the system represented as a function. The role of a neural network model is to uncover the underlying function, f, in the data. After the model is sufficiently trained, it can be deployed to a real-time system [217] and map real-time sensor measurements (e.g., IMUs and load cells) to an output prediction (e.g., ankle torque response), all without explicit knowledge of physical concepts or Newtonian mechanics.

Another potential advantage of utilizing DNNs for prosthetic control applications is their capabilities to forecast into the future. In certain cases, modeling approaches based on physics or first principles can provide insight toward immediate input-output relationships. However, these types of models are limited in their abilities to forecast future behavior since there are not always universally-accepted formulas or well-understood natural phenomena that can quantify long-term spatial-temporal relationships within a given system. On the other hand, DNN architectures have demonstrated an ability to forecast behaviors when mathematical process models are not known. Some successful applications include robotic trajectory forecasting [218], weather forecasting [164, 219], stock market predictions [163], predicting human behavior [220], and movement predictions of objects within image frames [221, 222]. The promise of these modeling approaches should be tested for human-prosthetic system modeling. If high-fidelity models that forecast human-prosthetic dynamics can be attained, prosthesis control actions can be optimized to achieve a desired long-term trajectory, improve the safety of the device, and reduce electrical demands.

It is important to note that most of the previous studies involving human-prosthesis system modeling have used simple shallow neural network architectures such as feedforward neural networks [77, 119]. These configurations do not take the dynamic spatial-temporal relationships of these physical systems into account. These factors could be important to account for not only due to the system's highly nonlinear behavior but also because the system might experience a sudden change in dynamics which can affect the response. Knowing that dynamics can change, the designed neural network must be able to handle sudden dynamic changes and possible disturbances to the system. Some DNN architectures allow for a richer expression of these time-dependent dynamic relationships due to their internal state (memory) and have the potential to adapt to disturbances [159, 223, 224].

Additionally, the approach toward human-prosthesis modeling should be a multivariate regression problem since there are likely many input features that affect the system dynamics.

As such, learning the most important and dominating input features would be advantageous toward this application. Machine learning attention mechanisms [160, 225, 226] will be investigated in this research in order to enhance and devote more computational power to the small but important subset of the feature space. Discovering the dominant features is also useful for developing model-based control laws that are less sensitive to small input changes and training simpler DNNs without sacrificing performance.

In this research, three DNN architectures were developed and trained: (i) a multilayer feedforward neural network [227], (ii) a gated recurrent network [224], and (iii) a dual-stage attention-based gated recurrent network [228]. While the ultimate goal is to implement a data-driven model-based control strategy for prosthetic devices, a key milestone and the focus of this chapter is the development of models that can achieve accurate predictions of human-PAFP dynamics. The output of the models was chosen to be the forward predictions of the prosthetic ankle torque. These output targets were computed using the full-body MoCap marker set (Vicon) and inverse dynamics principles to derive the total torque value at the ankle based on the movements of all anatomical structures that affect the joint. The reason for choosing the prosthetic ankle torque as the time series of interest is because many robotic prosthesis control methods aim to achieve a desired ankle torque or deploy a mathematical formula based on ankle impedance. Therefore, there is a need to observe ankle torque behavior in real time, however, angular kinetics from full-body inverse dynamics are much more challenging to measure using real-time sensors compared to joint kinematics or translational kinetics (i.e., GRF). This research aims to demonstrate models that provide accurate total ankle torque predictions using model inputs that can be easily measured in real time (i.e., encoder, load cell, and inertial sensor outputs). Previously collected experimental data where a subject walked using the prototype PAFP (in both passive and active modes) was used to train the models [27].

6.2 Methods

6.2.1 Baseline: Analytical Regression Model

Similar to training an artificial neural network (ANN), polynomial regression attempts to fit a nonlinear function to data. However, fitting a polynomial is typically much easier since polynomials have a much simpler form than most neural networks. In particular,

polynomial functions are expressed as linear functions of relevant monomials. Therefore, nonlinear least-squares fitting can be used to fit the set of coefficients (i.e., parameters) in the polynomial equation. The advantage of linear regression problems is that they are convex, which makes them easier to solve due to well-established convex optimization techniques. However, while ANN training is a non-convex problem which is typically harder to solve, requires tuning, and requires good initialization, the class of functions that can be learned effectively with an ANN or DNN is richer than polynomials. There are a number of reasons why DNNs may generalize better than polynomial models, which include: (i) DNNs learn to encode invariances due to their hierarchical nature, (ii) certain DNN architectures have mechanisms that learn temporal dependencies within the data (i.e., time delays, phase delays, non-minimum phase behavior, and nonlinearities), and (iii) certain DNNs have the ability to encode the relative importance of input features with regards to predicting the output variable. For this research, an analytical regression model was trained to predict total PAFP ankle torques. This model is used to analyze the benefits of using DNN models over linear regression models.

Recall from Equation 3.1 that the total prosthetic ankle torque τ_p is the sum of the passive torque from the parallel elastic element during loading τ_l and the active torque generated by the motor τ_a. The active torque, τ_a, is derived in Section 3.5 according to system geometry to create a nonlinear transmission model (Equation 3.34). The passive PAFP elements (i.e., cam device and spring) were designed to be approximated using a third-degree polynomial function [171]. Note that the damping and inertial effects of the motor are assumed to minimal and are therefore not included in the analytical model. The analytical regression problem is formulated as

$$\tau_p - R_\tau(\theta) k_\tau i_a = p_3\theta^3 + p_2\theta^2 + p_1\theta + p_0 \tag{6.2}$$

where θ represents the prosthetic ankle angle, $R_\tau(\theta)$ represents the transmission ratio as a function of θ (derived in Section 3.5), k_τ is the motor torque constant provided by Maxon, and i_a is the motor current command in Amps. Coefficients p_0, p_1, p_2, and p_3 represent the parameters of the PAFP's passive torque function to be solved in this nonlinear least-squares problem. The input-output data were fit to the nonlinear polynomial terms within Equation 6.2 using SciPy Optimize's leastsq function, which implements the Levenberg-Marquardt

algorithm to iteratively find local minimums.

Data were divided using the train-test split method where the last 15% of experimental walking trials were reserved to evaluate the polynomial model. Note that the test dataset was equivalent to the dataset used to evaluate the DNN models discussed in Section 6.2.2. The remaining data were shuffled then used for model training and cross-validation. Five-fold cross-validation was implemented where the original training/validation dataset was randomly partitioned into five equal-sized subsamples. One of these subsamples is used as the validation dataset to evaluate the model and the remaining subsamples are used for training. This process is repeated five times, with each of the five subsamples used once as the validation data. The parameters of the best-performing model out of the five trained models are then saved and used for testing and evaluation.

6.2.2 Neural Network Architectures

The DNN models were trained and implemented within the PyTorch framework [229]. Using the data collected in this research, the following neural network model architectures were trained and used to predict total PAFP torque targets:

FFN

A feedforward neural network (FFN), or fully connected network, consists of a series of fully connected layers that connect every neuron in one layer to every neuron in the other layer [230]. While being simple to implement and common across many applications, FFNs tend to not perform as well as more application-specific networks and cannot learn to modulate its input directly for better prediction results. Additionally, this type of network has no temporal memory since the layers connect unidirectionally, i.e., there are no cycles or feedback loops in the network.

GRU

A multi-layered gated recurrent unit (GRU) network consists of recurrent layers and a hidden state at each timestep [224]. At each timestep, the GRU layer adds information to or removes information from the hidden state which makes it particularly effective for learning temporal dependencies in time series data. The GRU layer uses a reset gate which controls the level

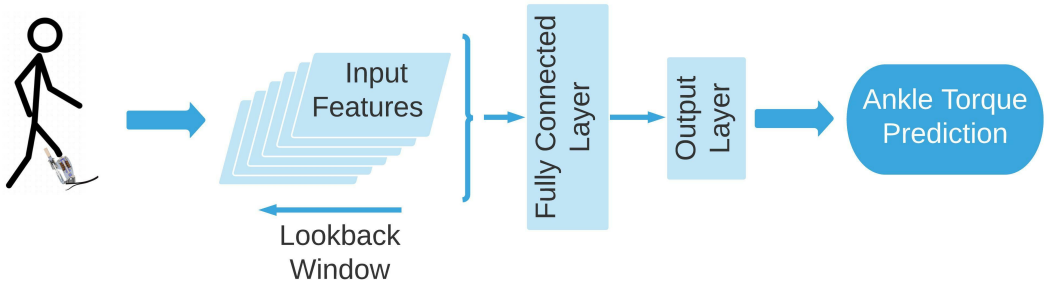


Figure 6.1: Visual Illustration of the feedforward network (FFN) architecture. A time history of input features are concatenated and fed into the FFN. The FFN is trained to output a predicted PAFP ankle torque from inverse dynamics. Note that only one fully connected layer is displayed in this diagram but multiple layers were tested during hyperparameter optimization.

of state reset, update gate which controls the level of state update, and candidate reset state which controls the level of update added to hidden state. This network is similar to long-short-term memory (LSTM) networks but has less parameters to train due to the lack of an output gate. However, the performance of GRUs have been shown to be similar to LSTMs for certain tasks [231, 232] and can even outperform them for training on smaller and less frequent datasets [233, 234].

DA-GRU

A dual-stage attention-based gated recurrent unit (DA-GRU) network, inspired by the DA-RNN [228], was developed for this study. This network uses a GRU architecture but also includes an attention mechanism that adaptively extracts the most relevant features at each timestep using an encoder-based hidden state [160]. An encoder is a type of network that maps input sequences into a representation where more relevant information is weighed more, i.e., it enhances and directs focus to the important parts of the input data [230]. Similarly, a temporal attention mechanism is also used to decode the relevant encoder hidden states across timesteps. Based upon these two attention mechanisms, the DA-GRU not only has the ability to capture the long-term temporal dependencies of the dataset similar to the GRU, but can also adaptively select the most relevant input features across the dataset.

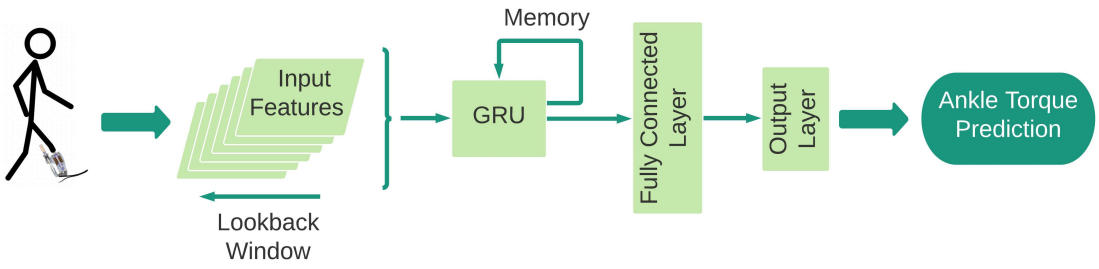


Figure 6.2: Visual Illustration of the gated recurrent unit (GRU) recurrent neural network architecture. A time history of input features are concatenated and fed into the GRU. The GRU is trained to output a predicted PAFP ankle torque from inverse dynamics. Note that only one GRU is displayed in this diagram but multiple layers were tested during hyperparameter optimization.

6.2.3 Data Processing

Data sampled and logged from experimental human subjects trials was used to create the datasets for neural network training and evaluation. The data were then re-sampled from 120 Hz to 30 Hz to reduce training time without any loss in performance since most human movement data information is contained within 15 Hz [235]. The target prediction output of the models was the prosthetic-side ankle torque computed using MoCap system measurements and inverse dynamics methods. The analytical regression model and the DNNs were trained to predict the PAFP ankle torque values one sample ahead into the future. Additionally, neural network models were also trained to forecast ankle torque values twenty samples ahead, which was approximately half a gait cycle into the future. The following were used as inputs for the neural network models:

1. Left GRF
2. Right GRF
3. Ankle Encoder Angle
4. Thigh Angle

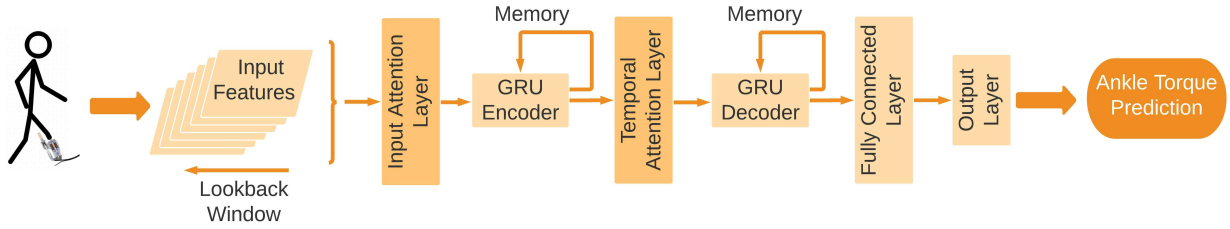


Figure 6.3: Visual Illustration of the dual-stage attention-based gated recurrent unit (DA-GRU) network architecture. A time history of input features are concatenated and fed into the DA-GRU which includes two attention layers. The GRU is trained to output a predicted PAFP ankle torque from inverse dynamics.

5. Motor Current Command

6. Motor Velocity

The data were then split into the three types of datasets commonly used in data-driven modeling applications: training, validation, and testing [236]. For each walking trial, the first 70% of the data were taken and used to create the training set. The training set is the actual dataset that is used to train the weights and biases of the neural network models, i.e., the models see and learn from this data. The following 15% of each of the trials were used to create the validation set. The validation set is used to frequently evaluate the given model, which is useful to fine-tune the model hyperparameters, reduce training time, and avoid overfitting. The models use this data during training but never learn from it so the validation set affects a model, but only indirectly. Finally, the remaining 15% of each trial was used to create the test set. The test set provides the gold standard used to evaluate the model once it is completely trained using the training and validation sets. The test set is generally what is used to evaluate competing models such as is the case for this research.

Randomly sampling (shuffling) the data every epoch to create the three sets is common practice in machine learning applications, however, this technique was not used in order to maintain the relevant temporal information within the time series. Datasets were converted into 4D tensors where the the first dimension represents the timestep, second dimension represents the walking trial (time series), third dimension represents the lookback time window

(sequence length), and fourth dimension represents the different model inputs (features). The training set was then split into batches along the first dimension in order to update the network parameters at more frequent intervals, which helps avoid local minima [237, 238]. The batch size is a tuneable hyperparameter that was optimized.

Data Scaling and Reshaping

A common strategy in deep learning is feature scaling, which is useful when multiple features are used that exhibit different range of motion (ROM) values [239]. This method is used to transform each feature to a similar scale in order to avoid sensitivity to magnitude shifts during learning and high-ROM features from dominating predictions. It is common practice to scale each feature x to the interval [0,1] based on its maximum and minimum values within the training dataset.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{6.3}$$

Additive Gaussian Noise

When DNNs are trained on small datasets, they are prone to memorize the training dataset instead of learning from the general features of the system. However, the use of noise as a regularizing method in a neural network can reduce overfitting to the training data [230]. The network is less likely to memorize training samples when noise is added because the features are constantly changing, resulting in smaller network weights and a more robust DNN that has lower generalization errors (i.e., lower variance). Adding noise is a common data augmentation strategy where new samples are being drawn from the domain in the vicinity of known samples, smoothing the structure of the input space. This smoothing may mean that the input-output mapping function is easier for the network to learn, resulting in better and faster learning. Random noise was added to each input feature as

$$x_{noisy} = x + \lambda\nu \tag{6.4}$$

where λ is a noise scaling hyperparameter and ν are random numbers from a normal distribution with mean 0 and variance 1. Note that random noise was added to the inputs of the training and validation sets but the test set was unchanged.

Rolling Lookback Time Window

A rolling lookback time window (i.e., sequence length) defines how many feature values of previous timesteps are used in order to predict the output of the subsequent timestep. This window, also known as the sequence length, is used in the backpropogation error computation and is treated as a tuneable hyperparameter. Choosing a larger window length increases the number of parameters that need to be trained since the window length affects the amount of DNN inputs and the time scale of the GRU hidden state. In addition, increasing the window length could result in overfitting since the network could simply memorize temporal patterns rather than learn a generalizable function. Alternatively, choosing too short of a sequence length increases the learning difficulty.

6.2.4 Loss Function & Network Parameter Optimization

A loss function is a measure of how good a prediction model does in terms of being able to predict the expected outcome. The loss function to be optimized was defined as the mean squared error (MSE) between the neural network output prediction and the measured target. MSE is the most common loss function when training regression neural networks in deep learning applications [240]. The AdamW optimization algorithm [241], an expansion of the Adam optimization strategy [242] with decoupled weight decay functionality, was used to update the network weights. This optimization algorithm was chosen over traditional stochastic gradient descent since it can handle sparse gradients on noisy problems and substantially improves Adam's generalization performance.

A learning rate scheduler was also implemented which allowed the dynamic learning rate to be reduced based on when validation predictions have stopped improving [243]. The number of epochs with no improvement after which learning rate will be reduced (i.e., patience) was set to 3. Thus, the optimizer will ignore the first 2 epochs with no improvement and will only decrease the learning rate after the third epoch if the loss still has not improved. Also, the number of epochs to wait before resuming normal operation after the learning rate has been reduced (i.e., cooldown) was set to 3. The factor by which the learning rate reduces γ was a tuneable hyperparameter. The minimum learning rate was set to 1e-5.

6.2.5 Network Hyperparameter Optimization Procedure & Outcomes

The Optuna framework was used to optimize the network training hyperparameters [244]. Optuna combines state-of-the-art Bayesian algorithms for sampling hyperparameters [245–247] and efficient methods for pruning unpromising trials [248–250] for automatic machine learning parameter optimization. In this study, 500 combinations of hyperparameters were tested for each network. Optimized hyperparameters for all networks included the sequence length, number of hidden layers, number of hidden units, dropout probability, noise scaling factor, batch size, initial learning rate, AdamW weight decay coefficient, and learning rate reduction factor. The DA-GRU hyperparameter optimization also included the number of hidden decoder units but excluded the dropout probability due to the layout of the network. Additionally, the number of hidden layers was fixed for DA-GRU hyperparameter optimization to match the original dual-layer architecture of the DA-RNN network [228].

Table 6.1: Hyperparameter Values Tested for Optimal Performance of the DNNs

Hyperparameter	**Range/Values**	**Optimal Value**					
		1-Sample			**20-Sample**		
		FFN	**GRU**	**DA-GRU**	**FFN**	**GRU**	**DA-GRU**
Sequence Length	[2, 3, ... , 19, 20]	16	15	18	20	18	20
Number of Layers	[1, 2, 3]	3	2	-	3	2	-
Number of HUs[a]	$[2^4, 2^5, ..., 2^8, 2^9]$	512	512	-	512	64	-
Number of Encoder HUs[a]	$[2^4, 2^5, ..., 2^8, 2^9]$	-	-	128	-	-	512
Number of Decoder HUs[a]	$[2^4, 2^5, 2^6, 2^7]$	-	-	16	-	-	64
Dropout Probability	[0.1 : 0.5]	0.114	0.199	-	0.100	0.121	-
Noise Scaling Factor	[0.1 : 1]	0.100	0.100	0.100	0.100	0.100	0.116
Batch Size	$[2^4, 2^5, 2^6, 2^7, 2^8]$	16	16	16	16	16	16
Initial LR[b]	$[10^{-5} : 10^{-1}]$	$5.1 \cdot 10^{-5}$	$1.3 \cdot 10^{-4}$	$2.8 \cdot 10^{-3}$	$4.6 \cdot 10^{-5}$	$4.5 \cdot 10^{-4}$	$1.8 \cdot 10^{-3}$
Weight Decay Coefficient	$[10^{-5} : 10^{-1}]$	0.062	0.019	$3.1 \cdot 10^{-3}$	0.073	0.013	0.070
LR[b] Reduction Factor	[0.1 : 0.9]	0.168	0.559	0.132	0.373	0.728	0.230

[a]HUs = Hidden Units
[b]LR = Learning Rate

The performance of the model was evaluated on the validation set every epoch. To reduce the chance of overfitting, an early stopping protocol was used to take the validation loss and count the number of epochs since the loss improved [251]. If the loss stops decreasing for 10 epochs in a row, the training stops and the best-performing model is saved. The maximum number of training epochs was set to 1000. Once all the hyperparameters were optimized for

the three networks over the 500 Optuna trials, predictions were generated and evaluated on the test dataset. The range of hyperparameter values tested and optimal values are shown in Table 6.1.

6.2.6 Analysis

Root Mean Squared Error (RMSE) and the Pearson correlation coefficient (PCC) were used as measures to report the performance of the different models. Ideally, the RMSE will be equal to zero degrees and PCC would be equal to 1. Additionally, percent RMSE values were used to compare the performance of the models. Percent RMSE was computed by dividing the RMSE value by the ankle torque's ROM within the walking trial.

6.3 Results & Discussion

Figures 6.4 and 6.5 show one-sample-ahead and twenty-sample-ahead time series predictions respectively for the different models. The periodic time series are time-normalized across the gait cycle for better visualization. Note that an analytical model was not derived for the twenty-sample ahead prediction problem so only the neural network architectures were compared for that particular test. Figures 6.6 and 6.7 show bar plots of the RMSE and percent RMSE values respectively for both the one-sample-ahead and twenty-sample-ahead time series predictions. Note that the time series data shown in this section were used only for model evaluation and were not part of the training data.

Table 6.2: Mean RMSE, percent RMSE, and PCC measures of one-sample-ahead model predictions across all time series (walking trial) data. Standard deviations are shown within parentheses.

	RMSE	**% RMSE**	**PCC**
Analytical	0.347(0.534)	26.625(40.932)	0.822(0.202)
FFN	0.036(0.024)	2.713(1.560)	0.996(0.006)
GRU	0.042(0.025)	3.166(1.681)	0.995(0.007)
DA-GRU	0.037(0.024)	2.768(1.542)	0.996(0.006)

The prediction accuracy of the DNN models was much higher across the board when compared to the analytical model. Note that in Figures 6.4, 6.6, and 6.7, the prediction

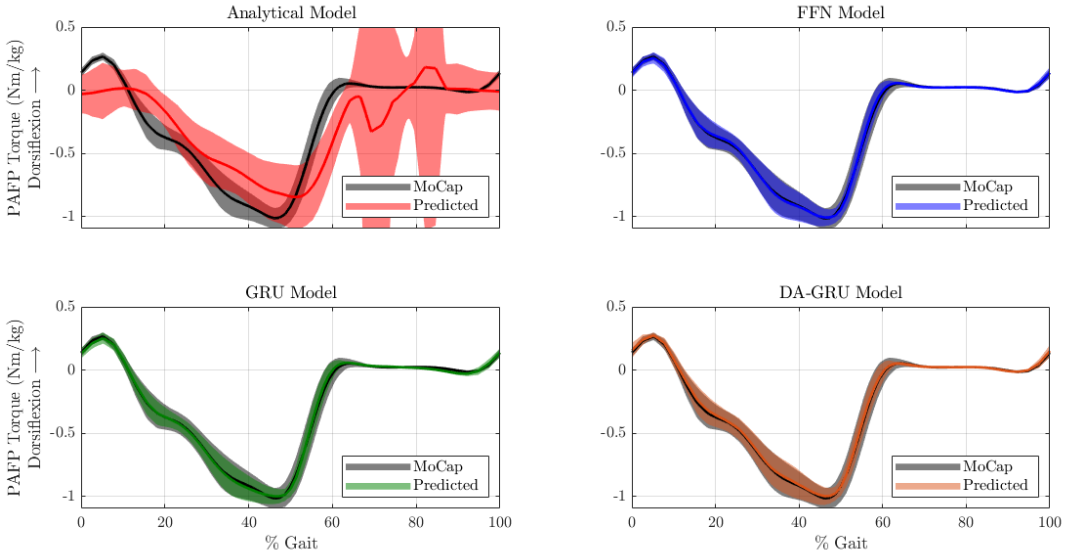


Figure 6.4: One-sample-ahead model predictions of total PAFP torques across gait cycles. Width of the traces represent ±1 standard deviation.

errors of the analytical model during swing were quite high. Therefore, the analytical swing phase predictions and errors were omitted from Figures 6.6 and 6.7 in order to not stretch the y-axis. Tables 6.2 and 6.3 show RMSE and PCC measures for the one-sample-ahead and twenty-sample-ahead predictions problems respectively for the different models.

DNN predictions showed a high PCC for both prediction tasks when compared to the analytical model. All correlation values for the DNNs were above 0.99 for the one-sample-ahead prediction task and the correlation only dropped by approximately 0.01 for the twenty-sample-ahead prediction task. Compared to the analytical model, which only showed a mean PCC value of 0.822 with a standard deviation close to 25% of its mean, the DNN results demonstrate a much higher correlation to the desired target predictions.

For both prediction tasks, the mean RMSE of the results were generally within 5% of the ROM of the total PAFP ankle torque. When considering approximately two thirds of the error data (i.e., mean ±1 standard deviation), DNN errors were only within 8% of the PAFP ankle torque ROM. The DNN errors were significantly lower than the analytical model which had a mean RMSE of 26.625% of the PAFP ankle torque ROM. When including the ranges

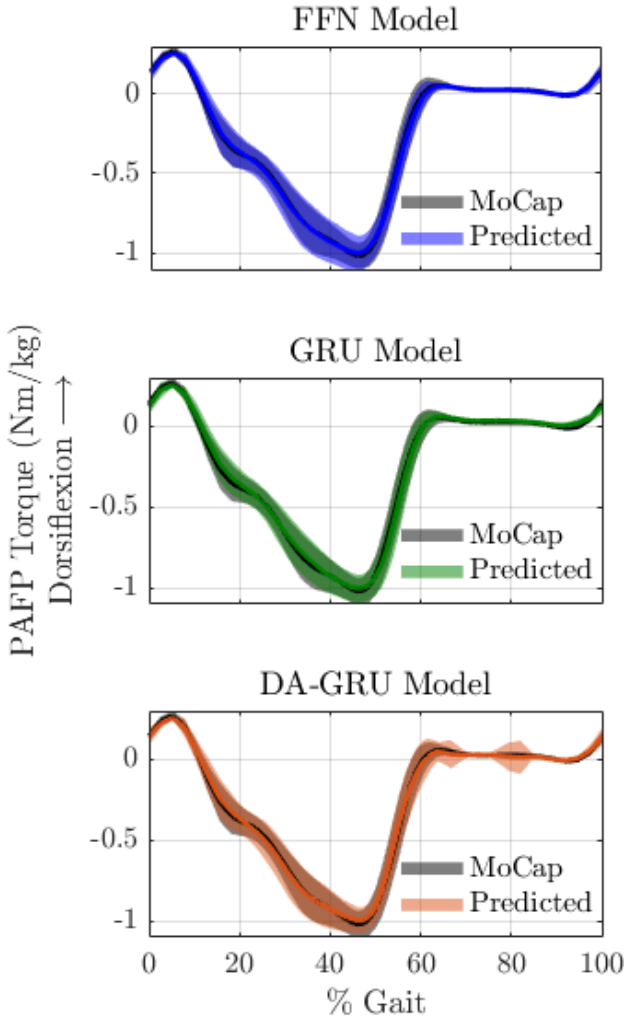


Figure 6.5: Twenty-sample-ahead model predictions of total PAFP torques across gait cycles. Width of the traces represent ±1 standard deviation.

within one standard deviation of the mean, the error went up to 58% of the ROM. From these results, all DNN models significantly outperformed the anlaytical regression model.

An interesting discovery was that all DNNs demonstrated approximately the same performance ($<$ 0.01 Nm/kg change,$<$ 0.75% RMSE change, and $<$ 0.01 change in PCC). Additionally, the results show that all DNNs can be retrained to predict approximately half a gait cycle into the future without compromising overall performance. When compared to

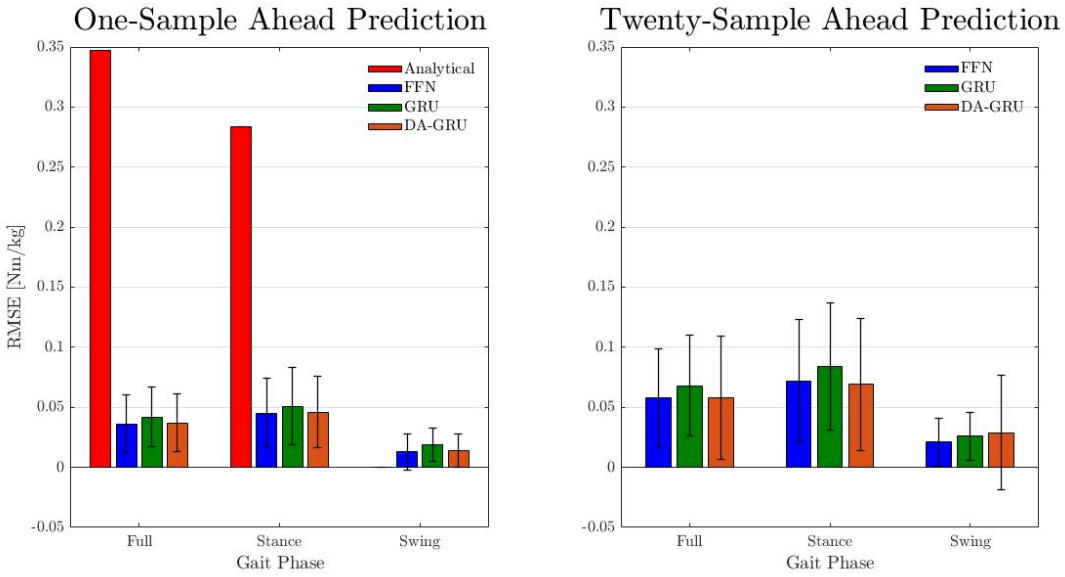


Figure 6.6: Root-mean-square error for each model class for both one-sample and twenty-sample ahead predictions. Errors are shown for stance and swing phases individually as well as the full gait cycle.

the one-sample ahead predictions, the twenty-sample-ahead prediction performance for all DNNs only decreased by an average of 0.023 Nm/kg and 1.68% of the PAFP ankle torque's ROM. The decrease in the PCC value across all DNNs was also small at approximately 0.01. Figures 6.6 and 6.7 display the approximately equivalent results for all the DNNs and how the predictions errors are within one standard deviation of each other.

Due to the simpler architecture and faster training time, it may be more beneficial to use the FFN for model-based control applications. Alternatively, the attention weights of the

Table 6.3: Mean RMSE, percent RMSE, and PCC measures of twenty-sample-ahead model predictions across all time series (walking trial) data. Standard deviations are shown within parentheses.

	RMSE	**% RMSE**	**PCC**
FFN	0.058(0.041)	4.301(2.824)	0.988(0.019)
GRU	0.068(0.042)	5.078(2.961)	0.985(0.024)
DA-GRU	0.058(0.051)	4.307(3.524)	0.985(0.030)

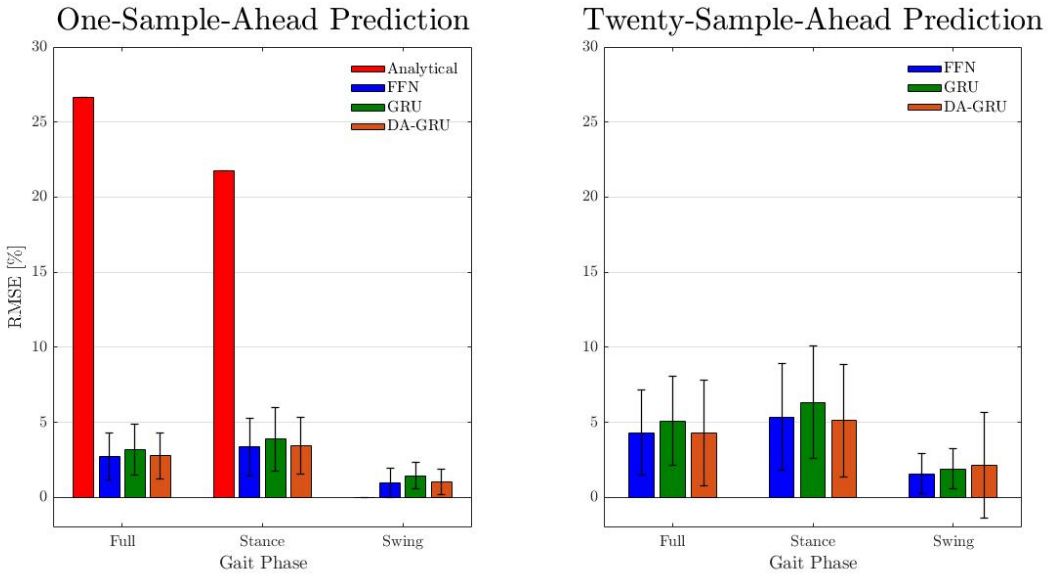


Figure 6.7: Root-mean-square percent error for each model class for both one-sample and twenty-sample ahead predictions. Errors are shown for stance and swing phases individually as well as the full gait cycle.

DA-GRU can be analyzed to determine the most important set of features when predicting total ankle torque behavior. Thus, the set of sensors could potentially be reduced. Both of these options have beneficial implications for deployment onto embedded systems. Even so, the results from this research indicate that highly-fidelity models can be constructed with minimum instrumentation and without using motion capture or full-body motion tracking suits. Particularly, a benefit of this research over other data-driven approaches toward prosthetic device control is that it does not rely on vision systems or EMG sensors to produce accurate predictions for steady state walking. However, this work can be expanded to include computer vision which could help the user avoid obstacles or anticipate activity changes.

The data-driven approach presented in this research eliminates the need to construct analytical models based on physics, which often require extensive bench tests and mechanical property identification. Instead, human locomotion data is collected and used to train highly-accurate models to predict total ankle torque values without the need of inverse dynamics or whole-body sensor systems. This modeling procedure results in predictions that are signifi-

cantly more accurate than the analytical regression method. The DNNs also demonstrated the ability to forecast future behavior with only a small loss in performance, which would likely be inconsequential for real-time control applications. Additionally, the DNNs were able to generalize across the whole gait cycle while the analytical model produced very poor results during swing phase. This strategy therefore eliminates the need to design hybrid system models and finite state machines, thus reducing the amount of control parameters that need to be tuned by experts.

For future work, one way to distinguish the capabilities of each DNN would be to analyze each of their dependence on data. For instance, one architecture may perform significantly better with less training data than others. This has important implications for clinical applications since it would reduce the amount of time to complete data collections, the number of clinical visits, and number of walking trials. Additional training data could also be collected for other tasks such as stair or ramp walking to determine if some DNN architectures perform better than others across various activities. Sensors could also be re-arranged to analyze changes in model performance, e.g., the thigh-mounted IMU or load cells could be configured in a different position or orientation. In cases such as this, the encoder-decoder functionality of the DA-GRU model may be advantageous since it is able to better learn an underlying representation of the training data and reconstructs its input features. The other two DNN models may be more sensitive to sensor configuration changes since the outputs may lay outside the training feature space.

Real-time tests on actual hardware should also be conducted in the future to test the feasibility of model-based predictive control of robotic prostheses. These tests would also give insight into prediction latency and the effects of sensor noise variations. Additionally, only a single participant was included in the training data which is an inherent limitation of this study. The focus of this research was to create predictive models that are personalized to the individual but more subjects should be included when training future models to see if its is feasible or if there are any benefits to utilizing inter-user models. Anthropometric and subject-specific data (e.g., weight, height, gender, limb lengths, etc.) could be used as additional features to possibly create models that generalize across individuals. Future tests should also assess how the models perform during non-rhythmic movements.

This research aims to take strides toward model-based control methods for PAFPs by developing data-driven predictive models that outperform the analytical regression model of

the system. Ambulation data from a single subject was used to train the analytical regression model and three DNN architectures. The trained DNNs demonstrate that sophisticated data-driven models can be used to accurately predict total ankle torque values that are normally only available after computing full-body inverse dynamics offline. Performance was within 5% of the PAFP torque's ROM for both one-sample-ahead prediction and predicting PAFP torque half a gait cycle (twenty samples at 30 Hz) ahead. Future experiments will expand these modeling efforts across individuals and tasks. Additionally, the DNNs will be implemented into real-time prediction applications and then into fully-realized model-based PAFP controllers.

Chapter 7

FUTURE WORK AND CONCLUSIONS

7.1 Future Work

7.1.1 Mechanical Design

Improvements to the mechanical design of the prototype PAFP are possible. For instance, the overall device mass can be reduced by using composite materials on the metal links and the cam spring [252]. In addition, the coil spring can be replaced with a leaf spring which would lower the stack height of the device and allow a wider range of subjects to be able to wear the device. Additionally, leaf springs are capable of handling much higher loads with less deflection than coil springs. Material can also be removed from the middle of the shank links to further reduce the mass of the device while sacrificing minimal stability. More complex cam shapes are also possible, e.g., cams that encode multiple trajectories and can switch automatically. Lastly, a more in-depth investigation into tailoring the passive nonlinear spring response would allow for a more informed procedure for customization and a deeper understanding of the resulting effects on joint mechanics.

7.1.2 Portability

Currently, the embedded system operates on a benchtop near the user so future work should focus on building a portable version of the embedded system. The first step would be to design an onboard battery system capable of powering the microcontroller, motor, and sensors. This may be challenging due to the high power demands of the servo controller. Additionally, the HSD setup could be improved by replacing the force plate sensors with an instrumented insole that includes an array of FSRs or integrating a load cell with the device. The IMU and encoder sensors could also be replaced with wireless or cloud-based options. Additionally, custom printed circuit boards could be fabricated to route signals, improve reliability, and minimize hardware real estate. Finally, the embedded system could be re-designed to be more compact and enclosed within a 3D-printed case. The portable

embedded system could then be attached to the prototype or worn by the subjects using Velcro straps.

7.1.3 Deploying Data-Driven Models

The results presented in Chapter 6 show that high fidelity human-robot system models are possible. These models would allow for increased learning, especially with respect to phase errors. Since the model would encode the frequency response of the actuator, phase errors greater than 90° would be accommodated. Moreover, the human-robot response dynamical model, which predicts achieved torques, can allow a ILC algorithm with model-inversion to be performed online and account for the human contributions. The data-driven models could also be used to build a simulation environment for the control algorithm to test and modify action policies without risking the safety of the user. The predictive capabilities of the deep neural network models could also be utilized for nonlinear model predictive control if future efforts are made toward inverting these networks. If inverting the neural networks is not feasible, the predictive networks can still be utilized in Monte Carlo methods (e.g., black box optimization, cross-entropy method, etc.) or model predictive path integral control [253]. In order for these real-time model-based control methods to be realized, significant work must be put toward utilizing GPU or FPGA hardware and parallel computation for efficient trajectory sampling. The high-fidelity models can also be used to better inform future mechanical designs for prototype PAFPs and how to best optimize the passive joint mechanics. Finally, introducing data from different tasks or activities may uncover some advantages between the different DNN models. For this reason, future work should include more experimental conditions such as different speeds, load carriage, different terrain, and transitions between them.

7.2 Conclusions

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

7.2.1 Personalizing the Active Response of the PAFP

This work proposes an impedance-based control strategy for robotic ankle-foot prostheses that personalizes to the user and targets their gait asymmetry. The control system utilizes a biomechanical phase variable calculation that acts as a time-invariant gait phase state estimator. Thus, the control strategy is more robust to perturbations or temporal changes in gait when compared to time-based control approaches. The feedback control method takes the form of an impedance-based law that modulates the motor torque command signal in response to the position of the prosthetic ankle. The novelty of this proposed methodology lies with the proposal to personalize the impedance law based on each individual's unique gait data. This data is used to learn a personalized virtual setpoint signal (i.e., trajectory planning) that is then encoded in the impedance control law and is indexed by the biomechanical phase variable in real time. This signal is modified using an adaptive gain iterative learning approach, which adjusts the prototype PAFP's angle-torque relationship to match the motion of prosthetic ankle to the intact ankle motion. By automatically tuning the control trajectory, the proposed approach does not rely on able-bodied gait data, which is a common limitation found in state-of-the-art PAFP controllers. In addition, this method does not require precise tuning of a large amount of parameters. Preliminary benchtop validation tests demonstrated that after a few iterations, the learning algorithm was able to modify the controller to track a fixed reference signal without any explicit programming. During human-in-the-loop experiments, the learning algorithm produced highly-personalized signals to each participant. The algorithm also maintained boundedness of the virtual trajectory, even in the presences of large variations in the reference signal brought on by human adaptations.

7.2.2 Experimental Evaluation with Target Patient Population

An experimental study ($n = 2$) was conducted to test if the learning algorithm and control strategy would reduce ankle asymmetries and loading factors associated with OA of the intact limb. The results indicate that the proposed control method was able to significantly improve ankle angle and ankle power symmetry when compared to using the prototype PAFP with no control (passive condition). Additionally, the proposed approach significantly reduced peak knee and hip abduction moments when compared to the passive and prescribed limb conditions. Secondary analyses also revealed significant adaptations at the knee and hip

of the intact limb. Therefore, this research demonstrates that adjusting control signals to improve symmetry about the ankle joints can result in improvements at the other joints of the lower limb, which motivates further investigation toward PAFP controllers.

7.2.3 Develop Models to Predict Prosthetic Ankle Torques from Full Body Inverse Dynamics

The total torque about a prototype powered ankle-foot prosthesis is well-approximated using DNN models and signals only from common wearable sensors. Experiments with test data demonstrate that the deep neural network models outperform analytical regression methods for time series prediction. Additionally, these neural networks demonstrated the ability to forecast PAFP torque values up to half a gait cycle into the future with minimal decreases in performance. As a result, these models and common wearable sensors can be used together to estimate dynamics that are typically unobservable in real time. Thus, these predictive models enable model-based control strategies aimed at optimizing prosthetic ankle torque, which have the potential to improve the mobility of powered limbs.

Bibliography

[1] D. C. Morgenroth, A. C. Gellhorn, and P. Suri, "Osteoarthritis in the Disabled Population: A Mechanical Perspective," may 2012.

[2] R. Gailey, K. Allen, J. Castles, J. Kucharik, and M. Roeder, "Review of secondary physical conditions associated with lower-limb amputation and long-term prosthesis use," pp. 15–30, 2008.

[3] D. A. Winter, "Energy generation and absorption at the ankle and knee during fast, natural, and slow cadences," *Clinical Orthopaedics and Related Research*, vol. No. 175, pp. 147–154, 1983.

[4] D. A. Winter and S. E. Sienko, "Biomechanics of below-knee amputee gait," *Journal of Biomechanics*, 1988.

[5] H. B. Skinner and D. J. Effeney, "Gait analysis in amputees," *American Journal of Physical Medicine*, vol. 64, no. 2, pp. 82–89, apr 1985.

[6] H. Bateni and S. J. Olney, "Kinematic and kinetic variations of below-knee amputee gait," pp. 2–10, 2002.

[7] P. G. Adamczyk and A. D. Kuo, "Mechanisms of gait asymmetry due to push-off deficiency in unilateral amputees," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 5, pp. 776–785, sep 2015.

[8] D. J. Sanderson and P. E. Martin, "Lower extremity kinematic and kinetic adaptations in unilateral below-knee amputees during walking," *Gait and Posture*, vol. 6, no. 2, pp. 126–136, oct 1997.

[9] E. Isakov, O. Keren, and N. Benjuya, "Trans-tibial amputee gait: Time-distance parameters and EMG activity," *Prosthetics and Orthotics International*, vol. 24, no. 3, pp. 216–220, 2000.

[10] S. J. Mattes, P. E. Martin, and T. D. Royer, "Walking symmetry and energy cost in persons with unilateral transtibial amputations: Matching prosthetic and intact limb inertial properties," *Archives of Physical Medicine and Rehabilitation*, vol. 81, no. 5, pp. 561–568, may 2000.

[11] R. D. Snyder, C. M. Powers, C. Fontaine, and J. Perry, "The effect of five prosthetic feet on the gait and loading of the sound limb in dysvascular below-knee amputees," *Journal of Rehabilitation Research and Development*, vol. 32, no. 4, pp. 309–315, nov 1995.

[12] M. S. Pinzur, W. Cox, J. Kaiser, T. Morris, A. Patwardhan, and L. Vrbos, "The Effect of Prosthetic Alignment on Relative Limb Loading in Persons with Trans-tibial Amputation : A Preliminary Report," Department of Veteran Affairs, Tech. Rep. 4, 1995.

[13] A. P. Arya, A. Lees, H. C. Nirula, and L. Klenerman, "A biomechanical comparison of the SACH, Seattle and Jaipur feet using ground reaction forces," University Department of Orthopaedics and Trauma Surgery, Tech. Rep., 1995.

[14] R. J. Zmitrewicz, R. R. Neptune, J. G. Walden, W. E. Rogers, and G. W. Bosker, "The Effect of Foot and Ankle Prosthetic Components on Braking and Propulsive Impulses During Transtibial Amputee Gait," *Archives of Physical Medicine and Rehabilitation*, vol. 87, no. 10, pp. 1334–1339, oct 2006.

[15] J. R. Engsberg, A. G. Lee, K. G. Tedford, and J. A. Harder, "Normative ground reaction force data for able-bodied and trans-tibial amputee children during running," *Prosthetics and Orthotics International*, vol. 17, no. 2, pp. 83–89, aug 1993.

[16] C. Beyaert, C. Grumillier, N. Martinet, J. Paysant, and J. M. André, "Compensatory mechanism involving the knee joint of the intact limb during gait in unilateral below-knee amputees," *Gait and Posture*, vol. 28, no. 2, pp. 278–284, aug 2008.

[17] D. C. Morgenroth, A. D. Segal, K. E. Zelik, J. M. Czerniecki, G. K. Klute, P. G. Adamczyk, M. S. Orendurff, M. E. Hahn, S. H. Collins, and A. D. Kuo, "The effect of prosthetic foot push-off on mechanical loading associated with knee osteoarthritis in lower extremity amputees," *Gait and Posture*, vol. 34, no. 4, pp. 502–507, 2011.

[18] E. Russell Esposito and J. M. Wilken, "Biomechanical risk factors for knee osteoarthritis when using passive and powered ankle-foot prostheses," *Clinical Biomechanics*, vol. 29, no. 10, pp. 1186–1192, dec 2014.

[19] H. M. Herr and A. M. Grabowski, "Bionic ankle-foot prosthesis normalizes walking gait for persons with leg amputation," *Proceedings of the Royal Society B: Biological Sciences*, vol. 279, no. 1728, pp. 457–464, 2012.

[20] R. Jiménez-Fabián and O. Verlinden, "Review of control algorithms for robotic ankle systems in lower-limb orthoses, prostheses, and exoskeletons," *Medical Engineering and Physics*, vol. 34, no. 4, pp. 397–408, 2012.

[21] A. S. Voloshina and S. H. Collins, "Lower limb active prosthetic systems—overview," *Wearable Robotics*, pp. 469–486, 2020.

[22] R. Müller, L. Tronicke, R. Abel, and K. Lechler, "Prosthetic push-off power in transtibial amputee level ground walking: A systematic review," *PLoS ONE*, vol. 14, no. 11, pp. 1–15, 2019.

[23] E. Chumacero, A. A. Masud, and D. Isik, "Advances in Powered Ankle-Foot Prostheses," *Critical Reviews in Biomedical Engineering*, vol. 46, no. 3, pp. 185–200, 2018.

[24] M. R. Tucker, J. Olivier, A. Pagel, H. Bleuler, M. Bouri, O. Lambercy, J. R. Del Millán, R. Riener, H. Vallery, and R. Gassert, "Control strategies for active lower extremity prosthetics and orthotics: A review," *Journal of NeuroEngineering and Rehabilitation*, vol. 12, no. 1, 2015.

[25] M. Windrich, M. Grimmer, O. Christ, S. Rinderknecht, and P. Beckerle, "Active lower limb prosthetics: A systematic review of design issues and solutions," *BioMedical Engineering Online*, vol. 15, no. 140, 2016.

[26] R. Versluys, P. Beyl, M. Van Damme, A. Desomer, R. Van Ham, and D. Lefeber, "Prosthetic feet: State-of-the-art review and the importance of mimicking human anklefoot biomechanics," *Disability and Rehabilitation: Assistive Technology*, vol. 4, no. 2, pp. 65–75, 2009.

[27] J. Realmuto, "Towards Personalized Powered Ankle-Foot Prostheses," *ProQuest Dissertations and Theses*, p. 262, 2017.

[28] D. A. Winter, "Kinematic and kinetic patterns in human gait: Variability and compensating effects," *Human Movement Science*, vol. 3, no. 1-2, pp. 51–76, 1984.

[29] S. K. Au, J. Weber, and H. Herr, "Powered ankle-foot prosthesis improves walking metabolic economy," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 51–66, 2009.

[30] P. Cherelle, V. Grosu, L. Flynn, K. Junius, M. Moltedo, B. Vanderborght, and D. Lefeber, "The Ankle Mimicking Prosthetic Foot 3—Locking mechanisms, actuator design, control and experiments with an amputee," *Robotics and Autonomous Systems*, vol. 91, pp. 327–336, may 2017.

[31] J. Sun and P. A. Voglewede, "Powered Transtibial Prosthetic Device Control System Design, Implementation, and Bench Testing," *Journal of Medical Devices*, vol. 8, no. 1, mar 2014.

[32] A. H. Shultz, B. E. Lawson, and M. Goldfarb, "Variable Cadence Walking and Ground Adaptive Standing with a Powered Ankle Prosthesis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 4, pp. 495–505, apr 2016.

[33] J. M. Caputo and S. H. Collins, "Prosthetic ankle push-off work reduces metabolic rate but not collision work in non-amputee walking," *Scientific Reports*, vol. 4, pp. 37–41, 2014.

[34] E. S. Gardinier, B. M. Kelly, J. Wensman, and D. H. Gates, "A controlled clinical trial of a clinically-tuned powered ankle prosthesis in people with transtibial amputation," *Clinical Rehabilitation*, vol. 32, no. 3, pp. 319–329, 2018.

[35] M. F. Eilenberg, H. Geyer, and H. Herr, "Control of a powered ankle-foot prosthesis based on a neuromuscular model," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 2, pp. 164–173, apr 2010.

[36] D. Gopinath, S. Jain, and B. D. Argall, "Human-in-the-loop optimization of shared autonomy in assistive robotics," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 247–254, 2017.

[37] D. Quintero, A. E. Martin, and R. D. Gregg, "Toward Unified Control of a Powered Prosthetic Leg: A Simulation Study," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 305–312, 2018.

[38] J. De la Fuente, T. G. Sugar, and S. Redkar, "Nonlinear, phase-based oscillator to generate and assist periodic motions," *Journal of Mechanisms and Robotics*, vol. 9, no. 2, apr 2017.

[39] E. J. Rouse, L. J. Hargrove, E. J. Perreault, T. A. Kuiken, E. J. Perreault, and T. A. Kuiken, "Estimation of human ankle impedance during the stance phase of walking," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 870–878, 2014.

[40] V. T. Inman, "Human Locomotion," *Canadian Medical Association journal*, vol. 94, no. 20, pp. 1047–1054, may 1966.

[41] D. A. Winter, *Biomechanics and Motor Control of Human Movement: Fourth Edition.* Wiley, 2009.

[42] ——, "Human balance and posture control during standing and walking," pp. 193–214, dec 1995.

[43] M. L. Palmer, "Sagittal Plane Charactrization of Normal Human Ankle Function Across a Range of Walking Gait Speeds," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[44] B. L. Riemann, R. G. Demont, K. Ryu, and S. M. Lephart, "The effects of sex, joint angle, and the gastrocnemius muscle on passive ankle joint complex stiffness," University of Pittsburgh, Tech. Rep. 4, 2001.

[45] M. M. Mirbagheri, H. Barbeau, M. Ladouceur, and R. E. Kearney, "Intrinsic and reflex stiffness in normal and spastic, spinal cord injured subjects," *Experimental Brain Research*, vol. 141, no. 4, pp. 446–459, 2001.

[46] N. H. Molen, "Energy/speed relation of below-knee amputees walking on a motor-driven treadmill," *Internationale Zeitschrift für Angewandte Physiologie Einschließlich Arbeitsphysiologie*, vol. 31, no. 3, pp. 173–185, sep 1973.

[47] G. R. Colborne, S. Naumann, P. E. Longmuir, and D. Berbrayer, "Analysis of mechanical and metabolic factors in the gait of congenital below knee amputees," *American Journal of Physical Medicine & Rehabilitation*, vol. 71, no. 5, pp. 272–278, oct 1992.

[48] R. S. Gailey, M. A. Wenger, M. Raya, N. Kirk, K. Erbs, P. Spyropoulos, and M. S. Nash, "Energy expenditure of trans-tibia1 amputees during ambulation at self-selected pace," University of Miami Division of Physical Therapy, Tech. Rep., 1994.

[49] W. Felt, J. C. Selinger, J. Maxwell Donelan, and C. David Remy, "Body-In-The-Loop: Optimizing Device Parameters Using Measures of Instantaneous Energetic Cost," *PLoS ONE*, 2015.

[50] J. R. Koller, D. H. Gates, D. P. Ferris, and C. David Remy, "Body-in-the-Loop Optimization of Assistive Robotic Devices: A Validation Study," University of Michigan, Tech. Rep., 2016.

[51] J. Zhang, P. Fiers, K. A. Witte, R. W. Jackson, K. L. Poggensee, C. G. Atkeson, and S. H. Collins, "Human-in-the-loop optimization of exoskeleton assistance during walking," *Science*, vol. 356, no. 6344, pp. 1280–1283, jun 2017.

[52] Y. Ding, M. Kim, S. Kuindersma, and C. J. Walsh, "Human-in-the-loop optimization of hip assistance with a soft exosuit during walking," *Science Robotics*, vol. 3, no. 15, feb 2018.

[53] K. A. Witte, P. Fiers, A. L. Sheets-Singer, and S. H. Collins, "Improving the energy economy of human running with powered and unpowered ankle exoskeleton assistance," *Science Robotics*, vol. 5, no. 40, p. 9108, mar 2020.

[54] C. G. Welker, A. S. Voloshina, V. L. Chiu, and S. H. Collins, "Shortcomings of human-in-the-loop optimization for an ankle-foot prosthesis: A case series," *bioRxiv*, 2020.

[55] M. Mussman, W. Altwerger, J. Eisenstein, A. Turturro, A. Glockenberg, and L. Bubbers, "Contralateral lower extremity evaluation with a lower limb prosthesis." *Journal of the American Podiatry Association*, vol. 73, no. 7, pp. 344–346, 1983.

[56] J. Kulkarni, J. Adams, E. Thomas, and A. Silman, "Association between amputation, arthritis and osteopenia in British male war veterans with major lower limb amputations," *Clinical Rehabilitation*, vol. 12, no. 4, pp. 348–353, aug 1998.

[57] M. J. Burke, V. Roman, and V. Wright, "Bone and joint changes in lower limb amputees," *Annals of the Rheumatic Diseases*, vol. 37, no. 3, pp. 252–254, 1978.

[58] P. A. Struyf, C. M. van Heugten, M. W. Hitters, and R. J. Smeets, "The Prevalence of Osteoarthritis of the Intact Hip and Knee Among Traumatic Leg Amputees," *Archives of Physical Medicine and Rehabilitation*, vol. 90, no. 3, pp. 440–446, mar 2009.

[59] D. C. Norvell, J. M. Czerniecki, G. E. Reiber, C. Maynard, J. A. Pecoraro, and N. S. Weiss, "The prevalence of knee pain and symptomatic knee osteoarthritis among veteran traumatic amputees and nonamputees," *Archives of Physical Medicine and Rehabilitation*, vol. 86, no. 3, pp. 487–493, 2005.

[60] A. J. Baliunas, D. E. Hurwitz, A. B. Ryals, A. Karrar, J. P. Case, J. A. Block, and T. P. Andriacchi, "Increased knee joint loads during walking are present in subjects with knee osteoarthritis," *Osteoarthritis and Cartilage*, vol. 10, no. 7, pp. 573–579, 2002.

[61] D. E. Hurwitz, K. C. Foucher, D. R. Sumner, T. P. Andriacchi, A. G. Rosenberg, and J. O. Galante, "Hip motion and moments during gait relate directly to proximal femoral bone mineral density in patients with hip osteoarthritis," *Journal of Biomechanics*, vol. 31, no. 10, pp. 919–925, oct 1998.

[62] C. H. Lloyd, S. J. Stanhope, I. S. Davis, and T. D. Royer, "Strength asymmetry and osteoarthritis risk factors in unilateral trans-tibial, amputee gait," *Gait and Posture*, vol. 32, no. 3, pp. 296–300, jul 2010.

[63] T. D. Royer and C. A. Wasilewski, "Hip and knee frontal plane moments in persons with unilateral, trans-tibial amputation," *Gait and Posture*, vol. 23, no. 3, pp. 303–306, apr 2006.

[64] R. L. Attwells, S. A. Birrell, R. H. Hooper, N. J. Mansfield, R. L. Attwells, S. A. Birrell, R. H. Hooper, and N. J. Mansfield, "Influence of carrying heavy loads on

soldiers ' posture , movements and gait Influence of carrying heavy loads on soldiers ' posture , movements and gait," *Ergonomics*, vol. 49, no. 14, pp. 1527–1537, 2006.

[65] S. A. Birrell and R. A. Haslam, "The effect of military load carriage on 3-D lower limb kinematics and spatiotemporal parameters," *Ergonomics*, vol. 52, no. 10, pp. 1298–1304, 2009.

[66] B. Smith, K. M. Ashton, D. Bohl, R. C. Clark, J. B. Metheny, and S. Klassen, "Influence of carrying a backpack on pelvic tilt, rotation, and obliquity in female college students," *Gait & Posture*, vol. 23, pp. 263–267, 2006.

[67] S. A. Birrell and R. A. Haslam, "The effect of load distribution within military load carriage systems on the kinetics of human gait," *Applied Ergonomics*, vol. 41, no. 4, pp. 585–590, 2010.

[68] C. R. James, L. T. Atkins, J. S. Dufek, and B. T. Bates, "An exploration of load accommodation strategies during walking with extremity-carried weights," *Human Movement Science*, vol. 35, pp. 17–29, 2014.

[69] A. G. Lucas-cuevas, P. Pérez-soriano, M. Bush, A. Crossman, S. Llana, J. M. Cortell-tormo, and J. A. Pérez-turpin, "Effects of Different Backpack Loads in Acceleration Transmission during Recreational Distance Walking Effects by," *Journal of Human Kinetics*, vol. 37, no. June, pp. 81–89, 2013.

[70] J. J. Knapik, K. L. Reynolds, and E. Harman, "Soldier Load Carriage: Historical, Physiological, Biomechanical, and Medical Aspects," *Military Medicine*, vol. 169, no. 1, pp. 45–56, 2004.

[71] T. W. P. Huang and A. D. Kuo, "Mechanics and energetics of load carriage during human walking," *Journal of Experimental Biology*, vol. 217, no. 4, pp. 605–613, feb 2014.

[72] A. Silder, S. L. Delp, and T. Besier, "Men and women adopt similar walking mechanics and muscle activation patterns during load carriage," *Journal of Biomechanics*, vol. 46, no. 14, pp. 2522–2528, sep 2013.

[73] S. S. Doyle, E. D. Lemaire, M. Besemann, and N. L. Dudek, "Changes to level ground transtibial amputee gait with a weighted backpack," *Clinical Biomechanics*, vol. 29, no. 2, pp. 149–154, 2014.

[74] B. L. Schnall, B. D. Hendershot, J. C. Bell, and E. J. Wolf, "Kinematic analysis of males with transtibial amputation carrying military loads," *Journal of Rehabilitation Research and Development*, vol. 51, no. 10, pp. 1505–1514, 2014.

[75] A. Brandt, M. Liu, and H. H. Huang, "Does the impedance of above-knee powered prostheses need to be adjusted for load-carrying conditions?" *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2016-Octob, pp. 5075–5078, 2016.

[76] J. Hitt and T. Sugar, "Load carriage effects on a robotic transtibial prosthesis," *ICCAS 2010 - International Conference on Control, Automation and Systems*, pp. 139–142, 2010.

[77] S. Au, M. Berniker, and H. Herr, "Powered ankle-foot prosthesis to assist level-ground and stair-descent gaits," *Neural Networks*, vol. 21, no. 4, pp. 654–666, may 2008.

[78] E. C. Martinez-Villalpando, L. Mooney, G. Elliott, and H. Herr, "Antagonistic active knee prosthesis. A metabolic cost of walking comparison with a variable-damping prosthetic knee," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2011. Annu Int Conf IEEE Eng Med Biol Soc, 2011, pp. 8519–8522.

[79] K. A. Shorter, J. Xia, E. T. Hsiao-Wecksler, W. K. Durfee, and G. F. Kogler, "Technologies for powered ankle-foot orthotic systems: Possibilities and challenges," pp. 337–347, 2013.

[80] H. A. Varol, F. Sup, and M. Goldfarb, "Multiclass real-time intent recognition of a powered lower limb prosthesis," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 3, pp. 542–551, mar 2010.

[81] H. Herr and A. Wilkenfeld, "User-adaptive control of a magnetorheological prosthetic knee," *Industrial Robot*, vol. 30, no. 1, pp. 42–55, 2003.

[82] F. Sup, H. A. Varol, and M. Goldfarb, "Upslope walking with a powered knee and ankle prosthesis: Initial results with an amputee subject," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, no. 1, pp. 71–78, feb 2011.

[83] B. E. Lawson, H. A. Varol, and M. Goldfarb, "Standing stability enhancement with an intelligent powered transfemoral prosthesis," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2617–2624, sep 2011.

[84] Y. David Li and E. T. Hsiao-Wecksler, "Gait mode recognition and control for a portable-powered ankle-foot orthosis," in *IEEE International Conference on Rehabilitation Robotics*, vol. 2013. IEEE Int Conf Rehabil Robot, 2013.

[85] H. Quintero, R. Farris, C. Hartigan, I. Clesson, and M. Goldfarb, "A powered lower limb orthosis for providing legged mobility in paraplegic individuals," *Topics in Spinal Cord Injury Rehabilitation*, vol. 17, no. 1, pp. 25–33, may 2011.

[86] F. Zhang, M. Liu, and H. Huang, "Preliminary study of the effect of user intent recognition errors on volitional control of powered lower limb prostheses," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2012. Annu Int Conf IEEE Eng Med Biol Soc, 2012, pp. 2768–2771.

[87] M. Goršič, R. Kamnik, L. Ambrožič, N. Vitiello, D. Lefeber, G. Pasquini, and M. Munih, "Online phase detection using wearable sensors for walking with a robotic prosthesis," *Sensors (Switzerland)*, vol. 14, no. 2, pp. 2776–2794, feb 2014.

[88] F. Zhang, M. Liu, and H. Huang, "Effects of locomotion mode recognition errors on volitional control of powered above-knee prostheses," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 1, pp. 64–72, jan 2015.

[89] H. Kawamoto, S. Kanbe, and Y. Sankai, "Power assist method for HAL-3 estimating operator's intention based on motion information," in *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, 2003, pp. 67–72.

[90] A. H. Shultz, B. E. Lawson, and M. Goldfarb, "Running with a powered knee and ankle

prosthesis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 3, pp. 403–412, may 2015.

[91] B. E. Lawson, A. H. Shultz, and M. Goldfarb, "Evaluation of a coordinated control system for a pair of powered transfemoral prostheses," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 3888–3893.

[92] Y. Sankai, "HAL: Hybrid assistive limb based on cybernics," in *Springer Tracts in Advanced Robotics*, vol. 66. Springer, Berlin, Heidelberg, 2010, pp. 25–34.

[93] R. J. Farris, H. A. Quintero, and M. Goldfarb, "Performance evaluation of a lower limb exoskeleton for stair ascent and descent with Paraplegia," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2012. NIH Public Access, 2012, pp. 1908–1911.

[94] C. D. Hoover, G. D. Fulk, and K. B. Fite, "Stair ascent with a powered transfemoral prosthesis under direct myoelectric control," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 3, pp. 1191–1200, 2013.

[95] D. Y. Li, A. Becker, K. A. Shorter, T. Bretl, and E. T. Hsiao-Wecksler, "Estimating system state during human walking with a powered ankle-foot orthosis," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 5, pp. 835–844, oct 2011.

[96] K. Fite, J. Mitchell, F. Sup, and M. Goldfarb, "Design and control of an electrically powered knee prosthesis," in *2007 IEEE 10th International Conference on Rehabilitation Robotics, ICORR'07*, 2007, pp. 902–905.

[97] B. E. Lawson, H. A. Varol, A. Huff, E. Erdemir, and M. Goldfarb, "Control of stair ascent and descent with a powered transfemoral prosthesis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 21, no. 3, pp. 466–473, 2013.

[98] B. E. Lawson, H. A. Varol, F. Sup, and M. Goldfarb, "Stumble detection and classification for an intelligent transfemoral prosthesis," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, vol. 2010. Annu Int Conf IEEE Eng Med Biol Soc, 2010, pp. 511–514.

[99] M. Liu, F. Zhang, P. Datseris, and H. H. Huang, "Improving Finite State Impedance Control of Active-Transfemoral Prosthesis Using Dempster-Shafer Based State Transition Rules," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 76, no. 3-4, pp. 461–474, oct 2014.

[100] S. Murray and M. Goldfarb, "Towards the use of a lower limb exoskeleton for locomotion assistance in individuals with neuromuscular locomotor deficits," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2012. NIH Public Access, 2012, pp. 1912–1915.

[101] A. H. Shultz, J. E. Mitchell, D. Truex, B. E. Lawson, and M. Goldfarb, "Preliminary evaluation of a walking controller for a powered ankle prosthesis," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., 2013, pp. 4838–4843.

[102] F. Sup, H. A. Varol, J. Mitchell, T. J. Withrow, and M. Goldfarb, "Preliminary evaluations of a self-contained anthropomorphic transfemoral prosthesis," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 6, pp. 667–676, dec 2009.

[103] F. Sup, A. Bohara, and M. Goldfarb, "Design and Control of a Powered Transfemoral Prosthesis." *The International journal of robotics research*, vol. 27, no. 2, pp. 263–273, feb 2008.

[104] F. Sup, H. A. Varol, J. Mitchell, T. Withrow, and M. Goldfarb, "Design and control of an active electrical knee and ankle prosthesis," in *Proceedings of the 2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2008*, vol. 2008. NIH Public Access, 2008, pp. 523–528.

[105] H. A. Varol and M. Goldfarb, "Decomposition-based control for a powered knee and ankle transfemoral prosthesis," in *2007 IEEE 10th International Conference on Rehabilitation Robotics, ICORR'07*, 2007, pp. 783–789.

[106] H. Wang, M. Kia, and D. C. Dickin, "Influences of load carriage and physical activity history on tibia bone strain," *Journal of Sport and Health Science*, vol. 8, no. 5, pp. 478–485, 2016.

[107] D. Zlatnik, B. Steiner, and G. Schweitzer, "Finite-state control of a trans-femoral (TF) prosthesis," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 3, pp. 408–420, may 2002.

[108] A. M. Simon, K. A. Ingraham, N. P. Fey, S. B. Finucane, R. D. Lipschutz, A. J. Young, and L. J. Hargrove, "Configuring a powered knee and ankle prosthesis for transfemoral amputees within five specific ambulation modes," *PLoS ONE*, vol. 9, no. 6, jun 2014.

[109] M. Li, X. Gao, Y. Wen, J. Si, and H. H. Huang, "Offline policy iteration based reinforcement learning controller for online robotic knee prosthesis parameter tuning," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., may 2019, pp. 2831–2837.

[110] Y. Wen, J. Si, A. Brandt, X. Gao, and H. Huang, "Online Reinforcement Learning Control for the Personalization of a Robotic Knee Prosthesis," *IEEE Transactions on Cybernetics*, vol. PP, pp. 1–11, 2019.

[111] D. Joshi and M. E. Hahn, "Terrain and Direction Classification of Locomotion Transitions Using Neuromuscular and Mechanical Input," *Annals of Biomedical Engineering*, vol. 44, no. 4, pp. 1275–1284, apr 2016.

[112] E. C. Martinez-Villalpando and H. Herr, "Agonist-antagonist active knee prosthesis: A preliminary study in level-ground walking," *Journal of Rehabilitation Research and Development*, vol. 46, no. 3, pp. 361–374, 2009.

[113] A. J. Young, A. M. Simon, N. P. Fey, and L. J. Hargrove, "Classifying the intent of novel users during human locomotion using powered lower limb prostheses," in *International IEEE/EMBS Conference on Neural Engineering, NER*, 2013, pp. 311–314.

[114] H. Huang, T. A. Kuiken, and R. D. Lipschutz, "A strategy for identifying locomotion modes using surface electromyography," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 1, pp. 65–73, jan 2009.

[115] D. C. Tkach and L. J. Hargrove, "Neuromechanical sensor fusion yields highest accuracies in predicting ambulation mode transitions for trans-tibial amputees," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine*

and Biology Society, EMBS, vol. 2013. Annu Int Conf IEEE Eng Med Biol Soc, 2013, pp. 3074–3077.

[116] H. Huang, F. Zhang, L. J. Hargrove, Z. Dou, D. R. Rogers, and K. B. Englehart, "Continuous locomotion-mode identification for prosthetic legs based on neuromuscular - Mechanical fusion," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 10 PART 1, pp. 2867–2875, oct 2011.

[117] L. J. Hargrove, A. M. Simon, A. J. Young, R. D. Lipschutz, S. B. Finucane, D. G. Smith, and T. A. Kuiken, "Robotic Leg Control with EMG Decoding in an Amputee with Nerve Transfers," *New England Journal of Medicine*, vol. 369, no. 13, pp. 1237–1242, sep 2013.

[118] A. J. Young, A. M. Simon, N. P. Fey, and L. J. Hargrove, "Intent recognition in a powered lower limb prosthesis using time history information," *Annals of Biomedical Engineering*, vol. 42, no. 3, pp. 631–641, mar 2014.

[119] S. K. Au, P. Bonato, and H. Herr, "An EMG-position controlled system for an active ankle-foot prosthesis: An initial experimental study," in *Proceedings of the 2005 IEEE 9th International Conference on Rehabilitation Robotics*, vol. 2005, 2005, pp. 375–379.

[120] S. Huang, J. P. Wensman, and D. P. Ferris, "Locomotor Adaptation by Transtibial Amputees Walking with an Experimental Powered Prosthesis under Continuous Myoelectric Control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 5, pp. 573–581, may 2016.

[121] N. Hogan, "Impedance control: An approach to manipulation: Part II-implementation," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 107, no. 1, pp. 8–16, mar 1985.

[122] C. D. Hoover, G. D. Fulk, and K. B. Fite, "The Design and Initial Experimental Validation of an Active Myoelectric Transfemoral Prosthesis," *Journal of Medical Devices, Transactions of the ASME*, vol. 6, no. 1, mar 2012.

[123] K. H. Ha, H. A. Varol, and M. Goldfarb, "Volitional control of a prosthetic knee using

surface electromyography," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 1, pp. 144–151, jan 2011.

[124] K. Bhakta, J. Camargo, P. Kunapuli, L. Childers, and A. Young, "Impedance Control Strategies for Enhancing Sloped and Level Walking Capabilities for Individuals with Transfemoral Amputation Using a Powered Multi-Joint Prosthesis," *MILITARY MEDICINE*, vol. 185, no. 1, pp. 490–499, 2020.

[125] H. Gomi and M. Kawato, "Equilibrium-point control hypothesis examined by measured arm stiffness during multijoint movement," *Science*, vol. 272, no. 5258, pp. 117–120, apr 1996.

[126] H. Lee, E. J. Rouse, and H. I. Krebs, "Summary of Human Ankle Mechanical Impedance during Walking," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 4, no. September, pp. 1–7, 2016.

[127] J. L. Stein and W. C. Flowers, "Stance phase control of above-knee prostheses: Knee control versus SACH foot design," *Journal of Biomechanics*, vol. 20, no. 1, pp. 19–28, 1987.

[128] R. D. Bellman, M. A. Holgate, and T. G. Sugar, "SPARKy 3: Design of an active robotic ankle prosthesis with two actuated degrees of freedom using regenerative kinetics," in *Proceedings of the 2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2008*, 2008, pp. 511–516.

[129] J. K. Hitt, R. Bellman, M. Holgate, T. G. Sugar, and K. W. Hollander, "The sparky (spring ankle with regenerative kinetics) project: Design and analysis of a robotic transtibial prosthesis with regenerative kinetics," in *2007 Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, DETC2007*, vol. 5 PART C, 2008, pp. 1587–1596.

[130] J. Hitt, T. Sugar, M. Holgate, R. Bellman, and K. Hollander, "Robotic transtibial prosthesis with biomechanical energy regeneration," *Industrial Robot*, vol. 36, no. 5, pp. 441–447, 2009.

[131] J. K. Hitt, T. G. Sugar, M. Holgate, and R. Bellman, "An active foot-ankle prosthesis with biomechanical energy regeneration," *Journal of Medical Devices, Transactions of the ASME*, vol. 4, no. 1, mar 2010.

[132] D. L. Grimes, W. C. Flowers, and M. Donath, "Feasibility of an active control scheme for above knee prostheses," *Journal of Biomechanical Engineering*, vol. 99, no. 4, pp. 215–221, nov 1977.

[133] W. J. Wang, J. Li, W. D. Li, and L. N. Sun, "An echo-based gait phase determination method of lower limb prosthesis," in *Advanced Materials Research*, vol. 706-708, 2013, pp. 629–634.

[134] N. A. Borghese, L. Bianchi, and F. Lacquaniti, "Kinematic deferminants of human locomotion," *Journal of Physiology*, vol. 494, no. 3, pp. 863–879, aug 1996.

[135] A. H. Hansen and D. S. Childress, "Investigations of roll-over shape: Implications for design, alignment, and evaluation of ankle-foot prostheses and orthoses," *Disability and Rehabilitation*, vol. 32, no. 26, pp. 2201–2209, 2010.

[136] M. A. Holgate, A. W. Böhler, and T. G. Sugar, "Control algorithms for ankle robots: A reflection on the state-of-the-art and presentation of two novel algorithms," in *Proceedings of the 2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2008*, 2008, pp. 97–102.

[137] R. D. Gregg, E. J. Rouse, L. J. Hargrove, and J. W. Sensinger, "Evidence for a time-invariant phase variable in human ankle control," *PLoS ONE*, vol. 9, no. 2, 2014.

[138] D. J. Villarreal and R. D. Gregg, "A survey of phase variable candidates of human locomotion," *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014*, pp. 4017–4021, 2014.

[139] D. Quintero, D. J. Villarreal, D. J. Lambert, S. Kapp, and R. D. Gregg, "Continuous-Phase Control of a Powered Knee-Ankle Prosthesis: Amputee Experiments Across Speeds and Inclines," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 686–701, 2018.

[140] R. D. Gregg, T. Lenzi, L. J. Hargrove, and J. W. Sensinger, "Virtual constraint control of a powered prosthetic leg: From simulation to experiments with transfemoral amputees," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1455–1471, 2014.

[141] D. Quintero, D. J. Villarreal, and R. D. Gregg, "Preliminary experiments with a unified controller for a powered knee-ankle prosthetic leg across walking speeds," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 5427–5433, 2016.

[142] D. Quintero, D. J. Lambert, D. J. Villarreal, and R. D. Gregg, "Real-Time continuous gait phase and speed estimation from a single sensor," *1st Annual IEEE Conference on Control Technology and Applications, CCTA 2017*, vol. 2017-Janua, pp. 847–852, 2017.

[143] D. J. Villarreal, D. Quintero, and R. D. Gregg, "Piecewise and unified phase variables in the control of a powered prosthetic leg," *IEEE International Conference on Rehabilitation Robotics*, pp. 1425–1430, 2017.

[144] S. Rezazadeh, D. Quintero, N. Divekar, E. Reznick, L. Gray, and R. D. Gregg, "A Phase Variable Approach for Improved Rhythmic and Non-Rhythmic Control of a Powered Knee-Ankle Prosthesis," *IEEE Access*, vol. 7, pp. 109 840–109 855, 2019.

[145] D. J. Villarreal and R. D. Gregg, "Controlling a Powered Transfemoral Prosthetic Leg Using a Unified Phase Variable," in *Wearable Robotics*, P. W. F. Jacob Rosen, Ed. Academic Press, 2020, ch. 24, pp. 487–506.

[146] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, mar 2009.

[147] A. Phinyomark, G. Petri, E. Ibáñez-Marcelo, S. T. Osis, and R. Ferber, "Analysis of Big Data in Gait Biomechanics: Current Trends and Future Directions," *Journal of Medical and Biological Engineering*, vol. 38, no. 2, pp. 244–260, 2018.

[148] H. T. T. Vu, D. Dong, H. L. Cao, T. Verstraten, D. Lefeber, B. Vanderborght, and J. Geeroms, "A review of gait phase detection algorithms for lower limb prostheses," *Sensors (Switzerland)*, vol. 20, no. 14, pp. 1–19, 2020.

[149] T. Zhen, L. Yan, and P. Yuan, "Walking gait phase detection based on acceleration signals using LSTM-DNN algorithm," *Algorithms*, vol. 12, no. 2, 2019.

[150] H. T. T. Vu, F. Gomez, P. Cherelle, D. Lefeber, A. Nowé, and B. Vanderborght, "ED-FNN: A new deep learning algorithm to detect percentage of the gait cycle for powered prostheses," *Sensors (Switzerland)*, vol. 18, no. 7, 2018.

[151] H. X. Tan, N. N. Aung, J. Tian, M. C. H. Chua, and Y. O. Yang, "Time series classification using a modified LSTM approach from accelerometer-based data: A comparative study for gait cycle detection," *Gait and Posture*, vol. 74, no. June, pp. 128–134, 2019.

[152] A. Mai and S. Commuri, "Intelligent control of a prosthetic ankle joint using gait recognition," *Control Engineering Practice*, vol. 49, pp. 1–13, apr 2016.

[153] B. Amos, I. Dario Jimenez Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for End-to-end Planning and Control," Carnegie Mellon University, Tech. Rep., 2019.

[154] N. Wagener, C.-A. Cheng, J. Sacks, and B. Boots, "An Online Learning Approach to Model Predictive Control," in *Robotics: Science and Systems*. Georgia Institute of Technology, 2019.

[155] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, dec 2018.

[156] I. Lenz, R. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Robotics: Science and Systems*, vol. 11. MIT Press Journals, 2015.

[157] A. Shekhar and A. Sharma, "Review of Model Reference Adaptive Control," in *2018 International Conference on Information, Communication, Engineering and Technology, ICICET 2018*. Institute of Electrical and Electronics Engineers Inc., nov 2018, pp. 1–5.

[158] A. D. Keleş and C. A. Yucesoy, "Development of a neural network based control algorithm for powered ankle prosthesis," *Journal of Biomechanics*, vol. 113, p. 110087, dec 2020.

[159] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3104–3112, sep 2014.

[160] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem. Neural information processing systems foundation, jun 2017, pp. 5999–6009.

[161] A. Graves, N. Jaitly, and A. R. Mohamed, "Hybrid speech recognition with Deep Bidirectional LSTM," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2013 - Proceedings*, 2013, pp. 273–278.

[162] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 779–788, jun 2015.

[163] Y. Wu, J. Miguel, H. Lobato, and Z. Ghahramani, "Dynamic Covariance Models for Multivariate Financial Time Series," University of Cambridge, Cambridge, Tech. Rep., 2013.

[164] P. Chakraborty, M. Marwah, M. Arlitt, and N. Ramakrishnan, "Fine-grained Photovoltaic Output Prediction using a Bayesian Ensemble," Virginia Tech, Tech. Rep., 2012.

[165] Z. Liu and M. Hauskrecht, "A Regularized Linear Dynamical System Framework for Multivariate Time Series Analysis." *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 2015, pp. 1798–1804, jan 2015.

[166] R. Dedić and H. Dindo, "SmartLeg: An intelligent active robotic prosthesis for lower-limb amputees," *2011 23rd International Symposium on Information, Communication and Automation Technologies, ICAT 2011*, pp. 1–7, 2011.

[167] V. Rai, A. Sharma, and E. Rombokas, "Mode-free Control of Prosthetic Lower Limbs," *2019 International Symposium on Medical Robotics, ISMR 2019*, pp. 1–7, 2019.

[168] V. Rai and E. Rombokas, "A framework for mode-free prosthetic control for unstructured terrains," *IEEE International Conference on Rehabilitation Robotics*, vol. 2019-June, pp. 796–802, 2019.

[169] V. Rai, A. Sharma, P. Preechayasomboon, and E. Rombokas, "Coordinated Movement for Prosthesis Reference Trajectory Generation: Temporal Factors and Attention," *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, vol. 2020-Novem, pp. 939–945, 2020.

[170] J. Realmuto, G. Klute, and S. Devasia, "Preliminary Investigation of Symmetry Learning Control for Powered Ankle-Foot Prostheses," *2019 Wearable Robotics Association Conference, WearRAcon 2019*, pp. 40–45, 2019.

[171] ——, "Nonlinear passive cam-based springs for powered ankle prostheses," *Journal of Medical Devices, Transactions of the ASME*, vol. 9, no. 1, 2015.

[172] M. Grimmer, M. Eslamy, S. Gliech, and A. Seyfarth, "A comparison of parallel- and series elastic elements in an actuator for mimicking human ankle joint in walking and running," in *Proceedings - IEEE International Conference on Robotics and Automation*. Saint Paul, Minnesota: IEEE, 2012, pp. 2463–2470.

[173] J. Realmuto, R. B. Warrier, and S. Devasia, "Iterative learning control for human-robot collaborative output tracking," *MESA 2016 - 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications - Conference Proceedings*, no. 4, pp. 1–6, 2016.

[174] Y. Zhang, B. Chu, and Z. Shu, "A Preliminary Study on the Relationship between Iterative Learning Control and Reinforcement Learning," *IFAC-PapersOnLine*, vol. 52, no. 29, pp. 314–319, 2019.

[175] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of Robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, jun 1984.

[176] S. Arimoto, "Learning control theory for robotic motion," *International Journal of Adaptive Control and Signal Processing*, vol. 4, no. 6, pp. 543–564, 1990.

[177] C. G. Atkeson and J. McIntyre, "Robot trajectory learning through practice," in *IEEE International Conference on Robotics and Automation*. IEEE, 1986, pp. 1737–1742.

[178] M. Norrlöf, "An adaptive iterative learning control algorithm with experiments on an industrial robot," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 245–251, apr 2002.

[179] ——, "An adaptive approach to iterative learning control with experiments on an industrial robot," *2001 European Control Conference, ECC 2001*, vol. 18, no. 2, pp. 220–225, 2001.

[180] J. Realmuto, R. B. Warrier, and S. Devasia, "Data-Inferred Personalized Human-Robot Models for Iterative Collaborative Output Tracking," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 91, no. 2, pp. 137–153, 2018.

[181] S. Tien, S. Devasia, and Q. Zou, "Iterative Control of Dynamics-Coupling-Caused Errors in Piezoscanners During High-Speed AFM Operation," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 6, pp. 921–931, 2005.

[182] G. M. Jeong and C. H. Choi, "Iterative learning control for linear discrete time non-minimum phase systems," *Automatica*, vol. 38, no. 2, pp. 287–291, feb 2002.

[183] P. B. Goldsmith, "On the equivalence of causal LTI iterative learning control and feedback control," *Automatica*, vol. 38, no. 4, pp. 703–708, apr 2002.

[184] M. Verwoerd, G. Meinsma, and T. de Vries, "On admissible pairs and equivalent feedback-Youla parameterization in iterative learning control," *Automatica*, vol. 42, no. 12, pp. 2079–2089, dec 2006.

[185] J. Ghosh and B. Paden, "A pseudoinverse-based iterative learning control," *IEEE Transactions on Automatic Control*, vol. 47, no. 5, pp. 831–837, may 2002.

[186] ——, "Nonlinear repetitive control," *IEEE Transactions on Automatic Control*, vol. 45, no. 5, pp. 949–954, 2000.

[187] J. Zhang, T. Chow, J. Ho, and X.-D. Li, "Iterative learning control with initial rectifying action for nonlinear continuous systems," *IET Control Theory & Applications*, vol. 3, no. 1, pp. 49–55, jan 2009.

[188] K. S. Kim and Q. Zou, "A modeling-free inversion-based iterative feedforward control for precision output tracking of linear time-invariant systems," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1767–1777, 2013.

[189] S. Rossignol, R. Dubuc, and J. P. Gossard, "Dynamic sensorimotor interactions in locomotion," pp. 89–154, jan 2006.

[190] D. J. Villarreal and R. D. Gregg, "Unified phase variables of relative degree two for human locomotion," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2016-Octob, pp. 6262–6267, 2016.

[191] D. G. E. Robertson, G. E. Caldwell, J. Hamill, G. Kamen, and S. N. Whittlesey, *Research Methods in Biomechanics*. Human Kinetics, 2014.

[192] I. T. Jolliffe, "Principal Component Analysis, Second Edition," *Encyclopedia of Statistics in Behavioral Science*, vol. 30, no. 3, 2002.

[193] K. H. Zou, K. Tuncali, and S. G. Silverman, *Correlation and Simple Linear Regression*. Radiology, 2003, vol. 227, no. 3.

[194] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.

[195] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd ed. Boston: MIT Press, 2018, vol. 3, no. 1.

[196] Å. Björck, *Numerical Methods for Least Squares Problems*. SIAM: Society for Industrial and Applied Mathematics, 1996.

[197] R. C. Team, "R: A language and environment for statistical computing," 2021. [Online]. Available: https://www.r-project.org/

[198] H. Wickham, M. Averick, J. Bryan, W. Chang, L. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. Pedersen, E. Miller, S. Bache, K. Müller, J. Ooms, D. Robinson, D. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani, "Welcome to the Tidyverse," *Journal of Open Source Software*, vol. 4, no. 43, 2019.

[199] V. Russel, "emmeans: Estimated Marginal Means, aka Least-Squares Means," 2020. [Online]. Available: https://cran.r-project.org/package=emmeans

[200] Z. Hao, "kableExtra: Construct Complex Table with 'kable' and Pipe Syntax," 2020. [Online]. Available: https://cran.r-project.org/package=kableExtra

[201] K. E. Zelik and E. C. Honert, "Ankle and foot power in gait analysis: Implications for science, technology and clinical assessment," *Journal of Biomechanics*, vol. 75, pp. 1–12, 2018.

[202] J. A. Boyan and A. W. Moore, "Generalization in Reinforcement Learning: Safely Approximating the Value Function," *Advances in Neural Information Processing Systems*, vol. 7, 1995.

[203] J. N. Tsitsiklis and B. V. Roy, "An Analysis of Temporal-Difference Learning with Function Approximation," *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, vol. 42, no. 5, 1997.

[204] D. H. Jacobson, "Differential Dynamic Programming Methods for Determining Optimal Control of Non-Linear Systems," Ph.D. dissertation, University of London, 1967.

[205] C. Atkeson, "Using Local Trajectory Optimizers to Speed Up Global Optimization in Dynamic Programming," *Advances in Neural Information Processing Systems*, vol. 6, 1993.

[206] Y. Tassa, T. Erez, and B. Smart, "Receding Horizon Differential Dynamic Programming," *arXiv*, 2021.

[207] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2017.

[208] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," *Advances in Neural Information Processing Systems*, vol. 12, pp. 1057–1063, 2000.

[209] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, jun 2015, pp. 1889–1897.

[210] S. Kakade, "A Natural Policy Gradient," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 2001, pp. 1531–1538.

[211] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning," in *2018 IEEE International Conference on Robotics and Automation*, 2018, pp. 7559–7566.

[212] W. C. Wong, E. Chee, J. Li, and X. Wang, "Recurrent Neural Network-Based Model Predictive Control for Continuous Pharmaceutical Manufacturing," *Mathematics 2018, Vol. 6, Page 242*, vol. 6, no. 11, p. 242, nov 2018.

[213] Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing," *Journal of Artificial Intelligence Research*, vol. 57, no. 1, pp. 345–420, 2016.

[214] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, jan 1989.

[215] G. Box, *Time Series Analysis: forecasting and control*, 4th ed. Wiley, 1976.

[216] G. N. A. Weigend A. S., "Time Series Prediction: Forecasting the Future and Understanding the Past," in *Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis*, 1992.

[217] N. M. Rezk, M. Purnaprajna, T. Nordstrom, and Z. Ul-Abdin, "Recurrent Neural Networks: An Embedded Computing Perspective," *IEEE Access*, vol. 8, pp. 57 967–57 996, 2020.

[218] S. Gomez-Gonzalez, S. Prokudin, B. Scholkopf, and J. Peters, "Real Time Trajectory Prediction Using Deep Conditional Generative Models," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 970–976, 2020.

[219] X. Shi, Z. Gao, L. Lausen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Deep learning for precipitation nowcasting: A benchmark and a new model," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5618–5628, 2017.

[220] A. Almeida and G. Azkune, "Predicting human behaviour with recurrent neural networks," *Applied Sciences (Switzerland)*, vol. 8, no. 2, 2018.

[221] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–18, 2017.

[222] K. Ehsani, D. Gordon, T. Nguyen, and R. Mottaghi, "What can you learn from your data?" *Hospital peer review*, vol. 38, no. 6, pp. 65–66, 2013.

[223] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, nov 1997.

[224] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. Association for Computational Linguistics (ACL), jun 2014, pp. 1724–1734.

[225] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical Attention Networks for Document Classification," in *NAACL*, 2016.

[226] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, sep 2014.

[227] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, jan 2015.

[228] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 0, pp. 2627–2633, apr 2017.

[229] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[230] I. G. Courville, Y. Bengio, and Aaron, *Deep Learning*. MIT Press, 2016.

[231] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light Gated Recurrent Units for Speech Recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, apr 2018.

[232] Y. Su and C. C. J. Kuo, "On Extended Long Short-term Memory and Dependent Bidirectional Recurrent Neural Network," *Neurocomputing*, vol. 356, pp. 151–161, feb 2018.

[233] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv*, dec 2014.

[234] N. Gruber and A. Jockisch, "Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text?" *Frontiers in Artificial Intelligence*, vol. 3, p. 40, jun 2020.

[235] M. Sun and J. O. Hill, "A method for measuring mechanical work and work efficiency during human activities," *Journal of Biomechanics*, vol. 26, no. 3, 1993.

[236] Y. Xu and R. Goodacre, "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning," *Journal of Analysis and Testing*, vol. 2, no. 3, pp. 249–262, jul 2018.

[237] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*. Springer Science and Business Media Deutschland GmbH, 2010, pp. 177–186.

[238] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points: Online stochastic gradient for tensor decomposition," in *Journal of Machine Learning Research*, vol. 40, no. 2015. Microtome Publishing, mar 2015.

[239] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning, ICML 2015*, vol. 1. International Machine Learning Society (IMLS), feb 2015, pp. 448–456.

[240] J. O. Berger, "Certain Standard Loss Functions," in *Statistical Decision Theory and Bayesian Analysis*, 2nd ed. New York: New York: Springer-Verlag, 1985, p. 60.

[241] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *7th International Conference on Learning Representations, ICLR 2019*, nov 2017.

[242] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, dec 2015.

[243] A. Patterson, Josh; Gibson, "Understanding Learning Rates," in *Deep Learning : A Practitioner's Approach*. O'Reilly, 2017, pp. 258–263.

[244] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, jul 2019.

[245] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *International Conference on Learning and Intelligent Optimization*, 2011, pp. 507–523.

[246] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Advances in Neural Information Processing Systems*, 2011.

[247] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2012, pp. 2951–2959.

[248] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*. PMLR, feb 2016, pp. 240–248.

[249] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar, "A System for Massively Parallel Hyperparameter Tuning," in *Proceedings of Machine Learning and Systems*, oct 2018.

[250] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, pp. 1–52, apr 2018.

[251] L. Prechelt, "Early stopping - But when?" *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7700, pp. 53–67, 2012.

[252] M. K. Shepherd and E. J. Rouse, "Design of a quasi-passive ankle-foot prosthesis with biomimetic, variable stiffness," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 6672–6678, 2017.

[253] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

Appendix A

EMBEDDED SYSTEM HARDWARE

Figure A.1 contains the wiring schematic and signal routing for the embedded system. A CAD rendering of the embedded system mounting case is shown in Figure A.2. The onboard processors (e.g., myRIO and Escon 70/10) with custom protoboard circuits are shown in Figure A.3. The full enclosure, placed on a table near the instrumented treadmill and secured with clamps, is shown in Figure A.4.

A.1 Wiring Diagram

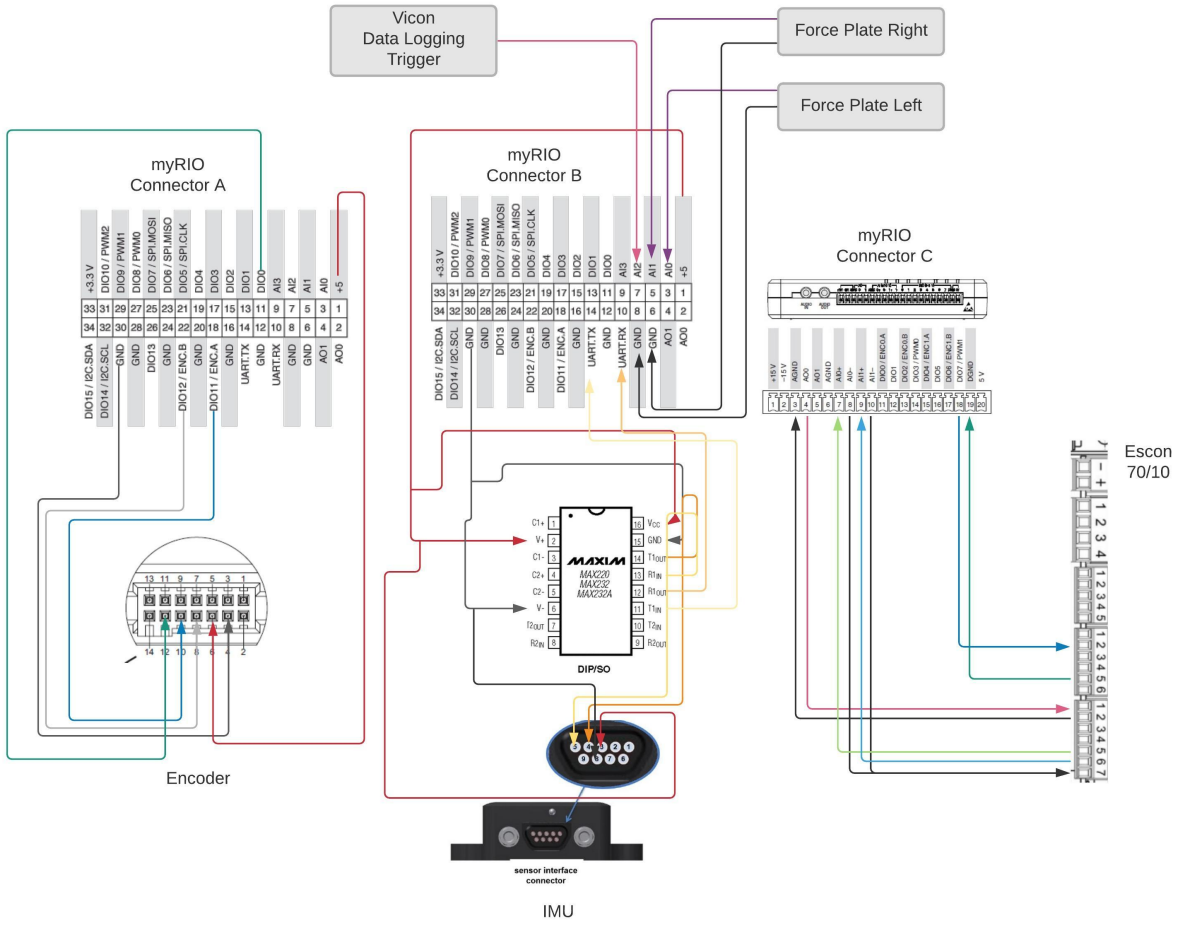


Figure A.1: Custom circuit used to route signals, including RS-232 serial to transistor-transistor logic converter (MAX232, Maxim) for sampling IMU signals via UART.

A.2 3D Printed Enclosure

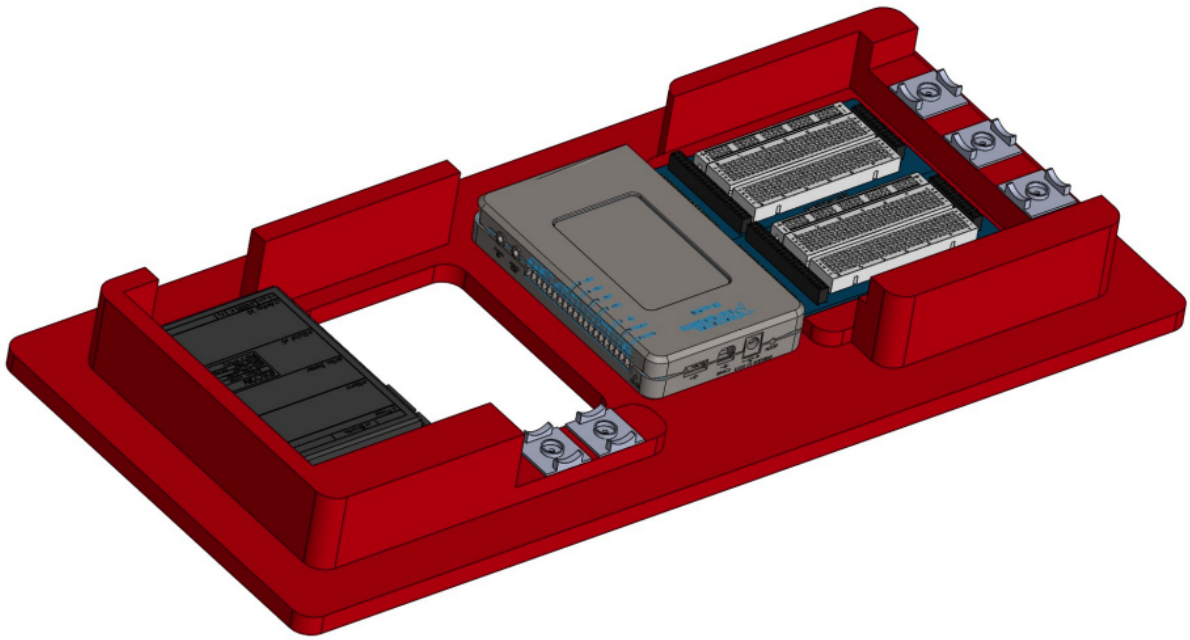


Figure A.2: CAD rendering of the embedded system hardware including the microcontrollers, 3D-printed case, breakout circuits, and strain relief clamps.

A.3 Images of Custom Embedded System

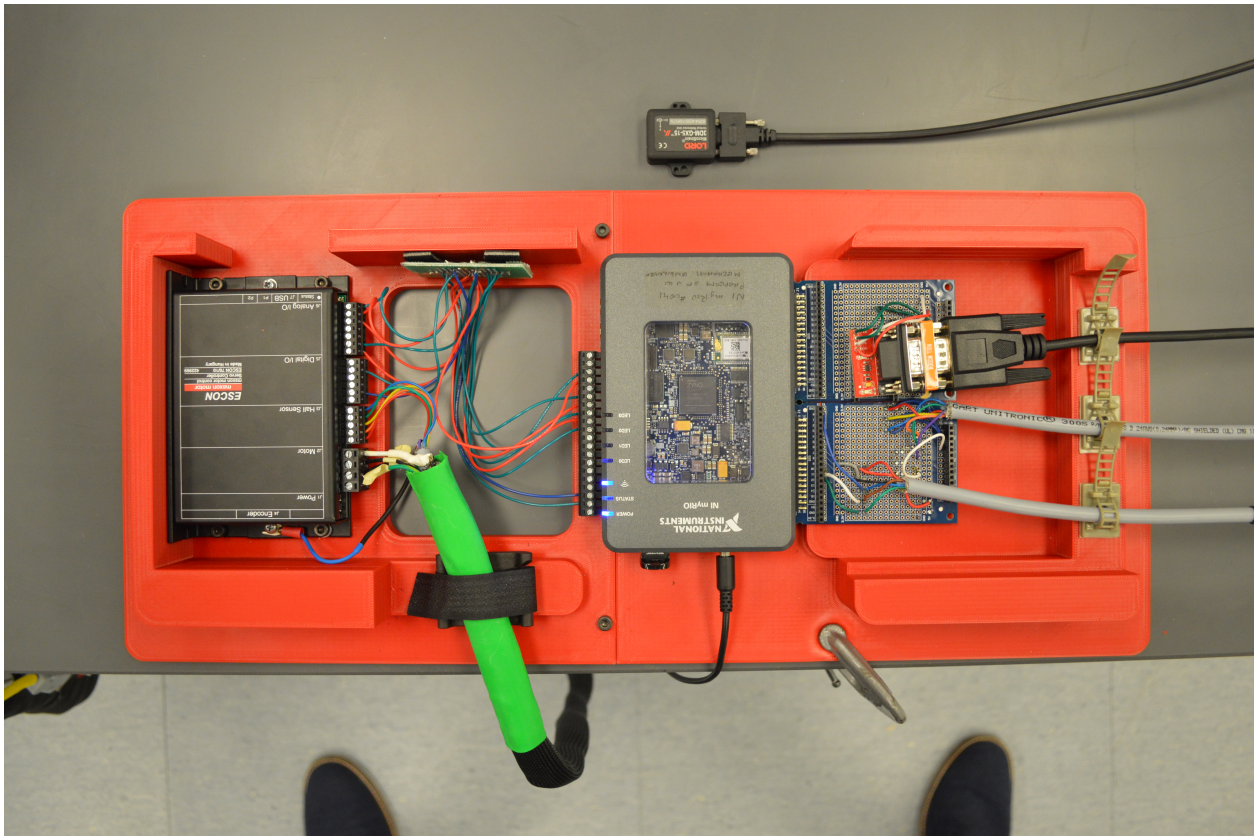


Figure A.3: Photo of the embedded system with microcontrollers and custom circuits secured to the 3D-printed case.

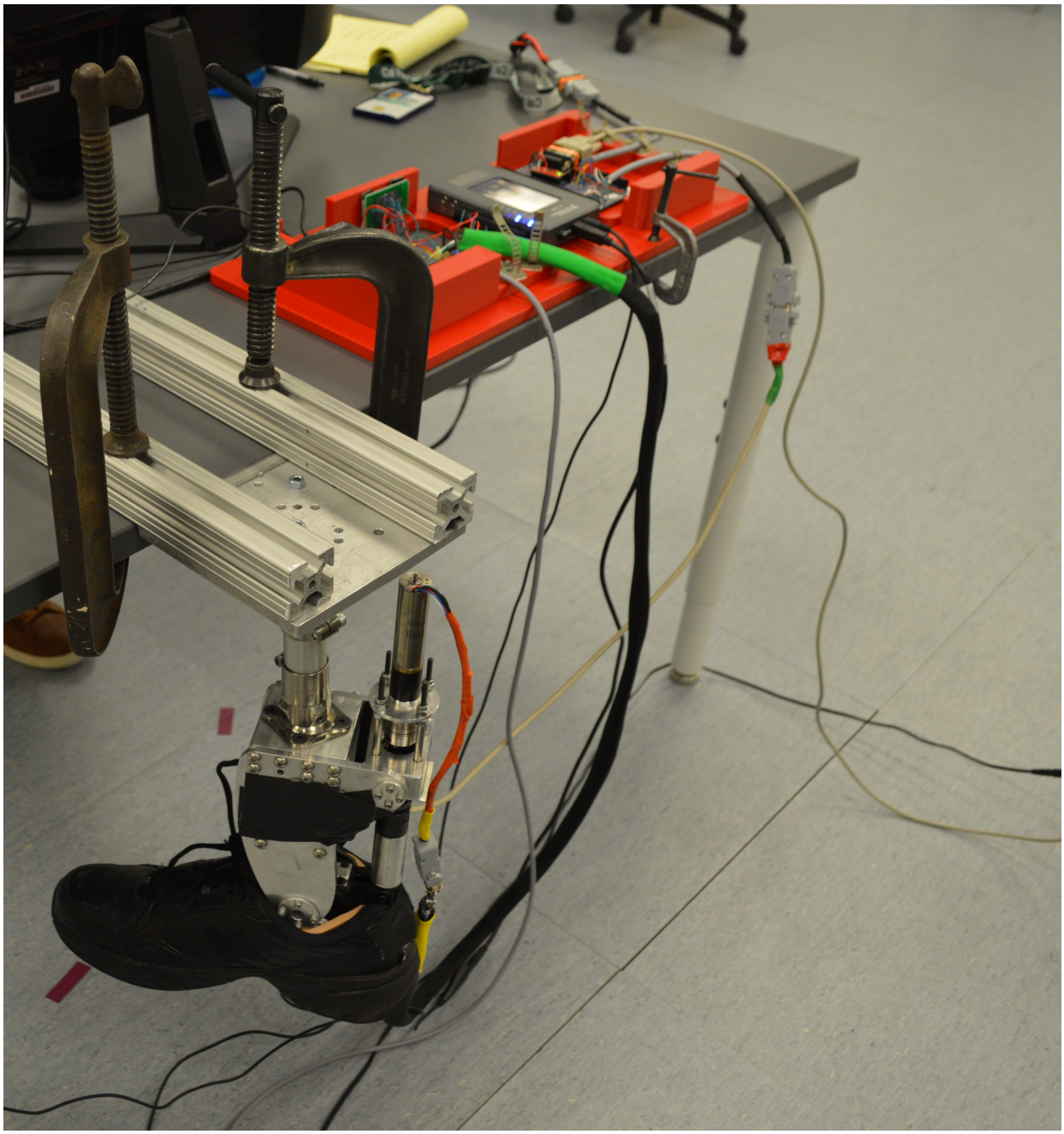


Figure A.4: A photo of the full embedded system and sensor interfaces during a benchtop test.

Appendix B

MATLAB CODE

This appendix contains the custom MATLAB scripts used to process data from experiments and conduct the offline learning procedures. The included scripts are listed with brief descriptions of their functionality. This code was modified from the biomech_tb toolbox: https://github.com/jonreal/biomechtb.

B.1 Data Processing Scripts

The first three scripts in this section process the data collected from the different systems used in the experimental procedure (Vicon, force plates, and myRIO device). The final script uses the previous three functions to process, label, and organize all data from a single experimental trial.

B.1.1 Motion Capture Data Processing Code

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

% This function parses vicon model csv files

nVarArgs = length(varargin);

% Default Vicon cutoff frequency for filtering
cutOff = 6;

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

% Plots
debug = 0;

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

fid = fopen(file,'r');

% Store trialName
% rtn.info.file = file;
```

```
37 [~,rtn.info.file,~] = fileparts(file);
38 rtn.params.processing_cutOff = cutOff;
39
40 % skip two words, go next line, grab sample freq
41 fs = fscanf(fid, '%*s %*s\n %f', 1);
42
43 rtn.params.fs = fs;
44
45 % skip line
46 tline = fgetl(fid);
47
48 % parse labels and create struct fields names
49 tline = fgetl(fid);
50 labels = textscan(tline, '%s', 'delimiter', ',');
51 [m, n] = size(labels{1});
52 labelCnt = 0;
53 for i=1:m
54     if ~(strcmp(labels{1}{i},''))
55         % pasre header on label
56         [token, remain] = strtok(labels{1}{i},':');
57         labelCnt = labelCnt + 1;
58         labelName{labelCnt} = remain(2:end);
59     end
60 end
61
62 % find number of subfields
63 tline = fgetl(fid);
64 sublabels = textscan(tline, '%s', 'delimiter', ',');
65 [m, n] = size(sublabels{1});
66
67 % skip line of units
68 fgetl(fid);
69
70 % put data in matrix
71 D = zeros(1,m);
72 formatSpec = [repmat(['%f'],1,m-1), '%f\n'];
73 rowCount = 1;
74 while(1)
75     C = textscan(fid, formatSpec, 'delimiter', ',');
76     dataRow = cell2mat(C);
77     if isempty(dataRow)
78         break;
79     end
80     D(rowCount,:) = cell2mat(C);
81     rowCount = rowCount + 1;
82 end
83
84 % Time = (frame number - 1) x dt
85 rtn.time = (D(:,1) - 1)*(1/fs);
86
```

```
87  % Butterworth 4th order
88  [b,a] = butter(4,2*(cutOff/fs));
89
90  % put data (X,Y,Z subfields per field)
91  % Fix units
92  % Pattern is (L/R)(joint)Angle, Moment, Power
93  for i=1:(labelCnt/3)
94
95      % Angle
            ----------------------------------------------------------------

96      jointAngle_X = deg2rad(D(:,3 + (9*i - 9)));
97      jointAngle_Y = deg2rad(D(:,4 + (9*i - 9)));
98      jointAngle_Z = deg2rad(D(:,5 + (9*i - 9)));
99
100     % Set NaN to zero
101     jointAngle_X(isnan(jointAngle_X)) = 0;
102     jointAngle_Y(isnan(jointAngle_Y)) = 0;
103     jointAngle_Z(isnan(jointAngle_Z)) = 0;
104
105     % Filter
106     jointAngle_X = filtfilt(b,a,jointAngle_X);
107     jointAngle_Y = filtfilt(b,a,jointAngle_Y);
108     jointAngle_Z = filtfilt(b,a,jointAngle_Z);
109
110     d_jointAngle_X = gradient(jointAngle_X)*fs;
111     d_jointAngle_Y = gradient(jointAngle_Y)*fs;
112     d_jointAngle_Z = gradient(jointAngle_Z)*fs;
113
114     d2_jointAngle_X = gradient(d_jointAngle_X)*fs;
115     d2_jointAngle_Y = gradient(d_jointAngle_Y)*fs;
116     d2_jointAngle_Z = gradient(d_jointAngle_Z)*fs;
117
118     % Moment
            ----------------------------------------------------------------

119     jointMoment_X = -D(:,6 + (9*i - 9))/1000;
120     jointMoment_Y = -D(:,7 + (9*i - 9))/1000;
121     jointMoment_Z = -D(:,8 + (9*i - 9))/1000;
122
123     % Set NaN to zero
124     jointMoment_X(isnan(jointMoment_X))= 0;
125     jointMoment_Y(isnan(jointMoment_Y))= 0;
126     jointMoment_Z(isnan(jointMoment_Z))= 0;
127
128     % Filter
129     jointMoment_X = filtfilt(b,a,jointMoment_X);
130     jointMoment_Y = filtfilt(b,a,jointMoment_Y);
131     jointMoment_Z = filtfilt(b,a,jointMoment_Z);
132
```

```
133     d_jointMoment_X = gradient(jointMoment_X)*fs;
134     d_jointMoment_Y = gradient(jointMoment_Y)*fs;
135     d_jointMoment_Z = gradient(jointMoment_Z)*fs;
136
137     d2_jointMoment_X = gradient(d_jointMoment_X)*fs;
138     d2_jointMoment_Y = gradient(d_jointMoment_Y)*fs;
139     d2_jointMoment_Z = gradient(d_jointMoment_Z)*fs;
140
141     % Power (calculate)
            ------------------------------------------------------
142     jointPower_X = d_jointAngle_X.*jointMoment_X;
143     jointPower_Y = d_jointAngle_Y.*jointMoment_Y;
144     jointPower_Z = d_jointAngle_Z.*jointMoment_Z;
145
146     d_jointPower_X = gradient(jointPower_X)*fs;
147     d_jointPower_Y = gradient(jointPower_Y)*fs;
148     d_jointPower_Z = gradient(jointPower_Z)*fs;
149
150     d2_jointPower_X = gradient(d_jointPower_X)*fs;
151     d2_jointPower_Y = gradient(d_jointPower_Y)*fs;
152     d2_jointPower_Z = gradient(d_jointPower_Z)*fs;
153
154     % Store Angle info
            ------------------------------------------------------
155     rtn.id.(labelName{1 + (3*i - 3)}).X = jointAngle_X;
156     rtn.id.(labelName{1 + (3*i - 3)}).Y = jointAngle_Y;
157     rtn.id.(labelName{1 + (3*i - 3)}).Z = jointAngle_Z;
158
159     % 1st time derivative
160     rtn.id.(['d_',labelName{1 + (3*i - 3)}]).X = d_jointAngle_X;
161     rtn.id.(['d_',labelName{1 + (3*i - 3)}]).Y = d_jointAngle_Y;
162     rtn.id.(['d_',labelName{1 + (3*i - 3)}]).Z = d_jointAngle_Z;
163
164     % 2nd time derivative
165     rtn.id.(['d2_',labelName{1 + (3*i - 3)}]).X = d2_jointAngle_X;
166     rtn.id.(['d2_',labelName{1 + (3*i - 3)}]).Y = d2_jointAngle_Y;
167     rtn.id.(['d2_',labelName{1 + (3*i - 3)}]).Z = d2_jointAngle_Z;
168
169     if(debug)
170         figure;
171         subplot(311); hold all;
172         title(['Saggital Plane ', labelName{1 + (3*i - 3)}],'fontsize'
                ,20)
173         plot(jointAngle_X,'k')
174         subplot(312); hold all;
175         plot(d_jointAngle_X,'k')
176         subplot(313); hold all;
177         plot(d2_jointAngle_X,'k')
178     end
179
```

```
180     % Store Moment info
            ------------------------------------------------------
181     rtn.id.(labelName{2 + (3*i - 3)}).X = jointMoment_X;
182     rtn.id.(labelName{2 + (3*i - 3)}).Y = jointMoment_Y;
183     rtn.id.(labelName{2 + (3*i - 3)}).Z = jointMoment_Z;
184
185     % 1st time derivative
186     rtn.id.(['d_',labelName{2 + (3*i - 3)}]).X = d_jointMoment_X;
187     rtn.id.(['d_',labelName{2 + (3*i - 3)}]).Y = d_jointMoment_Y;
188     rtn.id.(['d_',labelName{2 + (3*i - 3)}]).Z = d_jointMoment_Z;
189
190     % 2nd time derivative
191     rtn.id.(['d2_',labelName{2 + (3*i - 3)}]).X = d2_jointMoment_X;
192     rtn.id.(['d2_',labelName{2 + (3*i - 3)}]).Y = d2_jointMoment_Y;
193     rtn.id.(['d2_',labelName{2 + (3*i - 3)}]).Z = d2_jointMoment_Z;
194
195     if(debug)
196         figure;
197         subplot(311); hold all;
198         title(['Saggital Plane ', labelName{2 + (3*i - 3)}],'fontsize'
                ,20)
199         plot(jointMoment_X,'k')
200         subplot(312); hold all;
201         plot(d_jointMoment_X,'k')
202         subplot(313); hold all;
203         plot(d2_jointMoment_X,'k')
204     end
205
206     % Store Power info
            ------------------------------------------------------
207     rtn.id.(labelName{3 + (3*i - 3)}).X = jointPower_X;
208     rtn.id.(labelName{3 + (3*i - 3)}).Y = jointPower_Y;
209     rtn.id.(labelName{3 + (3*i - 3)}).Z = jointPower_Z;
210
211     % 1st time derivative
212     rtn.id.(['d_',labelName{3 + (3*i - 3)}]).X = d_jointPower_X;
213     rtn.id.(['d_',labelName{3 + (3*i - 3)}]).Y = d_jointPower_Y;
214     rtn.id.(['d_',labelName{3 + (3*i - 3)}]).Z = d_jointPower_Z;
215
216     % 2nd time derivative
217     rtn.id.(['d2_',labelName{3 + (3*i - 3)}]).X = d2_jointPower_X;
218     rtn.id.(['d2_',labelName{3 + (3*i - 3)}]).Y = d2_jointPower_Y;
219     rtn.id.(['d2_',labelName{3 + (3*i - 3)}]).Z = d2_jointPower_Z;
220
221     if(debug)
222         figure;
223         subplot(311); hold all;
224         title(['Saggital Plane ', labelName{3 + (3*i - 3)}],'fontsize'
                ,20)
225         plot(jointPower_X,'k')
```

```
226         subplot(312); hold all;
227         plot(d_jointPower_X,'k')
228         subplot(313); hold all;
229         plot(d2_jointPower_X,'k')
230     end
231
232 end
233
234 % CHANGE
235 % Marker Trajectories
236 % skip line
237 tline = fgetl(fid);
238
239 % skip line
240 tline = fgetl(fid);
241
242 % parse labels and create struct fields names
243 tline = fgetl(fid);
244 labels = textscan(tline, '%s', 'delimiter', ',');
245 [m, n] = size(labels{1});
246 labelCnt = 0;
247 for i=1:m
248     if ~(strcmp(labels{1}{i},''))
249         % pasre header on label
250         [token, remain] = strtok(labels{1}{i},':');
251         labelCnt = labelCnt + 1;
252         labelName{labelCnt} = remain(2:end);
253     end
254 end
255
256 % find number of subfields
257 tline = fgetl(fid);
258 sublabels = textscan(tline, '%s', 'delimiter', ',');
259 [m, n] = size(sublabels{1});
260
261 % skip line of units
262 fgetl(fid);
263
264 % put data in matrix
265 D = zeros(1,m);
266 formatSpec = [repmat(['%f'],1,m-1), '%f\n'];
267 rowCount = 1;
268 while(1)
269     C = textscan(fid, formatSpec, 'delimiter', ',');
270     dataRow = cell2mat(C);
271     if isempty(dataRow)
272         break;
273     end
274     D(rowCount,:) = cell2mat(C);
275     rowCount = rowCount + 1;
```

```
276 end
277
278 % put data (X,Y,Z subfields per field)
279 for i=1:labelCnt
280     rtn.markers.(labelName{i}).X = D(:,3 + (3*i - 3));
281     rtn.markers.(labelName{i}).Y = D(:,4 + (3*i - 3));
282     rtn.markers.(labelName{i}).Z = D(:,5 + (3*i - 3));
283 end
284
285 fclose(fid);
286 end
```

B.1.2 Force Plate Data Processing Code

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


nVarArgs = length(varargin);

% Default input values
cutOff = 60;
eventThreshold = 50;

% Collect and interpret function inputs
for i=1:2:nVarArgs
    switch varargin{i}
        case 'cutOff'
            cutOff = varargin{i+1};
        case 'eventThreshold'
            eventThreshold = varargin{i+1};
        case 'processVicon'
            processVicon = varargin{i+1};
        case 'embData'
            emb = varargin{i+1};
        otherwise
            fprintf('\n%s option not found!\n',varargin{i});
            return
    end
end

% Collect params and info
rtn.params.processing_cutOff = cutOff;
rtn.params.eventThreshold = eventThreshold;

if (processVicon)

    rtn.info.useEmb = 0;

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

    fid = fopen(file,'r');

    % skip line
```

```
49      tline = fgetl(fid);
50      fs =fscanf(fid,'%f', 1);
51
52      rtn.params.fs = fs;
53
54      fclose(fid);
55
56      % Data
57      D = csvread(file,5,0);
58      frame_ = D(:,1);
59      subframe_ = D(:,2);
60      frame = (frame_ - 1).*10 + subframe_;
61
62      % Butterworth 4th order
63      [b,a] = butter(4,2*(cutOff/fs));
64
65      % Filter/Store data
66      rtn.time = frame.*(1/fs);
67      rtn.r.Fx = filtfilt(b,a,D(:,3));
68      rtn.r.Fy = filtfilt(b,a,D(:,4));
69      if mean(D(:,5)) < 0
70          rtn.l.Fz = -filtfilt(b,a,D(:,5)); % CHANGE
71      else
72          rtn.l.Fz = filtfilt(b,a,D(:,5)); % CHANGE
73      end
74      rtn.l.Fx = filtfilt(b,a,D(:,6));
75      rtn.l.Fy = filtfilt(b,a,D(:,7));
76      if mean(D(:,8)) < 0
77          rtn.r.Fz = -filtfilt(b,a,D(:,8)); % CHANGE
78      else
79          rtn.r.Fz = filtfilt(b,a,D(:,8)); % CHANGE
80      end
81  else
82
83      rtn.info.useEmb = 1;
84
85      % Butterworth 4th order
86      rtn.time = emb.time;
87      fs = 1/(rtn.time(2) - rtn.time(1));
88      [b,a] = butter(4,2*(cutOff/fs));
89      rtn.params.fs = fs;
90
91      % Filter/Store data
92      rtn.r.Fz = filtfilt(b,a,emb.Right_Ground_Reaction_Force);
93      rtn.l.Fz = filtfilt(b,a,emb.Left_Ground_Reaction_Force);
94
95  end
96
97  % Gait events
98  % Right
```

```
 99 threshold = (rtn.r.Fz < eventThreshold);
100 dthreshold = threshold(2:end) - threshold(1:end-1);
101
102 to = find(dthreshold == 1);
103 hs = find(dthreshold == -1);
104
105 if(1)
106     figure; hold all;
107     plot(rtn.time,rtn.r.Fz);
108     plot(rtn.time(hs),rtn.r.Fz(hs),'ok')
109     plot(rtn.time(to),rtn.r.Fz(to),'or')
110 end
111
112 rtn.ge.r.hs_time(:,1) = rtn.time(hs);
113 rtn.ge.r.to_time(:,1) = rtn.time(to);
114 rtn.ge.r.hs_ind(:,1) = hs;
115 rtn.ge.r.to_ind(:,1) = to;
116
117 % Left
118 threshold = (rtn.l.Fz < eventThreshold);
119 dthreshold = threshold(2:end) - threshold(1:end-1);
120
121 to = find(dthreshold == 1);
122 hs = find(dthreshold == -1);
123
124 if(1)
125     figure; hold all;
126     plot(rtn.time,rtn.l.Fz);
127     plot(rtn.time(hs),rtn.l.Fz(hs),'ok')
128     plot(rtn.time(to),rtn.l.Fz(to),'or')
129 end
130
131 % Collect gait event times and array indices
132 rtn.ge.l.hs_time(:,1) = rtn.time(hs);
133 rtn.ge.l.to_time(:,1) = rtn.time(to);
134 rtn.ge.l.hs_ind(:,1) = hs;
135 rtn.ge.l.to_ind(:,1) = to;
136
137
138 end
```

B.1.3 myRIO Data Processing Code

```
function rtn = cp_labview_process_data(trialName,varargin)
% Process logged data from LabVIEW Real-Time VIs
% note: need "convertTDMS" function from Matlab File Exchange

nVarArgs = length(varargin);

% Defaults
k_t = 10.2/1000;      % 7.75/1000;        % torque constant - data sheet
%  k_s = 1230;          % speed constant - data sheet
k_s = 940; % 1223;    % maxon calibration (may be different than
   datasheet)
Vcc = 60;             % power supply voltage
maxMotorAmp = 14;     % setting in maxon
maxMotorRpm = 80000;  % setting in maxon
cutOff = 50;          % cutoff freq. for filtering
R_T = 1806;           % estimated from motor/ankle velocity data
syncVicon = 1;        % sync vicon data with sync pulse

% Collect and interpret input data
for i=1:2:nVarArgs
    switch varargin{i}
        case 'k_t'
            % torque constant (Nm/A)
            k_tau = varargin{i+1};
        case 'k_s'
            % speed constant (rpm/V)
            k_s = varargin{i+1};
        case 'Vcc'
            % operating voltage (V)
            Vcc = varargin{i+1};
        case 'R_T'
            % total tranmission ratio
            R_T = varargin{i+1};
        case 'maxMotorAmp'
            % max motor current (A) -- setting on ESCON 70/10 servo
               controller
            maxMotorAmp = varargin{i+1};
        case 'maxMotorRpm'
            % max motor speed (Rpm) -- setting on ESCON 70/10 servo
               controller
            maxMotorRpm = varargin{i+1};
        case 'cutOff'
            % filter cutoff frequency, 4-th order butterworth low-pass
            cutOff = varargin{i+1};
        case 'oldVersion'
            %  Old parameter format
            oldVersion = varargin{i+1};
        case 'syncVicon'
```

```
46              %  sync pin
47              syncVicon = varargin{i+1};
48          otherwise
49              fprintf('\n%s option not found!\n',varargin{i});
50              return
51      end
52  end
53
54  doplot = 1;
55
56  %
57  %% parse embedded parameters
58  %
59
60  % Check if metadata file exists
61  % metafile = ['./',trialName,'_metadata.tdms'];
62  metafile = [trialName,'_metadata.tdms'];
63  if exist(metafile,'file') ~= 2
64      fprintf('\n\tMetadata file not found\n');
65      rtn = []
66      return
67  end
68  % [t,~,meta_ChanNames] = convertTDMS(false, 'filename', fullfile(pwd,
        metafile));
69  [t,~,meta_ChanNames] = convertTDMS(false, 'filename', metafile);
70  for ii = 3:length(t.Data.MeasuredData)
71      group = extractBefore(t.Data.MeasuredData(ii).Name,'/');
72      t.Data.MeasuredData(ii).Name = erase(t.Data.MeasuredData(ii).Name,[
            group,'/']);
73  end
74  for ii = 1:length(meta_ChanNames{1})
75      group = extractBefore(meta_ChanNames{1}{ii},'/');
76      meta_ChanNames{1}{ii} = erase(meta_ChanNames{1}{ii},[group,'/']);
77  end
78  meta_ChanNames = meta_ChanNames{1};
79  t.Data.MeasuredData = t.Data.MeasuredData(3:end);
80  % store data in struct
81  [~,q] = size(t.Data.MeasuredData);
82  for i=1:q
83      name = strrep(t.Data.MeasuredData(i).Name, ' ', '_');
84      name = regexp(name,'^\w+','once','match');
85      if name(end) == '_'; name = name(1:end-1); end
86      rtn.params.(name) = t.Data.MeasuredData(i).Data;
87  end
88
89  fs = rtn.params.Sampling_Freq;
90
91  % EDIT --> first line is date and time
92  % rtn.info.date = fscanf(fid,'#Date: %s',1);
93  % rtn.info.time = fscanf(fid,'%s\n',1);
```

```
94  % rtn.info.file = trialName;
95
96  % System parameters (inputted into this function)
97  rtn.params.motor.k_t = k_t;
98  rtn.params.motor.k_s = k_s;
99  rtn.params.motor.Vcc = Vcc;
100 rtn.params.motor.R_T = R_T;
101 rtn.params.motor.maxMotorAmp = maxMotorAmp;
102 rtn.params.motor.maxMotorRpm = maxMotorRpm;
103
104 %
105 %% read logged data
106 %
107
108 % Check file exists
109 % file = ['./',trialName,'_LabView.tdms']; % ['./',trialName];
110 file = [trialName,'_LabView.tdms'];
111 if exist(file,'file') ~= 2
112     fprintf('\n\tEmbedded file not found\n');
113     rtn = []
114     return
115 end
116
117 % Open and Process Datafile
118 % [s,~,ChanNames,~,~] = convertTDMS(false, 'filename', fullfile(pwd,
        file));
119 [s,~,ChanNames,~,~] = convertTDMS(false, 'filename', file);
120 for ii = 3:length(s.Data.MeasuredData)
121     group = extractBefore(s.Data.MeasuredData(ii).Name,'/');
122     s.Data.MeasuredData(ii).Name = erase(s.Data.MeasuredData(ii).Name,[
          group,'/']);
123 end
124 for ii = 1:length(ChanNames{1})
125     group = extractBefore(ChanNames{1}{ii},'/');
126     ChanNames{1}{ii} = erase(ChanNames{1}{ii},[group,'/']);
127 end
128 ChanNames = ChanNames{1};
129 s.Data.MeasuredData = s.Data.MeasuredData(3:end);
130
131 % Size check
132 [~,n] = size(s.Data.MeasuredData);
133 nn = numel(ChanNames);
134
135 if nn ~= n
136     [nn, n]
137     error('Labels/column mismatch!');
138     return
139 end
140
141
```

```
142 % Find if/where data bits were dropped
143 % Time stamp difference: dt = 1 for all points unless data chunk is
       missing
144 %
145 % This happens, I think, because the kernel interrupts during circular
       buffer
146 % writes.
147
148 % if(doplot)
149 %     figure;
150 %     plot(D(:,1))
151 %     xlabel('Data Point','fontsize',20)
152 %     ylabel('Time Stamp','fontsize',20)
153 % end
154 %
155 % dt = D(2:end,1) - D(1:end-1,1);
156 % startMissingData_index = find(dt ~= 1);
157 % endMissingData_index = startMissingData_index + 1;
158 %
159 % if(doplot)
160 %     figure;
161 %     plot(1:numel(D(1:end-1,1)),dt)
162 %     xlabel('Data Point','fontsize',20)
163 %     ylabel('Stamp[k+1] - Stamp[k]','fontsize',20)
164 % end
165
166 % calculate time vector
167 cnt0 = 0;
168 cntEnd = s.Data.MeasuredData(1).Total_Samples * 1/fs;
169 rtn.data.time = linspace(cnt0, cntEnd, s.Data.MeasuredData(1).
       Total_Samples)';
170 time = rtn.data.time;
171
172 % store data in struct
173 for i=1:n
174     if contains(s.Data.MeasuredData(i).Name, 'STTTM') || contains(s.Data
          .MeasuredData(i).Name, 'Stiffness')
175     else
176         name = strrep(s.Data.MeasuredData(i).Name, ' ', '_');
177         name = regexp(name,'^\w+','once','match');
178         if name(end) == '_'; name = name(1:end-1); end
179         rtn.data.(name) = s.Data.MeasuredData(i).Data;
180     end
181 end
182
183 % butterworth 4th order
184 [b,a] = butter(4,2*(cutOff/fs));
185
186 % remove repeated hs
187 l_hs = find(rtn.data.Left_Heel_Strike_Boolean);
```

```
188 l_hs(l_hs < cnt0) = [];
189 r_hs = find(rtn.data.Right_Heel_Strike_Boolean);
190 r_hs(r_hs < cnt0) = [];
191
192 % collect HS indices
193 rtn.ge.l.hs_ind = l_hs;
194 rtn.ge.r.hs_ind = r_hs;
195
196 % find HS time
197 rtn.ge.l.hs_time = (l_hs - cnt0).*(1/fs);
198 rtn.ge.r.hs_time = (r_hs - cnt0).*(1/fs);
199
200 % Find TO from changes in state
201
202
203 % store important data
204 rtn.control.time = rtn.data.time;
205 rtn.control.u = rtn.data.Desired_Motor_Current;
206 rtn.control.i_m = rtn.data.True_Motor_Current;
207 rtn.control.omega_m_rpm = rtn.data.Motor_Velocity;
208 rtn.control.omega_m = rpm2radps(rtn.control.omega_m_rpm);
209 rtn.control.d_omega_m = gradient(rtn.control.omega_m)*fs;
210 rtn.control.tau_m = rtn.control.i_m.*k_t;
211 rtn.control.R_Tau = rtn.data.Transmission_Ratio;
212 rtn.control.tau_a = rtn.control.tau_m.*rtn.control.R_Tau;
213 rtn.control.V_m = rtn.control.omega_m_rpm/k_s;
214 rtn.control.P_m_elect = rtn.control.i_m.*rtn.control.V_m;
215 rtn.control.P_m_mech = rtn.control.tau_m.*rtn.control.omega_m;
216 rtn.control.ankPos = filtfilt(b,a,rtn.data.Ankle_Encoder_Angle);
217 rtn.control.d_ankPos = gradient(rtn.control.ankPos)*fs;
218 rtn.control.d2_ankPos = gradient(rtn.control.d_ankPos)*fs;
219 rtn.control.imu_roll = rtn.data.IMU_Roll_Angle;
220 rtn.control.imu_pitch = rtn.data.IMU_Pitch_Angle;
221 rtn.control.imu_yaw = rtn.data.IMU_Yaw_Angle;
222 rtn.control.thigh_angle = rtn.data.Thigh_Angle;
223 rtn.control.Phi = rtn.data.Phi;
224 rtn.control.GRF_l = rtn.data.Left_Ground_Reaction_Force;
225 rtn.control.GRF_r = rtn.data.Right_Ground_Reaction_Force;
226 rtn.control.GRF_l_filt = rtn.data.Filtered_Left_GRF;
227 rtn.control.GRF_r_filt = rtn.data.Filtered_Right_GRF;
228 rtn.control.dGRF_l = rtn.data.Left_GRF_Rate;
229 rtn.control.dGRF_r = rtn.data.Right_GRF_Rate;
230 rtn.control.theta_v = rtn.data.Desired_Ankle_Angle_Trajectory;
231 rtn.control.tau_ref = rtn.data.Torque_Reference;
232 rtn.control.tau_passive = rtn.data.Passive_Torque_Estimate;
233
234 end
```

B.1.4 Experimental Trial Data Processing Code

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

nVarArgs = length(varargin);

% Defaults
processEmb = 1; % yes = 1, no = 0
processVicon = 1;

% Motor 3 datasheet params
k_t = 10.2/1000;
k_s = 940;
Vcc = 60;
R_T = 1800;
maxMotorAmp = 14;
maxMotorRpm = 80000;

% Default cutoff freqs and Thresholds
modelCutOff = 6;
GRFCutOff = 50;
embCutOff = 50;
eventThreshold = 50;

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

```
                controller
48              maxMotorRpm = varargin{i+1};
49          case 'modelCutOff'
50              modelCutOff = varargin{i+1};
51          case 'GRFCutOff'
52              GRFCutOff = varargin{i+1};
53          case 'embCutOff'
54              embCutOff = varargin{i+1};
55          case 'eventThreshold'
56              eventThreshold = varargin{i+1};
57          otherwise
58              fprintf('\n%s option not found!\n',varargin{i});
59              return
60      end
61  end
62
63  %% Process subject params
64  % rtn.subject = cp_vicon_process_subject(trialName);
65
66  %% Process inverse dynamics model
67  if (processVicon)
68
69      model = cp_vicon_process_model(trialName, ...
70          'cutOff',modelCutOff);
71
72      rtn.info.model = model.info;
73      rtn.params.model = model.params;
74      rtn.time = model.time;
75      rtn.model = model.id;
76      rtn.markers = model.markers;
77
78  end
79
80  %% Process embedded data
81  if (processEmb)
82      emb = cp_labview_process_data(trialName, ...
83          'k_t',k_t, ...
84          'k_s',k_s, ...
85          'Vcc',Vcc, ...
86          'R_T',R_T, ...
87          'maxMotorAmp',maxMotorAmp, ...
88          'maxMotorRpm',maxMotorRpm, ...
89          'cutOff',embCutOff);
90
91      rtn.params.emb = emb.params;
92      rtn.control = emb.control;
93      rtn.ge.emb = emb.ge;
94      rtn.emb = emb.data;
95      % rtn.info.emb = emb.info; % EDIT
96
```

```
 97 end
 98
 99
100 %
101 %
102 %% Process vGRF
103
104 grf = cp_vicon_process_grf(trialName, ...
105     'cutOff',GRFCutOff, ...
106     'eventThreshold',eventThreshold,...
107     'processVicon',processVicon,...
108     'embData',emb.data);
109
110 rtn.info.grf = grf.info;
111 rtn.params.grf = grf.params;
112 rtn.grf.time = grf.time;
113 rtn.grf.r = grf.r;
114 rtn.grf.l = grf.l;
115 ge = grf.ge;
116
117 % Process HS
118 for foot = {'l','r'}
119
120     % Remove TO before first HS
121     ge.(foot{:}).to_time(ge.(foot{:}).to_time < ge.(foot{:}).hs_time(1))
          = [];
122
123     % Remove TO after last HS
124     ge.(foot{:}).to_time(ge.(foot{:}).to_time > ge.(foot{:}).hs_time(end
        )) = [];
125
126     % Reformat
127     segment.(foot{:}).gait.time =  ...
128         [ge.(foot{:}).hs_time(1:end-1), ge.(foot{:}).hs_time(2:end)];
129
130     segment.(foot{:}).stance.time =  ...
131         [ge.(foot{:}).hs_time(1:end-1), ge.(foot{:}).to_time(1:end)];
132
133     segment.(foot{:}).swing.time =  ...
134         [ge.(foot{:}).to_time(1:end), ge.(foot{:}).hs_time(2:end)];
135
136 end
137
138 rtn.ge.grf = ge;
139 rtn.segment.grf = segment;
140
141 %% Time normalize -- Whole Gait Cycle
142 %
143 %
144
```

```
145
146 % Gait cycle
147 gaitCycle = linspace(0,100,1001);
148 rtn.gait.gaitCycle = gaitCycle;
149
150 % Field names
151 if (processVicon)
152     q_name = fieldnames(rtn.model);
153 end
154
155 q_grf = fieldnames(rtn.grf.l);
156
157 % Process embedded data
158 if (processEmb)
159     q_control = fieldnames(rtn.control);
160 end
161
162 % Time Normalization
163 for foot = {'l','r'}
164
165     hs_time = rtn.segment.grf.(foot{:}).gait.time;
166
167     for i=1:numel(hs_time(:,1))
168
169         if (processVicon)
170             % If recorded hs occurs before model data, skip
171             if (hs_time(i,1) < rtn.time(1)) | (hs_time(i,2) > rtn.time(
                    end))
172                 continue;
173             end
174
175             % Find vicon time stamp corresponding to HS
176             temp = rtn.time - hs_time(i,1);
177             temp(temp < 0) = inf;
178             [~,ii] = min(temp);
179             hs1 = ii;
180
181             temp = rtn.time - hs_time(i,2);
182             temp(temp < 0) = inf;
183             [~,ii] = min(temp);
184             hs2 = ii;
185
186             rawTime = rtn.time(hs1:hs2);
187             rawGaitCylce = (rawTime - rawTime(1))./(rawTime(end) -
                    rawTime(1))*100;
188
189             rtn.gait.model.normalized.(foot{:}).time{i} = rawTime;
190
191             % Interpolate inverse dynamics (vicon)
192
```

```
193             for j=1:numel(q_name)
194                 rtn.gait.model.normalized.(foot{:}).(q_name{j}).X(:,i) =
                        ...
195                     interp1(rawGaitCylce, rtn.model.(q_name{j}).X(hs1:
                            hs2), ...
196                     gaitCycle, 'spline');
197                 rtn.gait.model.normalized.(foot{:}).(q_name{j}).Y(:,i) =
                        ...
198                     interp1(rawGaitCylce, rtn.model.(q_name{j}).Y(hs1:
                            hs2), ...
199                     gaitCycle, 'spline');
200                 rtn.gait.model.normalized.(foot{:}).(q_name{j}).Z(:,i) =
                        ...
201                     interp1(rawGaitCylce, rtn.model.(q_name{j}).Z(hs1:
                            hs2), ...
202                     gaitCycle, 'spline');
203             end
204
205         end
206
207         % Find grf time stamp corresponding to HS
208         temp = rtn.grf.time - hs_time(i,1);
209         temp(temp < 0) = inf;
210         [~,ii] = min(temp);
211         hs1 = ii;
212
213         temp = rtn.grf.time - hs_time(i,2);
214         temp(temp < 0) = inf;
215         [~,ii] = min(temp);
216         hs2 = ii;
217
218         rawTime = rtn.grf.time(hs1:hs2);
219         rawGaitCylce = (rawTime - rawTime(1))./(rawTime(end) - rawTime
                (1))*100;
220
221         rtn.gait.grf.normalized.(foot{:}).grf.time{i} = rawTime;
222
223         % Interpolate vGRF
224         for j=1:numel(q_grf)
225
226             rtn.gait.grf.normalized.(foot{:}).(['L',q_grf{j}])(:,i) =
                    ...
227                 interp1(rawGaitCylce, rtn.grf.l.(q_grf{j})(hs1:hs2), ...
228                 gaitCycle, 'spline');
229             rtn.gait.grf.normalized.(foot{:}).(['R',q_grf{j}])(:,i) =
                    ...
230                 interp1(rawGaitCylce, rtn.grf.r.(q_grf{j})(hs1:hs2), ...
231                 gaitCycle, 'spline');
232         end
233
```

```
234
235
236         % Interpolate embedded system data
237         if (processEmb)
238
239             % Find emb time stamp corresponding to HS
240             temp = rtn.control.time - hs_time(i,1);
241             temp(temp < 0) = inf;
242             [~,ii] = min(temp);
243             hs1 = ii;
244
245             temp = rtn.control.time - hs_time(i,2);
246             temp(temp < 0) = inf;
247             [~,ii] = min(temp);
248             hs2 = ii;
249
250             rawTime = rtn.control.time(hs1:hs2);
251             if isempty(rawTime) || sum(eq(rawTime,0))
252
253             else
254                 rawGaitCylce = (rawTime - rawTime(1))./(rawTime(end) -
                        rawTime(1))*100;
255
256                 rtn.gait.emb.normalized.(foot{:}).control.time{i} =
                        rawTime;
257
258                 for j=2:numel(q_control)
259                     rtn.gait.emb.normalized.(foot{:}).(q_control{j})(:,i
                            ) = ...
260                         interp1(rawGaitCylce, rtn.control.(q_control{j})
                                (hs1:hs2), ...
261                         gaitCycle, 'linear');
262                 end
263             end
264         end
265
266     end
267
268     % Mean, std
269     if (processVicon)
270         for j=1:numel(q_name)
271             jointVarMean_X =  ...
272                 mean(rtn.gait.model.normalized.(foot{:}).(q_name{j}).X,
                        2);
273             jointVarMean_Y = ...
274                 mean(rtn.gait.model.normalized.(foot{:}).(q_name{j}).Y,
                        2);
275             jointVarMean_Z =  ...
276                 mean(rtn.gait.model.normalized.(foot{:}).(q_name{j}).Z,
                        2);
```

```
277
278             jointVarStd_X =  ...
279                 std(rtn.gait.model.normalized.(foot{:}).(q_name{j}).X')
                        ';
280             jointVarStd_Y = ...
281                 std(rtn.gait.model.normalized.(foot{:}).(q_name{j}).Y')
                        ';
282             jointVarStd_Z = ...
283                 std(rtn.gait.model.normalized.(foot{:}).(q_name{j}).Z')
                        ';
284
285             rtn.gait.model.(foot{:}).(q_name{j}).X = ...
286                 [jointVarMean_X + jointVarStd_X, ...
287                 jointVarMean_X, ...
288                 jointVarMean_X - jointVarStd_X];
289
290             rtn.gait.model.(foot{:}).(q_name{j}).Y = ...
291                 [jointVarMean_Y + jointVarStd_Y, ...
292                 jointVarMean_Y, ...
293                 jointVarMean_Y - jointVarStd_Y];
294
295             rtn.gait.model.(foot{:}).(q_name{j}).Z = ...
296                 [jointVarMean_Z + jointVarStd_Z, ...
297                 jointVarMean_Z, ...
298                 jointVarMean_Z - jointVarStd_Z];
299         end
300     end
301
302     % Mean, std
303     for j=1:numel(q_grf)
304         varMean = ...
305             mean(rtn.gait.grf.normalized.(foot{:}).(['L',q_grf{j}]), 2);
306
307         varStd = ...
308             std(rtn.gait.grf.normalized.(foot{:}).(['L',q_grf{j}])')';
309
310         rtn.gait.grf.(foot{:}).(['L',q_grf{j}]) = [varMean + varStd, ...
311             varMean, ...
312             varMean - varStd];
313
314         varMean = ...
315             mean(rtn.gait.grf.normalized.(foot{:}).(['R',q_grf{j}]), 2);
316
317         varStd = ...
318             std(rtn.gait.grf.normalized.(foot{:}).(['R',q_grf{j}])')';
319
320         rtn.gait.grf.(foot{:}).(['R',q_grf{j}]) = [varMean + varStd, ...
321             varMean, ...
322             varMean - varStd];
323     end
```

```
324
325     % Mean, std
326     if (processEmb)
327         for j=2:numel(q_control)
328             varMean = mean(rtn.gait.emb.normalized.(foot{:}).(q_control{
                    j}), 2);
329             varStd = std(rtn.gait.emb.normalized.(foot{:}).(q_control{j
                    })')';
330
331             rtn.gait.emb.(foot{:}).(q_control{j}) = [varMean + varStd,
                    ...
332                 varMean, ...
333                 varMean - varStd];
334         end
335     end
336
337 end
338
339
340 %
341 %
342 %% Time normalize -- Stance Cycle
343 %
344 %
345
346
347 % Gait cycle
348 stanceCycle = linspace(0,100,1001);
349 rtn.stance.stanceCycle = stanceCycle;
350
351 % Time Normalization
352 for foot = {'l','r'}
353
354     hs_time = rtn.segment.grf.(foot{:}).stance.time;
355
356     for i=1:numel(hs_time(:,1))
357
358         if (processVicon)
359             % If recorded hs occurs before model data, skip
360             if hs_time(i,1) < rtn.time(1)
361                 continue;
362             end
363
364             % Find vicon time stamp corresponding to HS
365             temp = rtn.time - hs_time(i,1);
366             temp(temp < 0) = inf;
367             [~,ii] = min(temp);
368             hs1 = ii;
369
370             temp = rtn.time - hs_time(i,2);
```

```
371             temp(temp < 0) = inf;
372             [~,ii] = min(temp);
373             hs2 = ii;
374 
375             rawTime = rtn.time(hs1:hs2);
376             rawStanceCycle = (rawTime - rawTime(1))./(rawTime(end) -
                    rawTime(1))*100;
377 
378             rtn.gait.model.normalized.(foot{:}).time{i} = rawTime;
379 
380             % Interpolate inverse dynamics (vicon)
381             for j=1:numel(q_name)
382                 rtn.stance.model.normalized.(foot{:}).(q_name{j}).X(:,i)
                        = ...
383                     interp1(rawStanceCycle, rtn.model.(q_name{j}).X(hs1:
                            hs2), ...
384                     stanceCycle, 'spline');
385                 rtn.stance.model.normalized.(foot{:}).(q_name{j}).Y(:,i)
                        = ...
386                     interp1(rawStanceCycle, rtn.model.(q_name{j}).Y(hs1:
                            hs2), ...
387                     stanceCycle, 'spline');
388                 rtn.stance.model.normalized.(foot{:}).(q_name{j}).Z(:,i)
                        = ...
389                     interp1(rawStanceCycle, rtn.model.(q_name{j}).Z(hs1:
                            hs2), ...
390                     stanceCycle, 'spline');
391             end
392         end
393 
394         % Find grf time stamp corresponding to HS
395         temp = rtn.grf.time - hs_time(i,1);
396         temp(temp < 0) = inf;
397         [~,ii] = min(temp);
398         hs1 = ii;
399 
400         temp = rtn.grf.time - hs_time(i,2);
401         temp(temp < 0) = inf;
402         [~,ii] = min(temp);
403         hs2 = ii;
404 
405         rawTime = rtn.grf.time(hs1:hs2);
406         rawStanceCycle = (rawTime - rawTime(1))./(rawTime(end) - rawTime
                (1))*100;
407 
408         rtn.stance.grf.normalized.(foot{:}).grf.time{i} = rawTime;
409 
410         % Interpolate vGRF
411         for j=1:numel(q_grf)
412 
```

```
413             rtn.stance.grf.normalized.(foot{:}).(['L',q_grf{j}])(:,i) =
                    ...
414                 interp1(rawStanceCycle, rtn.grf.l.(q_grf{j})(hs1:hs2),
                        ...
415                 stanceCycle, 'spline');
416             rtn.stance.grf.normalized.(foot{:}).(['R',q_grf{j}])(:,i) =
                    ...
417                 interp1(rawStanceCycle, rtn.grf.r.(q_grf{j})(hs1:hs2),
                        ...
418                 stanceCycle, 'spline');
419         end
420
421
422         % Interpolate embedded system
423         if (processEmb)
424
425             % Find emb time stamp corresponding to HS
426             temp = rtn.control.time - hs_time(i,1);
427             temp(temp < 0) = inf;
428             [~,ii] = min(temp);
429             hs1 = ii;
430
431             temp = rtn.control.time - hs_time(i,2);
432             temp(temp < 0) = inf;
433             [~,ii] = min(temp);
434             hs2 = ii;
435
436             rawTime = rtn.control.time(hs1:hs2);
437             if isempty(rawTime) || sum(eq(rawTime,0))
438
439             else
440                 rawStanceCycle = (rawTime - rawTime(1))./(rawTime(end) -
                        rawTime(1))*100;
441
442                 rtn.stance.emb.normalized.(foot{:}).control.time{i} =
                        rawTime;
443
444                 for j=2:numel(q_control)
445                     rtn.stance.emb.normalized.(foot{:}).(q_control{j})
                            (:,i) = ...
446                         interp1(rawStanceCycle, rtn.control.(q_control{j
                            })(hs1:hs2), ...
447                         stanceCycle, 'linear');
448                 end
449             end
450         end
451
452     end
453
454     if (processVicon)
```

```
455         % Mean, std
456         for j=1:numel(q_name)
457             jointVarMean_X =  ...
458                 mean(rtn.stance.model.normalized.(foot{:}).(q_name{j}).X
                        , 2);
459             jointVarMean_Y = ...
460                 mean(rtn.stance.model.normalized.(foot{:}).(q_name{j}).Y
                        , 2);
461             jointVarMean_Z =  ...
462                 mean(rtn.stance.model.normalized.(foot{:}).(q_name{j}).Z
                        , 2);
463
464             jointVarStd_X =  ...
465                 std(rtn.stance.model.normalized.(foot{:}).(q_name{j}).X
                        ')';
466             jointVarStd_Y = ...
467                 std(rtn.stance.model.normalized.(foot{:}).(q_name{j}).Y
                        ')';
468             jointVarStd_Z = ...
469                 std(rtn.stance.model.normalized.(foot{:}).(q_name{j}).Z
                        ')';
470
471             rtn.stance.model.(foot{:}).(q_name{j}).X = ...
472                 [jointVarMean_X + jointVarStd_X, ...
473                 jointVarMean_X, ...
474                 jointVarMean_X - jointVarStd_X];
475
476             rtn.stance.model.(foot{:}).(q_name{j}).Y = ...
477                 [jointVarMean_Y + jointVarStd_Y, ...
478                 jointVarMean_Y, ...
479                 jointVarMean_Y - jointVarStd_Y];
480
481             rtn.stance.model.(foot{:}).(q_name{j}).Z = ...
482                 [jointVarMean_Z + jointVarStd_Z, ...
483                 jointVarMean_Z, ...
484                 jointVarMean_Z - jointVarStd_Z];
485         end
486     end
487
488     % Mean, std
489     for j=1:numel(q_grf)
490         varMean = ...
491             mean(rtn.stance.grf.normalized.(foot{:}).(['L',q_grf{j}]),
                    2);
492
493         varStd = ...
494             std(rtn.stance.grf.normalized.(foot{:}).(['L',q_grf{j}])')';
495
496         rtn.stance.grf.(foot{:}).(['L',q_grf{j}]) = [varMean + varStd,
                ...
```

```
497             varMean, ...
498             varMean - varStd];
499
500         varMean = ...
501             mean(rtn.stance.grf.normalized.(foot{:}).(['R',q_grf{j}]),
                    2);
502
503         varStd = ...
504             std(rtn.stance.grf.normalized.(foot{:}).(['R',q_grf{j}])')';
505
506         rtn.stance.grf.(foot{:}).(['R',q_grf{j}]) = [varMean + varStd,
                ...
507             varMean, ...
508             varMean - varStd];
509     end
510
511
512     if (processEmb)
513         for j=2:numel(q_control)
514             varMean = mean(rtn.stance.emb.normalized.(foot{:}).(
                    q_control{j}), 2);
515             varStd = std(rtn.stance.emb.normalized.(foot{:}).(q_control{
                    j})')';
516
517             rtn.stance.emb.(foot{:}).(q_control{j}) = [varMean + varStd,
                     ...
518                 varMean, ...
519                 varMean - varStd];
520         end
521     end
522
523 end
524 %%
525 if (processVicon)
526     figure;
527     subplot(311); hold all;
528     plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.r.RAnkleAngles.X,'
            r', ...
529          'lineWidth',0.1);
530     plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.l.LAnkleAngles.X,'
            k', ...
531          'lineWidth',0.1);
532     grid on
533     subplot(312); hold all;
534     plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.r.RAnkleMoment.X,'
            r', ...
535          'lineWidth',0.1);
536     plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.l.LAnkleMoment.X,'
            k', ...
537          'lineWidth',0.1);
```

```
538     grid on
539     subplot(313); hold all;
540     plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.r.RAnklePower.X,'r
            ', ...
541         'lineWidth',0.1);
542     plot(rtn.gait.gaitCycle,rtn.gait.model.normalized.l.LAnklePower.X,'k
            ', ...
543         'lineWidth',0.1);
544     grid on
545 end
546 end
```

B.2 Symmetry Learning Controller Scripts

This section includes a MATLAB implementation of the iterative learning and phase variable calculator algorithms, which are detailed in Sections 3.3 and 3.4 respectively.

B.2.1 Iterative Learning Code

```
function S_k = cp_ilc(yd_k, y_k, S_km1, varargin)
% adaptiveILC Iterative learning control with adaptive learning gains.
%
% name-value pairs:
%
% 'model' options:
%          'off'
%          'interp' use G_inv (not working currently)
%          'fly' use U_k / Y_k
%          'average' average model
%          'error-inversion' invert with error signal (not currently
    working)
%
% 'adapt' options:
%          'off' no adapting
%          'gain' adapt learning gain only
%          'gain-phase' adapt learning gain and phase
%          'data-based' invert and update gain using error signal
%
% 'backstepping' options:
%          'off'
%          'on' use 'best-so-far' in update law
%
%   Use:
%
%   First iteration:
%   S_0 = ILC(yd_k, y_k, gain_k, 0, 'init', maxHarm)
%
%
%   All others:
%   S_k = ILC(yd_k, y_k, gain_k, S_km1)
%
%                          yd_k    -  k_th desired output
%                          y_k     -  k_th output
%                          epsilon -  error buffer (use variance of
    output k=0)
%                          gain_k  -  learning weight
%                          S_km1   -  {k-1}_th stucture
%
```

```
38 %                          S_k : Yd_k      - k_th desired output
39 %                                yd_k      - k_th desired output (time
       domain)
40 %                                Y_k       - k_th ouput
41 %                                y_k       - k_th ouput (time domain)
42 %                                rho_k     - k_th learning gains
43 %                                E_bar_k   - k_th update error
44 %                                U_bar_k   - k_th update input
45 %                                E_k       - k_th error
46 %                                U_kp1     - (k+1)_th input
47 %                                u_kp1     - (k+1)_th input (time domain)
48 %                                e_k_1     - k_th norm(yd_k - y_k,1)
49 %                                e_k_2     - k_th norm(yd_k - y_k,2)
50 %
51 % Example:
52 %   S_0 = ILC(yd_0, y_0, gain_k, 0, 'init', maxHarm);
53 %   S_1 = ILC(yd_1, y_1, gain_k, S_0);
54 %   S_2 = ILC(yd_2, y_2, gain_k, S_1);
55 %   .
56 %   .
57 %   .
58 %   S_k = ILC(yd_k, y_k, gain_k, S_km1);
59 %
60
61 % Inputs
62 L = numel(yd_k);
63
64 f = (0:(L/2))/L;
65 f = f(:);
66
67 Yd_k = fft(yd_k);
68 Yd_k = Yd_k(1:(floor(L/2)+1));
69
70 Y_k = fft(y_k);
71 Y_k = Y_k(1:(floor(L/2)+1));
72
73 % Default Options
74 init = 0;
75 adapt = 'off';                % adaption technique
76 model = 'off';                % model technique
77 backstepping = 'off';         % backstepping
78 gain = 1;                     % gain
79 epsilon = 1e-10;                 % padding
80 maxHarm = numel(L)-1;         % maximum learning harmonic
81 alpha = 0.5;                  % alpha - rate of learing decrease
82 zeta = 1 + alpha*0.75;        % zeta - rate of learning increase
83 phi = 45;                     % phase shift
84 rho_max = Inf;                  % max rho
85 fs = 1000;
86 use_model = 0;
```

```
87  init_struct = [];
88
89  if (nargin > 3)
90      nVarArgs = length(varargin);
91      if strcmp(varargin{1},'init')
92          init = 1;
93          for j=2:2:nVarArgs
94              switch varargin{j}
95                  case 'gain'
96                      gain = varargin{j+1};
97                  case 'alpha'
98                      alpha = varargin{j+1};
99                  case 'zeta'
100                     zeta = varargin{j+1};
101                 case 'phi'
102                     phi = varargin{j+1};
103                 case 'epsilon'
104                     epsilon = varargin{j+1};
105                 case 'maxHarm'
106                     maxHarm = varargin{j+1};
107                 case 'model'
108                     model = varargin{j+1};
109                 case 'adapt'
110                     adapt = varargin{j+1};
111                 case 'backstepping'
112                     backstepping = varargin{j+1};
113                 case 'rho_max'
114                     rho_max = varargin{j+1};
115                 case 'G'
116                     temp = varargin{j+1};
117                     G = temp(:,1);
118                     f_G = temp(:,2);
119                 case 'fs'
120                     fs = varargin{j+1};
121                 case 'use_model'
122                     use_model = varargin{j+1};
123             end
124         end
125         fprintf('Adaptive ILC parameters:\n');
126         fprintf('\t\tmax Harmonic = %i\n',maxHarm);
127         fprintf('\t\tgain = %i\n',gain);
128         fprintf('\t\talpha = %i\n',alpha);
129         fprintf('\t\tzeta = %i\n',zeta);
130         fprintf('\t\trho_max = %i\n',rho_max);
131         fprintf('\t\tmodel = %s\n',model);
132         fprintf('\t\tadapt = %s\n',adapt);
133         fprintf('\t\tbackstepping = %s\n',backstepping);
134
135         % Initialize rho
136         rho_k = gain.*ones(numel(f),1);
```

```
137         rho_k((maxHarm+2):end) = 0;
138         rho_ang_bar_k  = 0.*f;
139 
140         % Error
141         E_k = Yd_k - Y_k;
142         e_k_inf = norm(yd_k - y_k,inf);
143         e_k_2 = norm(yd_k - y_k,2);
144         E_k_inf = norm(Yd_k - Y_k,inf);
145         E_k_2 = norm(Yd_k - Y_k,2);
146 
147         if strcmp(model,'interp')
148             % interpolate model
149             G = interp1(f_G.*fs,abs(G),f.*fs,'pchip') ...
150                 .* exp(1j.*interp1(f_G.*fs,angle(G),f.*fs,'pchip'));
151             G_inv_k = 1 ./ G;
152         else
153             G_inv_k = 1;
154         end
155 
156         % First control signal
157         U_kp1 = rho_k .* G_inv_k .* E_k;
158         u_kp1 = real(ifft([U_kp1; conj(flipud(U_kp1(2:end-1)))]));
159 
160         S_k.params.use_model = use_model;
161         S_k.params.model = model;
162         S_k.params.adapt = adapt;
163         S_k.params.backstepping = backstepping;
164         S_k.params.fs = fs;
165         S_k.params.f = f;
166         S_k.params.maxHarm = maxHarm;
167         S_k.params.epsilon = epsilon;
168         S_k.params.gain = gain;
169         S_k.params.rho_max = rho_max;
170         S_k.params.zeta = zeta;
171         S_k.params.alpha = alpha;
172         S_k.G_inv_model = 1;
173         S_k.G_inv_k = G_inv_k;
174         S_k.G_inv_km1 = G_inv_k;
175         S_k.k = 0;
176         S_k.Yd_k = Yd_k;
177         S_k.yd_k = yd_k;
178         S_k.Y_k = Y_k;
179         S_k.y_k = y_k;
180         S_k.rho_k = rho_k;
181         S_k.E_bar_k = E_k;
182         S_k.E_hat_k = E_k;
183         S_k.U_bar_k = zeros(numel(f),1);
184         S_k.E_k = E_k;
185         S_k.U_kp1 = U_kp1;
186         S_k.u_kp1 = u_kp1;
```

```
187         S_k.e_k_inf = e_k_inf;
188         S_k.e_k_2 = e_k_2;
189         S_k.E_k_inf = E_k_inf;
190         S_k.E_k_2 = E_k_2;
191         return
192     end
193 else
194     epsilon = S_km1.params.epsilon;
195     gain = S_km1.params.gain;
196     rho_max = S_km1.params.rho_max;
197     zeta = S_km1.params.zeta;
198     alpha = S_km1.params.alpha;
199     use_model = S_km1.params.use_model;
200 end
201
202 E_k = Yd_k - Y_k;
203 e_k_inf = norm(yd_k - y_k,inf);
204 e_k_2 = norm(yd_k - y_k,2);
205 E_k_inf = norm(Yd_k - Y_k,inf);
206 E_k_2 = norm(Yd_k - Y_k,2);
207 rho_km1 = S_km1.rho_k;
208 U_k = S_km1.U_kp1;
209 E_bar_km1 = S_km1.E_bar_k;
210 U_bar_km1 = S_km1.U_bar_k;
211 E_km1 = S_km1.E_k;
212 E_hat_km1 = S_km1.E_hat_k;
213
214 % Freq where mag of error has increased error + pad
215 E_incr = abs(E_k) > (abs(E_bar_km1) + epsilon);
216
217 % Freq the magnitude of error has decreased error - pad
218 E_decr = abs(E_k) < (abs(E_bar_km1) - epsilon);
219
220 E_inside_pad = ~(E_decr) & ~(E_incr);
221
222 % Deadzone
223 DZ = abs(E_k) < ((1/100).*abs(Yd_k));
224
225 % Backstepping
226 if strcmp(S_km1.params.backstepping,'on')
227     E_bar_k = (E_no_decr .* E_bar_km1) + (E_decr.* E_k);
228     U_bar_k = (E_no_decr.* U_bar_km1) + (E_decr .* U_k);
229 else
230     E_bar_k = E_k;
231     U_bar_k = U_k;
232 end
233
234 % Init vars
235 E_hat_k = (E_bar_km1 - E_k)./E_bar_km1;
236
```

```
237 % Model
238 if strcmp(S_km1.params.model,'fly')
239     G_inv_k = U_k ./ Y_k;
240     G_inv_k(isnan(G_inv_k)) = 0;
241     G_inv_k(isinf(G_inv_k)) = 0;
242     G_inv_model = U_k ./ Y_k;
243 
244 elseif strcmp(S_km1.params.model,'error-inversion')
245     G_inv_k = S_km1.G_inv_km1.*(E_bar_k .* rho_km1) ./ (E_bar_km1 - E_k)
        ;
246     G_inv_k(isnan(G_inv_k)) = 0;
247     G_inv_k(isinf(G_inv_k)) = 0;
248     G_inv_model = U_k ./ Y_k;
249 
250 elseif strcmp(S_km1.params.model,'average')
251     G_inv_k = U_k ./ Y_k;
252     G_inv_k(isnan(G_inv_k)) = 0;
253     G_inv_k(isinf(G_inv_k)) = 0;
254     avg_magG = abs(S_km1.G_inv_k);
255     avg_angG = angle(S_km1.G_inv_k);
256     n = (S_km1.k + 1);
257     if n > 1
258         new_magG = (abs(G_inv_k) + (n-1).*avg_magG) ./ n;
259         new_angG = (angle(G_inv_k) + (n-1).*avg_angG) ./ n;
260         % G_inv_k = new_angG .* exp(1j.*new_angG);
261         G_inv_k = new_magG .* exp(1j.*new_angG);
262     end
263     G_inv_model = U_k ./ Y_k;
264 else
265     G_inv_k = S_km1.G_inv_k;
266     G_inv_model = U_k ./ Y_k;
267 end
268 
269 % Updates
270 if strcmp(S_km1.params.adapt,'gain')
271 
272     rho_k = (~DZ).*((E_incr .* rho_km1*alpha)...
273                   + (E_decr .* rho_km1*zeta) ...
274                   + (E_inside_pad .* rho_km1)) ...
275         + (DZ).*(rho_km1);
276     rho_k(abs(rho_k) > rho_max) = rho_max ...
277         *exp(1j.*angle(rho_k(abs(rho_k) > rho_max)));
278 
279 elseif strcmp(S_km1.params.adapt,'gain-phase')
280     phasor = exp(1j*-deg2rad(phi));
281     rho_k = E_no_decr .* rho_km1*alpha.*phasor...
282         +   E_no_decr.* rho_km1*alpha...
283         +   E_decr.* rho_km1*zeta;
284     rho_k(abs(rho_k) > rho_max) = rho_max ...
285         *exp(1j.*angle(rho_k(abs(rho_k) > rho_max)));
```

```
286
287 elseif strcmp(S_km1.params.adapt,'data-based')
288     E_hat_k = (E_bar_km1 - E_k) ./ E_bar_km1;
289     rho_k = (1./abs(E_hat_k)).*abs(rho_km1) ...
290         .* exp(1j.*(angle(rho_km1) - angle(E_hat_k)));
291     rho_k(abs(rho_k) > rho_max) = rho_max ...
292         *exp(1j.*angle(rho_k(abs(rho_k) > rho_max)));
293 else
294     rho_k = rho_km1;
295 end
296
297 % Update Law
298 if use_model
299     U_kp1 = U_bar_k + (~DZ).*(rho_k .* G_inv_k .* E_bar_k);
300 else
301     U_kp1 = U_bar_k + (~DZ).*(rho_k .* 1 .* E_bar_k);
302 end
303 u_kp1 = real(ifft([U_kp1; conj(flipud(U_kp1(2:end-1)))]));
304
305 % Copy previous state
306 S_k = S_km1;
307 S_k.k = S_k.k + 1;
308 S_k.E_hat_k = E_hat_k;
309
310 % Update Struct Members
311 S_k.rho_k = rho_k;
312
313
314 S_k.Yd_k = Yd_k;
315 S_k.yd_k = yd_k;
316 S_k.Y_k = Y_k;
317 S_k.y_k = y_k;
318 S_k.E_bar_k = E_bar_k;
319 S_k.U_bar_k = U_bar_k;
320 S_k.E_k = E_k;
321 S_k.G_inv_k = G_inv_k;
322 S_k.G_inv_km1 = G_inv_k;
323 S_k.G_inv_model = G_inv_model;
324 S_k.U_kp1 = U_kp1;
325 S_k.u_kp1 = u_kp1;
326 S_k.e_k_inf = e_k_inf;
327 S_k.e_k_2 = e_k_2;
328 S_k.E_k_inf = E_k_inf;
329 S_k.E_k_2 = E_k_2;
330 end
```

B.2.2 Phase Variable Calculator Code

```
function [Phi, Xs, Ys, k, ix0] = cp_PHI(x,t)
% PHI - is a function that computes the biomechanical phase variable PHI
% from the integral of the global thigh angle and the global thigh angle
%
% INPUTS:
% x - Real (1 x n) -    is a measurement of the global thigh angle
    during a gait cycle
% t - Real (1 x n) -    is the time vector of the gait cycle (or number
    of samples)
%
% OUTPUTS:
% Phi - Real (1 x n) -  is the normalized phase variable computed from
    the polar angle
%                       of the thigh angle-integral pgase portrait
%%

% Computing the integral of the hip joint
ix0 = trapz(x)/length(t); % shift in the hip angle for symmetry w.r.t
    the Y axis.
X = -(x - ix0);

% Y = cumtrapz(X)/length(t);
Y = cumtrapz(X);

% Parameters to increase linearity in the phase variable
k = abs(max(X)-min(X))/abs(max(Y)-min(Y)); % Scale factor
Xs = X;
Ys = k*Y;

% Phase Variable Phi
phi = atan2(Ys,Xs);
% phi = unwrap(phi);       % Shift phase angles

% Normalized Phase Variable
% Phi = (phi - phi(1))/(phi(end) - phi(1));
tol = -1e-2;
if all(diff(phi(3:end-2))>= tol)
    Phi = (phi + pi)/(2*pi); % works for stritly increasing phi
else
    % Phi = (phi + pi)/(2*pi); % works for stritly increasing phi
    Phi = (mod(phi,2*pi) - 0)/(2*pi); % works for phi with
        discontinuities
end

disCont_ix = find(diff(Phi) < -0.5);
for ii = 1:length(disCont_ix)
    ix = disCont_ix(ii);
   if  abs(ix-1) < abs(ix - length(t))
```

```
44          Phi(ix) = 0;
45      elseif abs(ix - length(t)) < abs(ix-1)
46          Phi(ix+1) = 1;
47      end
48  end
49
50  % X = x;
51  % Y = cumtrapz(X)/length(t);
52  %
53  % X_max = max(X);
54  % X_min = min(X);
55  % Y_max = max(Y);
56  % Y_min = min(Y);
57  %
58  % z = abs(X_max - X_min)/abs(Y_max - Y_min);
59  % gamma = -(X_max + X_min)/2;
60  % Gamma = -(Y_max + Y_min)/2;
61  %
62  % Xs = X + gamma;
63  % Ys = z*(Y + Gamma);
64  %
65  % phi = atan2(Ys,Xs);
66  % % phi = unwrap(phi) - 2*pi;
67  %
68  % % Phi = (phi + pi)/(2*pi);
69  % Phi = (phi - phi(1))/(phi(end) - phi(1));
70
71  end
```

B.3 Learning User Interface Implementation Code

This code processes the Vicon inverse dynamics, ground reaction force, and embedded LabVIEW data files, and stores them as MATLAB structures. Iterative learning is then conducted and a graphical user interface is updated to monitor the signals. Lookup tables are then generated. All commands from the user are processed through a text-based user interface within MATLAB's command line. This program is used during experiments and provides the experimentalist with details of the state of the learning algorithm, data logging, and control algorithms.

```
function rtn = cp_poweredAnkle_tui(varargin)
%%
% cp_poweredAnkle_tui  Text based user interface for iterative learning control.
%
%   cp_poweredAnkle_tui() default configuration.
%
%   cp_poweredAnkle_tui(Name,Value, ...) Name, Value pairs for adjustable settings.
%

% Count number of input arguments
nVarArgs = length(varargin);


% Defauts Settings
processVicon = 1;

% algorithm setting
gain = 0.5; % 1
maxharmonic = 10; % 10
smooth_vector = [80,5]; % [90,5]
zeta = 1; % 1.5
alpha = 0.5;
adapt = 'gain';
model = 'average'; % average
backstepping = 'off';
eps_scale = 3; % higher = more adaptations, lower = less, have it at least 1
end_learning = 0.70; % starting point of when to set learning to 0
use_model = 0; % boolean to trigger the use of the model on or off

% Motor 3 settings
k_t = 10.2/1000;
k_s = 940;
```

```
33 Vcc = 60;
34 R_T = 1800;
35 maxMotorAmp = 14;
36 maxMotorRpm = 80000;
37 file_suffix = '';
38
39 % Zero-phase filter settings
40 gaitCycle = 0.1:0.1:100;
41 N = numel(gaitCycle);
42 fc_norm = (2*10)/N;
43
44 % Collect Arguments
45 % Override default settings when applicable
46 for i=1:2:nVarArgs
47     switch varargin{i}
48         case 'gain'
49             gain = varargin{i+1};
50         case 'maxharmonic'
51             maxharmonic = varargin{i+1}
52         case 'fc_norm'
53             fc_norm = varargin{i+1}
54         case 'smooth_vector'
55             smooth_vector = varargin{i+1};
56         case 'k_t'
57             % torque constant (Nm/A)
58             k_tau = varargin{i+1};
59         case 'k_s'
60             % speed constant (rpm/V)
61             k_s = varargin{i+1};
62         case 'Vcc'
63             % operating voltage (V)
64             Vcc = varargin{i+1};
65         case 'R_T'
66             % total tranmission ratio
67             R_T = varargin{i+1};
68         case 'maxMotorAmp'
69             % max motor current (A) -- setting on ESCON 70/10 servo
                controller
70             maxMotorAmp = varargin{i+1};
71         case 'maxMotorRpm'
72             % max motor speed (Rpm) -- setting on ESCON 70/10 servo
                controller
73             maxMotorRpm = varargin{i+1};
74         case 'epsilon'
75             eps_scale = varargin{i+1};
76         otherwise
77             fprintf('\n%s option not found!\n',varargin{i})
78             rtn = [];
79             return
80     end
```

```
81  end
82
83  %    ------------------------------------------------------------------------
84  % Initialize Plots
85  %    ------------------------------------------------------------------------
86  fig = figure;
87
88  % |E_k| v freq.
89  subplot(3,3,4); hold all;
90  h_E_bar_v_f_p_gamma = stem(0:maxharmonic, 0.*(0:maxharmonic),'--r');
91  h_E_bar_v_f = stem(0:maxharmonic, 0.*(0:maxharmonic),'k');
92  h_E_v_f = stem((0:maxharmonic)+0.2, 0.*(0:maxharmonic),'m');
93  ylabel('$| E |$','fontsize',17,'interpreter','latex');
94  xlabel('Harmonic','fontsize',17,'interpreter','latex');
95  grid on; box on;
96  xlim([-0.3,maxharmonic+0.5])
97  l = legend([h_E_bar_v_f,h_E_bar_v_f_p_gamma,h_E_v_f], ...
98      {'$E^*_k$','$E^*_k + \gamma$', '$E_k$'});
99  set(l,'interpreter','latex', ...
100     'Position', [0.0446 0.57 0.0571  0.0726],...
101     'FontSize',10);
102
103
104 % |U_k| v freq.
105 subplot(3,3,7), hold all;
106 h_U_bar_v_f = stem(0:maxharmonic, 0.*(0:maxharmonic),'k');
107 h_U_v_f = stem((0:maxharmonic)+0.2, 0.*(0:maxharmonic),'m');
108 h_U_kp1_v_f = stem((0:maxharmonic)-0.2, 0.*(0:maxharmonic),'b');
109 ylabel('$| U |$','fontsize',17,'interpreter','latex');
110 xlabel('Harmonic','fontsize',17,'interpreter','latex');
111 grid on; box on;
112 xlim([-0.3,maxharmonic+0.5])
113 l = legend([h_U_bar_v_f,h_U_v_f,h_U_kp1_v_f], ...
114     {'$U^*_k$','$U_k$','$U_{k+1}$'});
115 set(l,'interpreter','latex', ...
116     'Position',[0.0581    0.2779    0.0400    0.0500],...
117     'FontSize',10);
118
119
120
121 % rho v freq
122 subplot(3,3,1); hold all;
123 h_rho_v_f = stem(0:maxharmonic, 0.*(0:maxharmonic),'k');
124 ylabel('$\rho$','fontsize',17,'interpreter','latex');
125 xlabel('Harmonic','fontsize',17,'interpreter','latex');
126 grid on; box on;
```

```
127 xlim([-0.3,maxharmonic+0.5])
128
129 % y v t
130 subplot(3,3,2:3); hold all;
131 h_title = title(['Iteration = ', num2str(0)], ...
132     'interpreter','latex','fontsize',17);
133 h_yd_v_t = plot(gaitCycle,0.*gaitCycle,'k');
134 h_y_v_t = plot(gaitCycle,0.*gaitCycle,'r');
135 h_u_k_v_t = plot(gaitCycle,0.*gaitCycle,'g');
136 h_u_kp1_v_t = plot(gaitCycle,0.*gaitCycle,'c');
137 h_e_kp1_v_t = plot(gaitCycle,0.*gaitCycle,'m');
138 h_stance_1_v_t = plot([0 0],[0,0],'--k');
139 h_pre_1_v_t = plot([0 0],[0,0],'--k');
140 ylabel('(rad)','fontsize',17,'interpreter','latex');
141 grid on; box on
142 l = legend([h_y_v_t,h_yd_v_t,h_u_k_v_t,h_u_kp1_v_t,h_e_kp1_v_t], ...
143     {'$\tilde{\theta}_{p,k}$','$\tilde{\theta}_{b,k}$', ...
144     '$\tilde{\theta}_{v,k}$', ...
145     '$\tilde{\theta}_{v,k+1}$', ...
146     '$\theta_{v,k+1}$'});
147 set(l,'interpreter','latex', ...
148     'Position', [0.9122,0.7228,0.0742,0.22],...
149     'FontSize',14);
150
151
152 % u v t
153 subplot(3,3,5:6); hold all;
154 title('$U_{k+1} = U^*_k + \rho E^*_k$','interpreter','latex')
155 h_e_v_t = plot(gaitCycle,0.*gaitCycle,'b');
156 h_u_v_t = plot(gaitCycle,0.*gaitCycle,'g');
157 h_u_filt_v_t = plot(gaitCycle,0.*gaitCycle,'m');
158 h_stance_2_v_t = plot([0 0],[0,0],'--k');
159 h_pre_2_v_t = plot([0 0],[0,0],'--k');
160 xlabel('\% gait','fontsize',17,'interpreter','latex');
161 ylabel('(rad)','fontsize',17,'interpreter','latex');
162 grid on; box on;
163
164 l = legend([h_e_v_t,h_u_v_t,h_u_filt_v_t], ...
165     {'$\theta_{e,k}$','$\theta_{v,k+1}$','$\hat{\theta}_{v,k+1}$'});
166 set(l,'interpreter','latex', ...
167     'Position',[0.9139,0.4661,0.0742,0.1233],...
168     'FontSize',14);
169
170
171 % e v t | f
172 subplot(3,3,8:9); hold all
173 h_er_inf_t = plot(0,0,'k');
174 h_er_2_t = plot(0,0,'r');
175 h_er_inf_f = plot(0,0,'--b');
176 h_er_2_f = plot(0,0,'--g');
```

```
177 h_er_inf_t_min = plot(0,0,'ok');
178 h_er_2_t_min = plot(0,0,'or');
179 h_er_inf_f_min = plot(0,0,'ob');
180 h_er_2_f_min = plot(0,0,'og');
181 xlabel('Iteration','fontsize',17,'interpreter','latex');
182 ylabel('Error','fontsize',17,'interpreter','latex')
183 grid on; box on;
184 l = legend('$\|e\|_\infty$','$\|e\|_2$','$\|E\|_\infty$','$\|E\|_2$');
185 set(l,'interpreter','latex','box','off',...
186     'FontSize',14);
187
188
189 mon = get(0, 'MonitorPositions');
190 [~,mon1] = min(mon(:,3));
191 set(fig,'Position', [mon(mon1,1)+100, mon(mon1,2)+100, 1000, 500])
192 % fig.Position = [3 61 1426 738];
193 fig.PaperPosition = [-5.3542,1.3542,19.2083,8.2917];
194
195 % Filter for uff smoothing
196 [b,a] = butter(1,fc_norm);
197
198 % Collect settings
199 settings.maxharmonic = maxharmonic;
200 settings.gain = gain;
201 settings.fc_norm = fc_norm;
202 rtn.settings = settings;
203
204 fprintf('\n----------------------\n')
205 fprintf('Iterative Learning UI\n')
206 fprintf('----------------------\n')
207 fprintf('\nSettings:\n')
208
209 fprintf('\tMaximum harmonic = %i\n',maxharmonic)
210 fprintf('\tLearning Gain = %.3f\n',gain)
211 fprintf('\tNormalized Cutoff Freq = %.3f\n',fc_norm)
212
213 % Error Vectors
214 Einf_t = [];
215 E2_t = [];
216 Einf_f = [];
217 E2_f = [];
218
219 % ILC Settings
220 maxHarmonic = maxharmonic;
221 initial_learning_gain = gain;
222
223 % Get directory
224 path = 'C:\Users\cpras\Documents\UW\Thesis\Testing';
225 selpath = uigetdir(path);
226 % selpath = uigetdir(pwd);
```

```
227
228 % Learning Loop
229 k=1;
230 while(1)
231     fprintf('\n..............\n')
232     fprintf('Iteration %i\n',k - 1);
233     fprintf('..............\n')
234
235     % Get k trial
236     while(1)
237         fprintf('\n\t');
238         trialName = input('Enter trial name: ','s');
239         fprintf(['\n\t',trialName]);
240         usr_input = input(' - Is this correct? [y/n] ','s');
241         if strcmp('y',usr_input)
242             break;
243         end
244     end
245
246     % Data Collection
247     trial  = fullfile(selpath, trialName);
248     fprintf('\n\tParsing data...\n');
249     rtn.T{k} = cp_process_trial(trial, ...
250         'k_t',k_t, ...
251         'k_s',k_s, ...
252         'Vcc',Vcc, ...
253         'R_T',R_T, ...
254         'maxMotorAmp',maxMotorAmp, ...
255         'maxMotorRpm',...
256         'processVicon',processVicon);
257
258     % Get info on k=0 iteration
259     if (k==1)
260
261         % Which leg is prosthetic on?
262         prostheticSide = rtn.T{k}.params.emb.Is_Prosthetic_Right;
263         if prostheticSide
264             prostheticSide = 'right';
265         else
266             prostheticSide = 'left';
267         end
268         % mass = rtn.T{k}.params.emb.Total_Mass; % CHANGE
269         % rtn.settings.mass = mass; % CHANGE
270         rtn.settings.prostheticSide = prostheticSide;
271         fs = rtn.T{k}.params.model.fs;
272
273         fprintf('\tProsthetic Side = %s\n',prostheticSide);
274         % fprintf('\tMass = %i\n',mass)
275     end
276
```

```
277     time = rtn.T{k}.time;
278     grf.time = rtn.T{k}.grf.time;
279     emb.time = rtn.T{k}.emb.time;
280 
281     % Desired output (biological) and real output (prosthetic)
282     if strcmp(prostheticSide,'left')
283 
284         % Mean Vicon Kinematics
285         yd_k = rtn.T{k}.gait.model.r.RAnkleAngles.X(:,2);
286         y_k = rtn.T{k}.gait.model.l.LAnkleAngles.X(:,2);
287 
288         % all normalized trajectories
289         yd_k_all = rtn.T{k}.gait.model.normalized.r.RAnkleAngles.X;
290         y_k_all = rtn.T{k}.gait.model.normalized.l.LAnkleAngles.X;
291 
292         % Timing
293         Tp = rtn.T{k}.segment.grf.l.gait.time(:,2) ...
294             - rtn.T{k}.segment.grf.l.gait.time(:,1);
295 
296         stance_percent = mean((rtn.T{k}.segment.grf.l.stance.time(:,2)
                ...
297             - rtn.T{k}.segment.grf.l.stance.time(:,1))./Tp).*100;
298 
299         % HS events
300         grf.hs = rtn.T{k}.ge.grf.l.hs_time;
301         emb.hs = rtn.T{k}.ge.emb.l.hs_time;
302 
303         % HS event indices
304         grf.hs_ind = rtn.T{k}.ge.grf.l.hs_ind;
305         emb.hs_ind = rtn.T{k}.ge.emb.l.hs_ind;
306 
307         % Torques
308         tau_p = rtn.T{k}.gait.model.l.LAnkleMoment.X(:,2);
309         tau_b = rtn.T{k}.gait.model.r.RAnkleMoment.X(:,2);
310 
311         % Encoder
312         theta_e = rtn.T{k}.gait.emb.l.ankPos(2:end,2);
313 
314     else
315 
316         % mean torques
317         yd_k = rtn.T{k}.gait.model.l.LAnkleAngles.X(:,2);
318         y_k = rtn.T{k}.gait.model.r.RAnkleAngles.X(:,2);
319 
320         % all normalized trajectories
321         yd_k_all = rtn.T{k}.gait.model.normalized.l.LAnkleAngles.X;
322         y_k_all = rtn.T{k}.gait.model.normalized.r.RAnkleAngles.X;
323 
324 
325         % Timing
```

```
326         Tp = rtn.T{k}.segment.grf.r.gait.time(:,2) ...
327             - rtn.T{k}.segment.grf.r.gait.time(:,1);
328 
329         stance_percent = mean((rtn.T{k}.segment.grf.r.stance.time(:,2)
                ...
330             - rtn.T{k}.segment.grf.r.stance.time(:,1))./Tp).*100;
331 
332         % HS events
333         grf.hs = rtn.T{k}.ge.grf.r.hs_time;
334         emb.hs = rtn.T{k}.ge.emb.r.hs_time;
335 
336         % HS event indices
337         grf.hs_ind = rtn.T{k}.ge.grf.r.hs_ind;
338         emb.hs_ind = rtn.T{k}.ge.emb.r.hs_ind;
339 
340         % Torques
341         tau_p = rtn.T{k}.gait.model.r.RAnkleMoment.X(:,2);
342         tau_b = rtn.T{k}.gait.model.l.LAnkleMoment.X(:,2);
343 
344         % Encoder
345         theta_e = rtn.T{k}.gait.emb.r.ankPos(2:end,2);
346 
347     end
348 
349     % Remove first point (0% == 100%) since we want N even for FFT
350     yd_k = yd_k(2:end);
351     y_k = y_k(2:end);
352 
353     % Default late smooth
354     % smooth_vector(1) = (stance_percent + 20);
355 
356     % Train MoCap to Encoder Mapper
357     % -- Collect training data for all trials collected up to this point
358     pred = [];
359     resp = [];
360     if strcmp(prostheticSide,'left')
361         for ii = 1:k
362             pred = [pred; rtn.T{ii}.model.LAnkleAngles.X];
363             resp = [resp; interp1(rtn.T{ii}.emb.time,...
364                 rtn.T{ii}.emb.Ankle_Encoder_Angle, rtn.T{ii}.time)]; %
                    CHANGE
365             % resp = [resp; (1.2*rtn.T{ii}.model.LAnkleAngles.X)-0.05];
366         end
367     else
368         for ii = 1:k
369             pred = [pred; rtn.T{ii}.model.RAnkleAngles.X];
370             resp = [resp; interp1(rtn.T{ii}.emb.time,...
371                 rtn.T{ii}.emb.Ankle_Encoder_Angle, rtn.T{ii}.time)]; %
                    CHANGE
372             % resp = [resp; (1.2*rtn.T{ii}.model.RAnkleAngles.X)-0.05];
```

```
373         end
374     end
375     mdl = fitlm(pred, resp, 'linear', 'RobustOpts', 'off');
376     p = polyfit(pred,resp,1);
377     fprintf('\n\tp = [%f , %f]\n',p)
378     if(1)
379         ypred = predict(mdl,pred);
380         % ypoly = polyval(p,pred);
381         figure; hold all;
382         %             set(groot, 'defaultAxesTickLabelInterpreter','
                latex'); set(groot, 'defaultLegendInterpreter','latex');
383         %             set(0, 'defaultTextInterpreter', 'latex');
384         set(findall(gcf,'-property','FontSize'),'FontSize',12)
385         set(findall(gcf,'-property','interpreter'),'interpreter','latex'
                )
386         plot(pred,'b')
387         plot(resp,'k')
388         plot(ypred,'r')
389         % plot(ypoly,'m')
390         % lh = legend('Vicon','Encoder','Linear Xform Map','Poly Xform
                Map','location','best');
391         lh = legend('Vicon','Encoder','Linear Xform Map','location','
                best');
392         set(lh,'Interpreter','latex','Fontsize', 12)
393         grid on
394         ylabel('Prosthetic Ankle Angle [rad]','interpreter','latex','
                Fontsize',14)
395         xlabel('Sample','interpreter','latex','Fontsize',14)
396         title(['y = ',num2str(p(1)),'x + (',num2str(p(2)),')'])
397     end
398
399     % Phase Variable
400     thigh_angle = rtn.T{k}.gait.emb.l.thigh_angle(2:end,2);
401     Phi = cp_PHI(thigh_angle,gaitCycle);
402     if(1)
403         figure()
404         plot(gaitCycle, Phi, '.')
405         xlabel('Gait'); ylabel('Phi', 'FontSize', 12);
406         title('Mean Thigh Angle-Integral Phase Variable')
407         grid on
408     end
409     rtn.S{k}.phase.Mean_Phi = Phi;
410
411     % Collected Phase Variable vs Offline Computation
412     th = rtn.T{k}.emb.Thigh_Angle;
413     t = rtn.T{k}.emb.time;
414     ind = interp1(t,1:length(t),grf.hs,'nearest'); % HS indices (grf
            events --> emb times)
415
416     % offline
```

```
417     phi = zeros(length(ind(1):ind(end)),1);
418     for i=1:numel(grf.hs)-1
419         range = ind(i):ind(i+1);
420         phi(range-ind(1)+1) = cp_PHI(th(range),t(range));
421     end
422     phi_meas = rtn.T{k}.emb.Phi(ind(1):ind(end)); % online
423 
424     if(0)
425         t_r = t(ind(1):ind(end));
426         figure(); plot(t_r,phi); hold on; plot(t_r,phi_meas)
427         xlabel('Time [s]'); ylabel('Normalized Phase Angle')
428         legend('Offline','Online')
429     end
430 
431     while(1)
432 
433         % If first iteration, initialize learning
434         if (k==1)
435             fprintf('\n\tInitializing learning structures...\n');
436 
437             % First iteration, find y_0 std in freq. domain
438             L = numel(y_k_all(2:end,1));
439             f = (0:(L/2));
440             Y_k_all = fft(y_k_all(2:end,:));
441             Y_k_all = Y_k_all(1:(L/2)+1,:);
442             Y_k_abs_mean = mean(abs(Y_k_all)')';
443             Y_k_abs_std = std(abs(Y_k_all)')';
444 
445             epsilon = eps_scale.*Y_k_abs_std;
446             %             epsilon = 0.*Y_k_abs_std;
447 
448             figure;
449             title('Standard deviation of output','fontsize',20);
450             plot(f,epsilon,'ok');
451             xlabel('Harmonic','fontsize',20);
452             ylabel('STD','fontsize',20)
453             xlim([0,maxharmonic])
454 
455             pause
456 
457             % Zero out 70-100%
458             ix = end_learning*length(yd_k):length(yd_k);
459             yd_k(ix) = 0;
460             y_k(ix) = 0;
461 
462             rtn.S{k} = cp_ilc(yd_k, y_k, 0, ...
463                 'init','gain',initial_learning_gain, ...
464                 'epsilon', epsilon,...
465                 'model',model,...
466                 'adapt',adapt,...
```

```
467                 'backstepping',backstepping,...
468                 'maxHarm',maxHarmonic, ...
469                 'rho_max',inf, ...
470                 'zeta',zeta, ...
471                 'alpha',alpha, ...
472                 'use_model',use_model, ...
473                 'fs', fs);
474 
475         else
476             fprintf('\n\tLearning...\n');
477 
478             % Zero out 70-100%
479             ix = end_learning*length(yd_k):length(yd_k);
480             yd_k(ix) = 0;
481             y_k(ix) = 0;
482 
483             rtn.S{k} = cp_ilc(yd_k,y_k,rtn.S{k-1});
484 
485         end
486 
487         if(0)
488             figure; hold all;
489             plot(time,y);
490             for i=1:numel(grf.hs)
491                 plot([grf.hs(i) grf.hs(i)], [min(y) max(y)],'-r')
492             end
493             for i=1:numel(emb.hs)
494                 plot([emb.hs(i) emb.hs(i)], [min(y) max(y)],'--r')
495             end
496         end
497 
498         % Find HS delay
499         if (numel(grf.hs) ~= numel(emb.hs))
500             fprintf('\t\t\tHS mismatch, cannot calculate delay');
501             est_delay = nan;
502         else
503             est_delay = mean(emb.hs - grf.hs);
504         end
505 
506         % smooth_vector(1) = (stance_percent + 20);
507         Tp_mean = mean(Tp);
508 
509         fprintf('\n\t\tTiming Info\n')
510         fprintf('\t\t-------------\n')
511         fprintf('\t\tEst. HS delay = %.3f (s), %.3f%%\n', ...
512             est_delay, (est_delay/Tp_mean).*100)
513         fprintf('\t\tTp = %.3f (s)\n',Tp_mean)
514         fprintf('\t\tInital Smoothing = %.2f%%\n',smooth_vector(2))
515         fprintf('\t\tStance = %.2f%%\n',smooth_vector(1))
516 
```

```
517         % Store Timing Info
518         info.Tp_mean = Tp_mean;
519         info.stance_percent = stance_percent;
520         info.smooth_vector = smooth_vector;
521 
522         % Store the errors
523         Einf_t(k) = rtn.S{k}.e_k_inf;
524         E2_t(k) = rtn.S{k}.e_k_2;
525         Einf_f(k) = rtn.S{k}.E_k_inf;
526         E2_f(k) = rtn.S{k}.E_k_2;
527 
528         [~,iinf_t] = min(Einf_t(1:k)./Einf_t(1));
529         [~,i2_t] = min(E2_t(1:k)./E2_t(1));
530         [~,iinf_f] = min(Einf_f(1:k)./Einf_f(1));
531         [~,i2_f] = min(E2_f(1:k)./E2_f(1));
532 
533         theta_v = rtn.S{k}.u_kp1;
534         thetaV_mapped = predict(mdl, theta_v);
535         rtn.S{k}.u_kp1_e = thetaV_mapped;
536 
537         temp = fft(rtn.S{k}.u_kp1_e);
538         rtn.S{k}.U_kp1_e = temp(1:(1000/2+1));
539 
540         if strcmp(prostheticSide,'left')
541             rtn.S{k}.theta_k_e = rtn.T{k}.gait.emb.l.ankPos(:,2);
542         else
543             rtn.S{k}.theta_k_e = rtn.T{k}.gait.emb.r.ankPos(:,2);
544         end
545 
546         rtn.S{k}.phase.Online_Phi = phi_meas;
547         rtn.S{k}.phase.Offline_Phi = phi;
548 
549         % Filtering
550         preFiltU_kp1 = rtn.S{k}.U_kp1;
551         preFiltU_kp1_e = rtn.S{k}.U_kp1_e;
552 
553         % Filter uff  [rad]
554         u_kp1_filt = rtn.S{k}.u_kp1;
555         u_kp1_filt(1:(round(smooth_vector(2)*10)+1)) = 0;
556         u_kp1_filt((round(smooth_vector(1)*10)+1):end) = 0;
557         u_kp1_filt = filtfilt(b,a,u_kp1_filt);
558         rtn.S{k}.u_kp1_filt = u_kp1_filt;
559         % Store in frequency domain
560         temp = fft(u_kp1_filt);
561         rtn.S{k}.U_kp1 = temp(1:(1000/2+1));
562 
563         % Filter mapped uff  [rad]
564         encoder = rtn.S{k}.theta_k_e;
565         u_kp1_e_filt = rtn.S{k}.u_kp1_e;
566 
```

```
567         % Change to fixed value if there's problems
568         % - Neutral angle while standing
569         neutral = [80,95];
570         gc = linspace(0,100,1001)';
571         swing_e = encoder(neutral(1)*10:neutral(2)*10);
572         mean_e = median(swing_e); % CHANGE
573         if(1)
574             n_gc = gc(neutral(1)*10:neutral(2)*10);
575             figure()
576             plot(n_gc,rad2deg(swing_e),'b.','LineWidth',1.5,'DisplayName
                ',['k = ',num2str(k-1)])
577             grid on
578             hold on
579             plot(n_gc, ones(1,length(n_gc)).*rad2deg(mean_e),'r--','
                LineWidth',1.5)
580             legend('Encoder','Mean','location','best')
581             xlabel('Gait')
582             ylabel('Angle [deg]')
583             legend('show','location','northwest')
584             title(['Mean = ',num2str(round(rad2deg(mean_e),2)),' deg'])
585         end
586
587         if (k == 1)
588             meanTheta_e = mean_e;
589         else
590             meanTheta_e = meanTheta_e;
591         end
592
593         u_kp1_e_filt(1:(round(smooth_vector(2)*10)+1)) = meanTheta_e; %
                CHANGE
594         u_kp1_e_filt((round(smooth_vector(1)*10)+1):end) = meanTheta_e;
                % CHANGE
595         u_kp1_e_filt = filtfilt(b,a,u_kp1_e_filt);
596         rtn.S{k}.u_kp1_e_filt = u_kp1_e_filt;
597         % Store in frequency domain
598         temp = fft(u_kp1_e_filt);
599         rtn.S{k}.U_kp1_e = temp(1:(1000/2+1));
600
601         if(0)
602             figure; hold all;
603             stem(abs(rtn.S{k}.U_kp1_e), '-k')
604             stem(abs(preFiltU_kp1_e), '--r')
605         end
606
607         % update plots
608         set(h_title,'string',['Iteration = ', num2str(k-1)]);
609         set(h_e_v_t,'YData',theta_e); % encoder angle (2,2) -- blue
610         set(h_u_v_t,'YData',rtn.S{k}.u_kp1_e); % virtual @ k+1 (2,2) --
                green
```

```
611         set(h_u_filt_v_t,'YData',rtn.S{k}.u_kp1_e_filt); % filtered
               virtual @ k+1 (2,2) - purple
612         set(h_stance_2_v_t,'XData',[smooth_vector(1),smooth_vector(1)],
               ...
613             'YData',[min(rtn.S{k}.u_kp1_e_filt),max(rtn.S{k}.
                   u_kp1_e_filt)]);
614         set(h_pre_2_v_t,'XData',[smooth_vector(2),smooth_vector(2)], ...
615             'YData',[min(rtn.S{k}.u_kp1_e_filt),max(rtn.S{k}.
                   u_kp1_e_filt)]);
616
617         set(h_e_kp1_v_t,'YData',rtn.S{k}.u_kp1_e_filt) % filtered
               virtual @ k+1 (2,1) -- purple
618
619         if k > 1
620             set(h_u_k_v_t, 'YData',rtn.S{k-1}.u_kp1_filt); % previous
                   filtered virtual (2,1) -- green
621             set(h_U_v_f,'YData',abs(rtn.S{k-1}.U_kp1(1:(maxharmonic+1)))
                   );
622         end
623
624         set(h_U_kp1_v_f,'YData',abs(rtn.S{k}.U_kp1(1:(maxharmonic+1))));
                % (1,3)
625
626         set(h_U_bar_v_f,'YData',abs(rtn.S{k}.U_bar_k(1:(maxharmonic+1)))
               ); % (1,3)
627         set(h_E_bar_v_f,'YData',abs(rtn.S{k}.E_bar_k(1:(maxharmonic+1)))
               ); % (1,2)
628         set(h_E_bar_v_f_p_gamma,'YData', ...
629             abs(rtn.S{k}.E_bar_k(1:(maxharmonic+1))) ...
630             + rtn.S{k}.params.epsilon(1:(maxharmonic+1)))  % (1,2)
631         set(h_y_v_t,'YData', rtn.S{k}.y_k);  % (2,1)
632         set(h_yd_v_t,'YData',rtn.S{k}.yd_k); % (2,1)
633
634         set(h_u_kp1_v_t,'YData', rtn.S{k}.u_kp1_filt); % filtered
               virtual @ k+1 (2,1) -- cyan
635         set(h_stance_1_v_t,'XData',[smooth_vector(1),smooth_vector(1)],
               ...
636             'YData',[min(rtn.S{k}.yd_k),max(rtn.S{k}.yd_k)]);
637         set(h_pre_1_v_t,'XData',[smooth_vector(2),smooth_vector(2)], ...
638             'YData',[min(rtn.S{k}.yd_k),max(rtn.S{k}.yd_k)]);
639
640         set(h_er_inf_t_min,'XData',iinf_t-1,'YData',Einf_t(iinf_t)./
               Einf_t(1)); % (2,3)
641         set(h_er_2_t_min,'XData',i2_t-1,'YData',E2_t(i2_t)./E2_t(1));  %
                (2,3)
642         set(h_er_inf_f_min,'XData',iinf_f-1,'YData',Einf_f(iinf_f)./
               Einf_f(1)); % (2,3)
643         set(h_er_2_f_min,'XData',i2_f-1,'YData',E2_f(i2_f)./E2_f(1)); %
               (2,3)
644
```

```
645         set(h_er_inf_t,'XData',(1:k)-1,'YData',Einf_t(1:k)./Einf_t(1));
646         set(h_er_2_t,'XData',(1:k)-1,'YData',E2_t(1:k)./E2_t(1));
647         set(h_er_inf_f,'XData',(1:k)-1,'YData',Einf_f(1:k)./Einf_f(1));
648         set(h_er_2_f,'XData',(1:k)-1,'YData',E2_f(1:k)./E2_f(1));
649 
650         set(h_rho_v_f,'YData',abs(rtn.S{k}.rho_k(1:(maxharmonic+1))));
651         set(h_E_v_f,'YData',abs(rtn.S{k}.E_k(1:(maxharmonic+1))));
652 
653         % Promt signal sufficient message
654         fprintf('\n\tSignal learned!');
655         %         fprintf('\t\t Is this signal sufficient?')
656         %         usr_input = input('[y/n]', 's');
657         %         fprintf('\n');
658         fprintf('\n\t');
659         fprintf('Is this signal sufficient?');
660         usr_input = input('[y/n]','s');
661         fprintf('\n');
662         if strcmp(usr_input,'n')
663             % Make adjustments to learning
664             while(1)
665                 fprintf('\t\tDo you want to change a parameter?\n')
666                 fprintf('\t\t 1 - gain\n')
667                 fprintf('\t\t 2 - earlysmooth\n')
668                 fprintf('\t\t 3 - latesmooth\n');
669                 fprintf('\t\t 4 - use model\n')
670                 fprintf('\t\t 5 - done\n')
671                 while(1)
672                     fprintf('\t\t\t');
673                     usr_input = input('Enter response: ');
674                     if isscalar(usr_input)
675                         break;
676                     end
677                 end
678 
679                 switch usr_input
680                     case 1
681                         fprintf('\t\t\t');
682                         usr_input = input('Enter new gain value: ');
683                         gain = usr_input;
684                         initial_learning_gain = gain;
685                     case 2
686                         fprintf('\t\t\t');
687                         usr_input = input('Enter new earlysmooth: ');
688                         smooth_vector(2) = usr_input;
689                     case 3
690                         fprintf('\t\t\t');
691                         usr_input = input('Enter new latesmooth: ');
692                         smooth_vector(1) = usr_input;
693                     case 4
694                         fprintf('\t\t\t')
```

```
695                         fprintf('Use inverted model?')
696                         usr_input = input('[y/n]: ','s');
697                         if strcmp(usr_input,'y')
698                             use_model = 1;
699                             rtn.S{k-1}.params.use_model = use_model;
700                         elseif strcmp(usr_input,'n')
701                             use_model = 0;
702                             rtn.S{k-1}.params.use_model = use_model;
703                         else
704                         end
705                     case 5
706                         fprintf('\tLearning Gain = %.3f\n',gain)
707                         fprintf('\tSmoothing = %i%% - %i%%\n',
                                smooth_vector)
708                         fprintf('\tUse Model = %.3f\n', use_model)
709                         break;
710                     otherwise
711                         warning('Not an option');
712                 end
713             end
714         else
715             break;
716         end
717     end
718
719     rtn.S{k}.info = info;
720
721     % End of Iteration Promt
722     fprintf('\n\t');
723     fprintf('Write feedforward signal to file?');
724     usr_input = input('[y/n]','s');
725     fprintf('\n');
726
727     % Write lookup tables
728     if strcmp(usr_input,'y')
729         writeFlag = 1;
730         file = ['./','uff_',file_suffix,num2str(k)];
731         if exist(file,'file') == 2
732             fprintf('\n\t\tFile already exist');
733             usr_input = input(' overwrite? [y/n]','s');
734             if strcmp('y',usr_input)
735                 writeFlag = 1;
736                 fid = fopen(file,'w');
737                 fprintf(fid,'%f\n',rtn.S{k}.u_kp1_e_filt);
738                 fclose(fid);
739             else
740                 writeFlag = 0;
741                 fprintf('\n\tSignal not written to file.\n');
742             end
743         end
```

```
744         if writeFlag
745             folder = 'D:\';
746             % Mean Phi
747             writematrix(reshape(Phi, 1, []),[folder,'meanPhi.csv'],'
                    Delimiter','comma')
748             % Desired Theta
749             writematrix(reshape(rtn.S{k}.u_kp1_e_filt, 1, []),[folder,'
                    thetaV.csv'],'Delimiter','comma')
750         end
751     end
752
753     fprintf('\n\t');
754     usr_input = input('Continue with next k? [y/n]','s');
755     if strcmp('n',usr_input)
756         fprintf('\t\t')
757         usr_input = input('Redo k? [y/n] (n will exit ui)','s');
758         if strcmp('y',usr_input)
759             continue;
760         end
761         break;
762     end
763
764     % Increment
765     k = k + 1;
766 end
767 end
```

Appendix C

PYTORCH SCRIPTS

This appendix contains the Python scripts used to build the human-robot regression models. Three deep-learning networks were trained and implemented within the PyTorch framework. Additionally, specified hyperparameters were optimized using the Optuna framework. The scripts are listed with brief descriptions of their functionality.

C.1 Main Program

This script is the entry point of the full hyperparameter optimization and neural network training procedure. The role of this script is to load the dataset, specify the hyperparameter search ranges, initialize neural network training, evaluate the optimized networks, and save the results.

```
"""
Created on Wed Apr 21 12:23:55 2021

Prosthetic Ankle Torque Predictor
Optimization Script for all Networks

@author: Chris Prasanna
"""

#%% Imports

import torch
import torch.nn as nn
from scipy import signal
import numpy as np

import os
import time
import sys

from DL_functions import loadmat
from Optuna_functions import optimize_hyperparams, get_dataLoaders
from Test_functions import test_network, visualize_results,
    fitted_histogram, nn_fitted_histogram
from Linear_Regression import train_poly

from GPUtil import showUtilization as gpu_usage

#%% Load and Organize Data

# Clear cuda memory
torch.cuda.empty_cache()
print("Initial GPU Usage")
gpu_usage()

# Load Data from .mat file
print('Loading Data...')
cwd = os.getcwd()
data_path = os.path.dirname(cwd)
matfile = os.path.join(data_path, 'JR_data_ankleTorque.mat')
# matfile = r'JR_data_ankleTorque.mat'
```

```
matdata = loadmat(matfile)
Data = matdata['Data']
print('Loading Complete')

# Size of Data
dims = Data['Features'].shape
num_trials = dims[0]
num_timestepsPerTrial = dims[1]
num_features = dims[2]

active_features = Data['activeFeatures']
passive_features = Data['passiveFeatures']
num_activeTrials = active_features.shape[0]
num_passiveTrials = passive_features.shape[0]

# Inputs and Outputs
features = Data['Features']
responses = Data['Responses']

#%% Hyperparameters and Optuna Options

# Set Device
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

# DNN Hyperparameters
input_size = num_features
output_size = 1
step_size = 1
fs = 120 # sampling freq
fsNew = 30 # data rate for sub-sampling
prediction_horizon = 1 # set as 1 for simulation

# Optimizer Hyperparameters
lr_step_size = 3           # number of epochs before reducing learning
    rate
lr_patience = 3
min_lr = 1e-5
amsgrad = False

# Loss Function
criterion = nn.MSELoss()

# Training Hyperparameters
num_epochs = 3
optuna_trials = 1
batch_size = num_trials
optuna_timeout = None

# Hyperparameters to Optimize Using Optuna
# - Hidden size
```

```
90  # - Number of Layers
91  # - Initial Learning Rate
92  # - STD of Gaussian Noise
93  # - Dropout Factor
94  # - Type of Optimizer (Adam, AdamW, SGD+momentum, RMSProp, etc.)
95  # - Weight decay / L2 Regularization
96  # - Learning Rate decrease factor
97  # - Sequence Length
98
99  # hidden_size = 256       # [4,8,16,32,64,128,256,512]
100 # num_layers = 2
101 # learning_rate = 0.001   # inital LR
102 # noiseSTD = 0.2
103 # dropout = 0.2
104 # weight_decay = 0.05     # L2Regularization (default = 0); tested: 1e
        -5, 0.05
105 # gamma = 0.5             # learning rate decrease factor
106 # sequence_length = 10
107
108 #%% Save Paths
109
110 timestr = time.strftime("%Y%m%d-%H%M%S")
111 directory = f'{timestr[:-2]}'
112
113 os.makedirs(os.path.join(cwd, 'Results',directory))
114
115 os.makedirs(os.path.join(cwd, 'Results',directory,'Polynomial'))
116 os.makedirs(os.path.join(cwd, 'Results',directory,'FFN'))
117 os.makedirs(os.path.join(cwd, 'Results',directory,'GRU'))
118 os.makedirs(os.path.join(cwd, 'Results',directory,'DA-RNN'))
119
120 PATH = os.path.join(cwd,'Results',directory)
121
122 #%% Resample Time Series Data
123
124 fsNew = 30 # data rate for sub-sampling
125 fsRatio = fsNew / fs
126 num_timestepsPerTrial = int(num_timestepsPerTrial * fsRatio)
127
128 for key in Data:
129      Data[key] = signal.resample(Data[key], num_timestepsPerTrial, axis
         =1)
130
131 #%% Create a Dictionary Object of Constants to pass through Functions
132
133 constants = {'device':device,
134              'results path':PATH,
135              'input size':input_size,
136              'output size':output_size,
137              'learning rate scheduler delay':lr_step_size,
```

```
138                 'learning rate scheduler patience':lr_patience,
139                 'minimum learning rate':min_lr,
140                 'number of walking trials':num_trials,
141                 'number of timesteps per trial':num_timestepsPerTrial,
142                 'loss function':criterion,
143                 'max number of epochs':num_epochs,
144                 'number of optuna trials':optuna_trials,
145                 'optuna timeout':optuna_timeout,
146                 'dataset': Data,
147                 'pred_horizon': prediction_horizon,
148                 'sub-sample freq': fsNew
149                 }
150 
151 # Independent Variables
152 #   - model name
153 #   - current optuna trial
154 #   - model
155 #   - scheduler
156 #   - optimizer
157 #   - sequence length
158 
159 #%% Polynominal Fit
160 
161 best_poly, poly_rmse, target_poly, pred_poly = train_poly(constants)
162 
163 #%% Optimize Hyperparameters for all NNs
164 
165 # remove pickle files
166 dir_name = os.getcwd()
167 test = os.listdir(dir_name)
168 for item in test:
169     if item.endswith(".pickle"):
170         os.remove(os.path.join(dir_name, item))
171 
172 print(" *** Optimizing the Networks...")
173 
174 torch.cuda.empty_cache()
175 FFN_model, FFN_trial = optimize_hyperparams('FFN', constants)
176 filename = f'FFN NN {timestr[:-2]}.pt'
177 FFN_PATH = os.path.join(PATH,'FFN',filename)
178 net = FFN_model.state_dict()
179 torch.save(net, FFN_PATH)
180 
181 torch.cuda.empty_cache()
182 GRU_model, GRU_trial = optimize_hyperparams('GRU', constants)
183 h = GRU_model.init_hidden(batch_size = 1)
184 filename = f'GRU NN {timestr[:-2]}.pt'
185 GRU_PATH = os.path.join(PATH,'GRU',filename)
186 net = GRU_model.state_dict()
187 torch.save(net, GRU_PATH)
```

```
188
189 torch.cuda.empty_cache()
190 DARNN_model, DARNN_trial = optimize_hyperparams('DA-RNN', constants)
191 filename = f'DA-RNN NN {timestr[:-2]}.pt'
192 DARNN_PATH = os.path.join(PATH,'DA-RNN',filename)
193 net = DARNN_model.state_dict()
194 torch.save(net, DARNN_PATH)
195
196 #%% Test Trained Network
197
198 print(" *** Testing the Networks...")
199
200 sequence_length_FFN = FFN_trial.params['sequence_length']
201 sequence_length_GRU = GRU_trial.params['sequence_length']
202 sequence_length_DARNN = DARNN_trial.params['sequence_length']
203
204 # FFN
205 train_loader_FFN, val_loader_FFN, test_loader_FFN = get_dataLoaders(
      FFN_trial, sequence_length_FFN,
206                                                                 const
                                                                    ,
                                                                    tr
                                                                    =
                                                                    False
                                                                    )
207 target_FFN, pred_FFN, RMSE_FFN, test_loss_FFN, pcc_FFN = test_network(
      test_loader_FFN, FFN_model,
208                                                       'FFN',
                                                          criterion,
                                                          PATH, device
                                                          ,
                                                          prediction_ho
                                                          )
209
210 # GRU
211 train_loader_GRU, val_loader_GRU, test_loader_GRU = get_dataLoaders(
      GRU_trial, sequence_length_GRU,
212                                                                 const
                                                                    ,
                                                                    tr
                                                                    =
                                                                    False
                                                                    )
213 target_GRU, pred_GRU, RMSE_GRU, test_loss_GRU, pcc_GRU = test_network(
      test_loader_GRU, GRU_model,
214                                                       'GRU',
                                                          criterion,
                                                          PATH, device
```

```
                                                                    ,
                                                                    prediction_ho
                                                                    )

# Da-RNN
train_loader_DARNN, val_loader_DARNN, test_loader_DARNN = get_dataLoaders(DARNN_trial, sequence_length_DARNN,
                                                                    const
                                                                        ,
                                                                        tr
                                                                        =
                                                                        False
                                                                        )
target_DARNN, pred_DARNN, RMSE_DARNN, test_loss_DARNN, pcc_DARNN = test_network(test_loader_DARNN, DARNN_model,
                                                                'DA-RNN',
                                                                    criterion,
                                                                    PATH, device
                                                                    ,
                                                                    prediction_ho
                                                                    )

# remove pickle files
dir_name = os.getcwd()
test = os.listdir(dir_name)
for item in test:
    if item.endswith(".pickle"):
        os.remove(os.path.join(dir_name, item))


#%% Visualize Test Data Results

print(" *** Plotting the Test Results...")

# FFN
percentFit_FFN, percentError_FFN = visualize_results(FFN_model, 'FFN', train_loader_FFN,
                                                val_loader_FFN,
                                                    test_loader_FFN,
                                                    num_trials,
                                                target_FFN, pred_FFN, PATH,
                                                    fsNew, device,
                                                    prediction_horizon)

# GRU
percentFit_GRU, percentError_GRU = visualize_results(GRU_model, 'GRU', train_loader_GRU,
```

```
242                                                 val_loader_GRU,
                                                        test_loader_GRU,
                                                        num_trials,
243                                                 target_GRU, pred_GRU, PATH,
                                                         fsNew, device,
                                                        prediction_horizon)
244
245 # DA-RNN
246 percentFit_DARNN, percentError_DARNN = visualize_results(DARNN_model, '
        DA-RNN', train_loader_DARNN,
247                                                 val_loader_DARNN,
                                                        test_loader_DARNN,
                                                        num_trials,
248                                                 target_DARNN, pred_DARNN,
                                                        PATH, fsNew, device,
                                                        prediction_horizon)
249
250 # All Models
251 fitted_histogram(target_FFN, pred_FFN,
252                  target_GRU, pred_GRU,
253                  target_DARNN, pred_DARNN,
254                  target_poly, pred_poly,
255                  PATH)
256
257 # Just NNs
258 nn_fitted_histogram(target_FFN, pred_FFN,
259                  target_GRU, pred_GRU,
260                  target_DARNN, pred_DARNN,
261                  PATH)
262
263 print("\n\n*** Finished ***")
```

C.2 Optuna Function Definitions for Hyperparameter Search & Optimization

This script includes the functions necessary to run the hyperparameter optimization procedure. The sub-function tasks include defining the specified model object, building the learning datasets, and evaluating the networks based on the objective function. Additionally, a supervisory program is defined which is called by the main program. This function runs the optimization trials, calls the sub-functions, and summarizes the results for the main program to interpret.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 21 14:29:17 2021

Objective functions for all NNs
Includes getting the model and datasets

@author: cpras
"""

#%% Imports

from NN_classes import FFN, GRU, TransformerModel, DARNN
from DL_classes import CustomTensorDataset
# from Train_functions import train_FFN, train_GRU, train_Transformer,
    train_DARNN
from main_Train import train_PAFP

import torch
import torch.nn as nn
import torch.optim as optim  # ADAM, SGD, etc
# import torch.utils.data.SubsetRandomSampler as SubsetRandomSampler

import optuna
from optuna.trial import TrialState
import pickle as pickle # import pickle5 as pickle

import numpy as np
import pandas as pd
import os
import time as timer

#%% Get Model

def define_model(trial, model_name, sequence_length, device, constants):

    # Unpack constants
    input_size = constants['input size']
```

```
38      output_size = constants['output size']
39
40      # Which model type are we optimizing?
41      if model_name == 'FFN':
42          num_layers = trial.suggest_int("num_layers", 1, 3)
43          hidden_size_power = trial.suggest_int("hidden_size_power", 4, 9)
44          hidden_size = 2**hidden_size_power
45          dropout = trial.suggest_float("dropout", 0.1, 0.5)
46
47          model = FFN(input_size, hidden_size, output_size, num_layers,
                sequence_length, dropout)
48
49      elif model_name == 'GRU':
50          num_layers = trial.suggest_int("num_layers", 1, 3)
51          hidden_size_power = trial.suggest_int("hidden_size_power", 4, 9)
52          hidden_size = 2**hidden_size_power
53          dropout = trial.suggest_float("dropout", 0.1, 0.5)
54
55          model = GRU(input_size, hidden_size, output_size, num_layers,
                sequence_length, dropout, device)
56
57      elif model_name == 'DA-RNN':
58          hidden_size_power = trial.suggest_int("hidden_size_power", 4, 9)
59          hidden_size = 2**hidden_size_power
60          P_power = trial.suggest_int("decoder_size_power", 4, 7)
61          P = 2**P_power
62
63          model = DARNN(input_size, hidden_size, P, sequence_length,
                device)
64
65      else:
66          print('Model Name is not one of the four options!!')
67          print('Must be FFN, GRU, Transformer, or DA-RNN')
68          model = []
69
70      return model
71
72  #%% Get Datasets
73
74  def get_dataLoaders(trial, sequence_length, constants, train_bool):
75
76      # Unpack constants
77      input_size = constants['input size']
78      output_size = constants['output size']
79      num_trials = constants['number of walking trials']
80      num_timestepsPerTrial = constants['number of timesteps per trial']
81      Data = constants['dataset']
82      prediction_horizon = constants['pred_horizon']
83
84      # Temporal Processing
```

```
85
86      print('Processing Time Series Data...')
87
88      # Inputs and Outputs
89      features = Data['Features']
90      responses = Data['Responses']
91
92      # Pre-allocate
93      X = np.zeros((num_trials, num_timestepsPerTrial, sequence_length,
            input_size))
94      y = np.zeros((num_trials, num_timestepsPerTrial, sequence_length,
            output_size))
95      target = np.zeros((num_trials, num_timestepsPerTrial, output_size))
96
97      for n in range(num_trials):
98
99          data_x = pd.DataFrame(features[n,:,:])
100         data_y = pd.DataFrame(responses[n,:])
101
102         for i, name in enumerate(list(data_x.columns)):
103             for j in range(sequence_length):
104                 X[n, :, j, i] = data_x[name].shift(sequence_length - j -
                        1).fillna(method="bfill")
105
106         for j in range(sequence_length):
107             y[n,:,j,0] = data_y[0].shift(sequence_length - j - 1).fillna
                    (method="bfill")
108
109         # prediction_horizon = 1
110         target[n,:,0] = data_y[0].shift(-prediction_horizon).fillna(
                method="ffill").values
111
112
113     X = X[:,sequence_length:-1]
114     y = y[:,sequence_length:-1]
115     target = target[:,sequence_length:-1]
116
117     # Split Dataset - Train, Val, Test
118
119     print('Splitting Data into Training, Validation, and Test Sets...')
120
121     timesteps = num_timestepsPerTrial - sequence_length - 1
122
123     tr_percent = 0.70
124     val_percent = 0.15
125     test_percent = 0.15
126
127     train_length = int(np.round(timesteps*tr_percent))
128     test_length = int(np.floor(timesteps*test_percent))
129     val_length = timesteps - train_length - test_length
```

```
130
131     X_train = X[:,:train_length]
132     X_val = X[:,train_length:train_length+val_length]
133     X_test = X[:,train_length+val_length:]
134     target_train = target[:,:train_length]
135     target_val = target[:,train_length:train_length+val_length]
136     target_test = target[:,train_length+val_length:]
137
138     # Data Scaling
139
140     print('Scaling Data from [0,1]...')
141
142     X_train_max = X_train.max(axis=(0,1,2))
143     X_train_min = X_train.min(axis=(0,1,2))
144     target_train_max = target_train.max(axis=(0,1))
145     target_train_min = target_train.min(axis=(0,1))
146
147     X_train = (X_train - X_train_min) / (X_train_max - X_train_min)
148     X_val = (X_val - X_train_min) / (X_train_max - X_train_min)
149     X_test = (X_test - X_train_min) / (X_train_max - X_train_min)
150
151     # target_train = (target_train - target_train_min) / (
            target_train_max - target_train_min)
152     # target_val = (target_val - target_train_min) / (target_train_max -
            target_train_min)
153     # target_test = (target_test - target_train_min) / (target_train_max
            - target_train_min)
154
155     # Convert to Tensors
156
157     print('Create Pytorch Data Loaders...')
158
159     X_train_t = torch.Tensor(X_train)
160     X_val_t = torch.Tensor(X_val)
161     X_test_t = torch.Tensor(X_test)
162     target_train_t = torch.Tensor(target_train)
163     target_val_t = torch.Tensor(target_val)
164     target_test_t = torch.Tensor(target_test)
165
166     # Add Guassian Noise to Training and Validation Inputs
167     if train_bool:
168         noiseSTD = trial.suggest_float("noiseSTD", 0.1, 1, log=False)
169     else:
170         noiseSTD = 0
171
172     Xtrain_noisy = X_train_t  + noiseSTD*torch.randn(X_train_t.size())
173     Xval_noisy = X_val_t + noiseSTD*torch.randn(X_val_t.size())
174
175     ## Create Data Loaders
176     if train_bool:
```

```
177
178         # Suggest a batch size for optimization
179         batch_size_power = trial.suggest_int("batch_size_power", 4, 8)
180         batch_size = 2**batch_size_power
181
182         # (timesteps, trials, sequence, features)
183         # This way we are dividing batches by timesteps and not trials
184         Xtrain_noisy = Xtrain_noisy.permute(1,0,2,3)
185         # X_val_t = X_val_t.permute(1,0,2,3)
186         # X_test_t = X_test_t.permute(1,0,2,3)
187         target_train_t = target_train_t.permute(1,0,2)
188         # target_val_t = target_val_t.permute(1,0,2)
189         # target_test_t = target_test_t.permute(1,0,2)
190
191         # Create datasets
192         train_dataset = CustomTensorDataset([Xtrain_noisy,
                target_train_t])
193         val_dataset = CustomTensorDataset([X_val_t, target_val_t])
194         test_dataset = CustomTensorDataset([X_test_t, target_test_t])
195
196         # dataloader options
197         train_shuffle = False   # data reshuffled at every epoch
198         batch_drop = True       # drop the last incomplete batch, if the
                dataset size is not divisible by the batch size
199         # sampler = SubsetRandomSampler()
200
201         # Create Dataloaders
202         train_loader = torch.utils.data.DataLoader(train_dataset,
                batch_size=batch_size,
203                                                    shuffle=train_shuffle
                                                       , drop_last=
                                                       batch_drop)
204         val_loader = torch.utils.data.DataLoader(val_dataset,batch_size
                =1,shuffle=False)
205         test_loader = torch.utils.data.DataLoader(test_dataset,
                batch_size=1,shuffle=False)
206
207     else:
208
209         # Create Datasets (trials, timesteps, sequence, features)
210         train_dataset = CustomTensorDataset([X_train_t, target_train_t])
211         val_dataset = CustomTensorDataset([X_val_t, target_val_t])
212         test_dataset = CustomTensorDataset([X_test_t, target_test_t])
213
214         # Create Dataloaders
215         train_loader = torch.utils.data.DataLoader(train_dataset,
                batch_size=1,shuffle=False)
216         val_loader = torch.utils.data.DataLoader(val_dataset,batch_size
                =1,shuffle=False)
```

```
217         test_loader = torch.utils.data.DataLoader(test_dataset,
                batch_size=1,shuffle=False)
218
219
220
221     return train_loader, val_loader, test_loader
222
223 #%% Objective
224
225 def objective(trial, model_name, constants):
226
227     # Empty GPU Cache
228     torch.cuda.empty_cache()
229
230     # Unpack constants
231     device = constants['device']
232     lr_step_size = constants['learning rate scheduler delay']
233     lr_patience = constants['learning rate scheduler patience']
234     lr_min = constants['minimum learning rate']
235     criterion = constants['loss function']
236     num_epochs = constants['max number of epochs']
237
238     # Start timer
239     start = timer.time()
240
241     # Suggest a sequence length
242     # one step ~= 153 points
243     # x = (sub sample freq/120)*(step length/2) = (30/120)*(153/2)
244     sequence_length = trial.suggest_int("sequence_length", 2, 20, log=
            False)
245
246     # Generate the Model
247     model = define_model(trial, model_name, sequence_length, device,
            constants).to(device)
248
249     # Get Training Dataset with Added Noise
250     train_loader, val_loader, test_loader = get_dataLoaders(trial,
            sequence_length, constants, train_bool=True)
251
252     # Optimizer Options
253     optimizer_name = "AdamW"
254     learning_rate = trial.suggest_float("learning_rate", 1e-5, 1e-1, log
            =True)
255     weight_decay = trial.suggest_float("weight_decay", 1e-5, 1e-1, log=
            False)
256     gamma = trial.suggest_float("gamma", 0.1, 0.9, log=False)
257
258     # Generate the Optimizer
259     optimizer = getattr(optim, optimizer_name)(model.parameters(), lr=
            learning_rate, weight_decay=weight_decay)
```

```
260
261     # Generate the Learning Rate Scheduler
262     scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, factor=
            gamma,
263                                                      patience=lr_patience,
                                                             cooldown=
                                                             lr_step_size,
264                                                      min_lr=lr_min, verbose=
                                                             True)
265
266     # Prints Params
267     print("\nTrain {} Trial {} with Params: ".format(model_name, trial.
            number))
268     for key, value in trial.params.items():
269         print("    {}: {}".format(key, value))
270     print('\n')
271
272     # Train
273     MSE = train_PAFP(trial, model_name, model, scheduler, optimizer,
            criterion,
274                      train_loader, val_loader, num_epochs, device)
275
276     # Train time
277     train_time = timer.time() - start
278
279     # Objective function
280     w_mse = 1 # weight associated with validation MSE
281     w_time = 0 # weight associated with training time
282     J = (w_mse*MSE) + (w_time*train_time) # Cost
283
284     return J
285
286 #%% Optimization
287
288 def optimize_hyperparams(model_name, constants):
289
290     # Unpack constants
291     PATH = constants['results path']
292     optuna_trials = constants['number of optuna trials']
293     optuna_timeout = constants['optuna timeout']
294
295     # Create an optimization project
296     print('\n')
297     study = optuna.create_study(study_name = "Optimize_{}".format(
            model_name), direction="minimize")
298
299     # Optimize the network
300     study.optimize(lambda trial: objective(trial, model_name, constants)
            ,
```

```
301                     n_trials=optuna_trials, timeout=optuna_timeout,
                            gc_after_trial=True)
302
303     # Summary Report
304     pruned_trials = study.get_trials(deepcopy=False, states=[TrialState.
            PRUNED])
305     complete_trials = study.get_trials(deepcopy=False, states=[
            TrialState.COMPLETE])
306
307     filename = os.path.join(PATH, model_name, 'Optimization_Results.txt'
            )
308     print("{} Study statistics: ".format(model_name), file=open(filename
            , "a"))
309     print("  Number of finished trials: ", len(study.trials), file=open(
            filename, "a"))
310     print("  Number of pruned trials: ", len(pruned_trials), file=open(
            filename, "a"))
311     print("  Number of complete trials: ", len(complete_trials), file=
            open(filename, "a"))
312
313     print("\n{} Best trial:".format(model_name), file=open(filename, "a"
            ))
314     trial = study.best_trial
315
316     print("\n  Validation Obj. Func. Value: ", trial.value, file=open(
            filename, "a"))
317
318     print("\n  {} Params: ".format(model_name), file=open(filename, "a")
            )
319     for key, value in trial.params.items():
320         print("    {}: {}".format(key, value), file=open(filename, "a"))
321
322     # Load the best model
323     pickle_file = "{}_{}.pickle".format(model_name, study.best_trial.
            number)
324     with open(pickle_file, "rb") as fin:
325         best_model = pickle.load(fin)
326
327     return best_model, trial
```

C.3 Neural Network Training & Validation Evaluation Function Definitions

This script executes the training and validation loops for each of the neural networks.

```
#%% Imports

from DL_classes import EarlyStopping

import torch
import numpy as np

import optuna
import pickle as pickle # import pickle5 as pickle

from tqdm import tqdm

from GPUtil import showUtilization as gpu_usage


#%% Function
def train_PAFP(trial, model_name, model, scheduler, optimizer, criterion
    , train_loader, val_loader, num_epochs, device):

    train_length = train_loader.dataset.data.shape[0]
    val_length = val_loader.dataset.data.shape[0]

    num_trials = train_loader.dataset.data.shape[1]

    # to track the training loss as the model trains
    train_losses = []
    # to track the validation loss as the model trains
    valid_losses = []
    # to track the average training loss per epoch as the model trains
    avg_train_losses = []
    # to track the average validation loss per epoch as the model trains
    avg_valid_losses = []

    # early stopping patience; how long to wait after last time
       validation loss improved.
    patience = 10
    # initialize the early_stopping object
    path = f"{model_name}_checkpoint.pt"
    early_stopping = EarlyStopping(patience=patience, verbose=True, path
       =path)

    for epoch in range(num_epochs):

        ####################
        # Train the model #
        ####################
```

```
44          model.train() # prep model for training
45
46          loop = tqdm(enumerate(train_loader), total=len(train_loader),
                leave=False)
47          for (idx, batch) in loop: # in enumerate(train_loader)
48
49              # NOTE:
50                  # Create for loop here going through the trials in dim=1
51                  # This way, the batch can stay as dimensions (timesteps/
                        batch, seq, features)
52
53              # to track training losses across batches and trials
54              current_iter_train_loss = []
55
56              if model_name == 'GRU':
57                  # Init hidden state
58                  batch_size = batch[0].shape[0]
59                  h = model.init_hidden(batch_size = batch_size)
60
61              for t in range(num_trials):
62                  batch_x = batch[0][:,t].squeeze().to(device)
63                  batch_y = batch[1][:,t].squeeze().to(device)
64
65                  # forward pass: compute predicted outputs by passing
                        inputs to the model
66                  # out = model(batch_x.float(),seqMode)
67                  if model_name == 'GRU':
68                      out,h = model(batch_x.float(),h)
69                      out = out[:,-1,0]
70                      h = h.detach() # detach hidden in between batches
71                  else:
72                      out = model(batch_x.float())
73
74                  # calculate the loss
75                  yTrue = batch_y.float() # seq2one
76                  loss = criterion(out.squeeze(), yTrue) # if seq to one,
                        only last timestep ([:,-1]) is used for loss
77
78                  # record training loss
79                  current_iter_train_loss.append(loss.item())
80                  train_losses.append(loss.item())
81
82                  # Backward
83                  optimizer.zero_grad() # clear the gradients of all
                        optimized variables
84                  loss.backward() # backward pass: compute gradient of the
                        loss with respect to model parameters
85                  torch.nn.utils.clip_grad_norm_(model.parameters(),
                        max_norm=1) # avoid gradient clipping
86
```

```
87                  # Gradient Descent or ADAM step
88                  optimizer.step()  # update the weights
89
90                  # Update Progress Bar
91                  loop.set_description(f"{model_name} Trial {trial.number}
                        Epoch [{epoch + 1}/{num_epochs}]")
92                  loop.set_postfix(loss=loss.item(), trial=t+1)
93                  # loop.set_postfix(trial=t+1)
94
95                  # delete intermediate values
96                  del batch_x, batch_y, out, yTrue, loss
97                  torch.cuda.empty_cache()
98
99          ######################
100         # Validate the model #
101         ######################
102         model.eval() # prep model for evaluation
103
104         with torch.no_grad():
105
106             for (idx, batch) in enumerate(val_loader):
107
108                 # NOTE:
109                     # Same for loop here
110
111                 if model_name == 'GRU':
112                     # Init hidden state
113                     batch_size = batch[0].shape[1]
114                     h_val = model.init_hidden(batch_size = batch_size)
115                     # h_val = model.init_hidden(batch_size = val_length)
116                 else:
117                     h_val = []
118
119                 batch_x = batch[0].squeeze().to(device)
120                 batch_y = batch[1].squeeze().to(device)
121
122                 # forward pass: compute predicted outputs by passing
                        inputs to the model
123                 if model_name == 'GRU':
124                     out,h_val = model(batch_x.float(),h_val)
125                     out = out[:,-1,0]
126                     h_val = h_val.detach() # detach hidden in between
                            batches
127                 else:
128                     out = model(batch_x.float())
129
130                 # calculate the loss
131                 yTrue = batch_y.float() # seq2seq
132                 loss = criterion(out.squeeze(), yTrue)
133
```

```
                # record validation loss
                valid_losses.append(loss.item())

                # delete intermediate values
                del batch_x, batch_y, out, yTrue, loss
                torch.cuda.empty_cache()

                # for t in range(num_trials):
                #     batch_x = batch[0][:,t].squeeze().to(device)
                #     batch_y = batch[1][:,t].squeeze().to(device)

                #     # batch_x = batch[0].squeeze().to(device)
                #     # batch_y = batch[1].squeeze().to(device)

                #     # forward pass: compute predicted outputs by passing inputs to the model
                #     if model_name == 'GRU':
                #         out,h_val = model(batch_x.float(),h_val)
                #         out = out[:,-1,0]
                #         h_val = h_val.detach() # detach hidden in between batches
                #     else:
                #         out = model(batch_x.float())

                #     # calculate the loss
                #     yTrue = batch_y.float() # seq2seq
                #     loss = criterion(out.squeeze(), yTrue)

                #     # record validation loss
                #     valid_losses.append(loss.item())

                #     # delete intermediate values
                #     del batch_x, batch_y, out, yTrue, loss
                #     torch.cuda.empty_cache()

        # print training/validation statistics
        # calculate average loss over an epoch
        train_loss = np.average(train_losses)
        valid_loss = np.average(valid_losses)
        avg_train_losses.append(train_loss)
        avg_valid_losses.append(valid_loss)
        epoch_len = len(str(num_epochs))
        print()
        print_msg = (f'{model_name} Trial {trial.number} Epoch [{epoch+1}/{num_epochs}] ' +
                     f'train_loss: {train_loss:.5f} ' +
                     f'valid_loss: {valid_loss:.5f}')
        print(print_msg)
        print()

```

```
181         # Update the learning rate
182         scheduler.step(valid_loss)
183 
184         # clear lists to track next epoch
185         train_losses = []
186         valid_losses = []
187 
188         ## Optuna
189         MSE = valid_loss
190         trial.report(MSE, epoch)
191         # Handle pruning based on the intermediate value.
192         if trial.should_prune():
193             raise optuna.exceptions.TrialPruned()
194 
195         # early_stopping needs the validation loss to check if it has
               decresed,
196         # and if it has, it will make a checkpoint of the current model
197         early_stopping(valid_loss, model)
198 
199         if early_stopping.early_stop:
200             print("Early stopping")
201             break
202 
203         # GPU Memory
204         print("GPU Usage after Train/Val Epoch")
205         gpu_usage()
206         print('\n')
207 
208     ## Save a trained model to a file.
209     with open("{}_{}.pickle".format(model_name,trial.number), "wb") as
           fout:
210         pickle.dump(model, fout)
211 
212     return MSE
```

C.4 Neural Network Evaluation Function Definitions using Test Dataset

This script evaluates each of the trained neural networks using the test dataset. The results are visualized using a plot of the test time series prediction results, an error histogram of the test results, and a plot of the full time series prediction results (training, validation, and test datasets).

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 21 12:58:18 2021

Functions to Test each NN

@author: cpras
"""

#%% Imports

import torch

import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error,
    mean_absolute_percentage_error
from scipy.stats import pearsonr
import scipy as scipy
import os

import matplotlib.pyplot as plt
from matplotlib import colors
from matplotlib.ticker import PercentFormatter
import matplotlib.ticker as tick

#%% Evaluate the Networks on the Test Dataset

def test_network(test_loader, model, model_name, criterion, PATH, device
    , pred_hor):

    test_length = test_loader.dataset.data.shape[1]
    gru_flag = False

    model.eval()

    test_loss = 0.0
    RMSE = []
    PCC = []
    ROM = []
```

```
    RMSE_per = []

    target = []
    pred = []

    # ***************
    # Note: include code that turns gradients off to save memory
    # >> is_train = False
    # >> with torch.set_grad_enabled(is_train):
    # OR if that doesn't work
    # >> torch.set_grad_enabled(False)
    # >> with torch.no_grad():
    #
    # Be sure to also remove the device dependencies as well
    # ***************
    with torch.no_grad():

        for (idx, batch) in enumerate(test_loader):

            batch_x = batch[0].squeeze().to(device)
            batch_y = batch[1].squeeze().to(device)

            # forward pass: compute predicted outputs by passing inputs
               to the model
            if model_name == 'FFN':
                output = model(batch_x.float())

            elif model_name == 'GRU':
                if gru_flag == False:
                    h = model.init_hidden(batch_size = test_length)
                    gru_flag = True

                output,h = model(batch_x.float(),h)
                output = output[:,-1,0]

            elif model_name == 'Transformer':
                output = model(batch_x.float())
                output = output[:,-1,0]

            elif model_name == 'DA-RNN':
                output = model(batch_x.float())

            else:
                print('Model Name is not one of the four options!!')
                print('Must be FFN, GRU, Transformer, or DA-RNN')
                model = []

            # calculate the loss
            yPred = output.squeeze().cpu()[:-pred_hor]
            yTrue = batch_y.float().cpu()[:-pred_hor]
```

```
89              loss = criterion(yPred, yTrue)
90              # update test loss
91              test_loss += loss.item()*yPred.size(0)
92              # compare predictions to true label
93              rms = mean_squared_error(yTrue.detach().numpy(), yPred.
                    detach().numpy(), squared=False)
94              RMSE.append(rms)
95              pcc , _ = pearsonr(yTrue.detach().numpy(), yPred.detach().
                    numpy())
96              PCC.append(pcc)
97              rom = np.ptp(yTrue)
98              ROM.append(rom)
99              RMSE_per.append((rms/rom)*100)
100             # for plotting
101             target.append(yTrue.detach().numpy())
102             pred.append(yPred.detach().numpy())
103
104             # delete intermediate values
105             del batch_x, batch_y, output, yPred, yTrue, loss
106
107     # calculate and print avg test loss
108     filename = os.path.join(PATH, model_name, 'Optimization_Results.txt'
            )
109     test_loss = test_loss/test_length
110     print('\nTest Loss: {:.6f}\n'.format(test_loss), file=open(filename,
            "a"))
111     print('RMSE: {:.6f} ({:.6f}) Nm/kg\n'.format(np.average(RMSE), np.
            std(RMSE)), file=open(filename, "a"))
112     print('% RMSE: {:.6f} ({:.6f}) Nm/kg\n'.format(np.average(RMSE_per),
            np.std(RMSE_per)), file=open(filename, "a"))
113     print('PCC: {:.6f} ({:.6f})\n'.format(np.average(PCC), np.std(PCC)),
            file=open(filename, "a"))
114
115     return target, pred, np.average(RMSE), test_loss, np.average(PCC)
116
117 #%% Visualize the Test Results
118
119 def visualize_results(model, model_name, train_loader, val_loader,
        test_loader,
120                       num_trials, target, pred, PATH, fs, device,
                            pred_hor):
121
122     train_length = train_loader.dataset.data.shape[1]
123     val_length = val_loader.dataset.data.shape[1]
124     test_length = test_loader.dataset.data.shape[1]
125
126     Targets = np.transpose(np.array(target))[:-pred_hor]
127     Predictions = np.transpose(np.array(pred))[:-pred_hor]
128     t = np.linspace(start=0,stop=len(Targets)/fs,num=len(Targets))
129
```

```
130     if model_name == 'FFN':
131         pred_color = 'blue'
132     elif model_name == 'GRU':
133         pred_color = 'green'
134     elif model_name == 'DA-RNN':
135         pred_color = 'orange'
136 
137     ######################
138     # Test Data Plots
139     ######################
140     for i in range(0,num_trials):
141         fig = plt.figure(figsize=(20,10))
142         plt.plot(t , Targets[:,i], label='MoCap', color='black',
                linewidth=4)
143         # plt.plot(t , Predictions[:,i], label='Predicted', color='red',
                linewidth=4)
144         plt.plot(t , Predictions[:,i], label='Predicted', color=
                pred_color,linewidth=4)
145         plt.xlabel('Time [s]',fontsize=20)
146         plt.ylabel('PAFP Torque [Nm/kg]',fontsize=20)
147         plt.xlim(0,t[-1]) # consistent scale
148         plt.grid(True)
149         plt.legend(prop={"size":20}, loc='upper left')
150         if model_name == 'DA-RNN':
151             plt.title(f'DA-GRU: Test Trial {i+1}',fontsize=30)
152         else:
153             plt.title(f'{model_name}: Test Trial {i+1}',fontsize=30)
154         plt.xticks(fontsize=20)
155         plt.yticks(fontsize=20)
156         plt.tight_layout()
157         plt.show()
158         # Save
159         filename = f'{model_name}-- Test{i+1}.jpg' # f'{model_name}--
                Test{i+1}.png'
160         savepath = os.path.join(PATH, model_name , filename)
161         fig.savefig(savepath, bbox_inches='tight')
162 
163     ######################
164     # Error Histogram
165     ######################
166     T = np.ndarray.flatten(Targets)
167     P = np.ndarray.flatten(Predictions)
168     err = T-P
169     meanErr = round(np.mean(err),4)
170     stdErr = round(np.std(err),4)
171     bins = 30
172 
173     fig = plt.figure(figsize=(20,10))
174     ax = fig.add_subplot(111)
```

```
175     N, bins, patches = ax.hist(err, bins=bins, density=True, histtype='
            bar')
176     fracs = N / N.max()
177     norm = colors.Normalize(fracs.min(), fracs.max())
178     # Now, we'll loop through our objects and set the color of each
            accordingly
179     for thisfrac, thispatch in zip(fracs, patches):
180         color = plt.cm.viridis(norm(thisfrac))
181         thispatch.set_facecolor(color)
182 
183     # Now we format the y-axis to display percentage
184     ax.yaxis.set_major_formatter(tick.PercentFormatter())
185 
186     plt.xlabel('Error [Nm/kg]',fontsize=20)
187     if model_name == 'DA-RNN':
188         plt.title('DA-GRU Testing Errors: ' + str(meanErr) + ' +/- ' +
                str(stdErr) + ' Nm/kg',fontsize=30)
189     else:
190         plt.title(f'{model_name} Testing Errors: ' + str(meanErr) + '
                +/- ' + str(stdErr) + ' Nm/kg',fontsize=30)
191     plt.grid(True)
192     plt.xticks(fontsize=20)
193     plt.yticks(fontsize=20)
194     plt.tight_layout()
195     plt.show()
196     # Save
197     filename = f'{model_name}-- Error Histogram.jpg' # f'{model_name}--
            Error Histogram.png'
198     savepath = os.path.join(PATH, model_name , filename)
199     fig.savefig(savepath, bbox_inches='tight')
200 
201     # Percent Error and Fit
202     percentError = mean_absolute_error(T, P)*100
203     percentFit = 100 - percentError
204 
205     ######################
206     # Full Trials
207     ######################
208 
209     model.eval()
210 
211     # ***************
212     # Note: include code that turns gradients off to save memory
213     # >> is_train = False
214     # >> with torch.set_grad_enabled(is_train):
215     # OR if that doesn't work
216     # >> torch.set_grad_enabled(False)
217     # >> with torch.no_grad():
218     #
219     # Be sure to also remove the device dependencies as well
```

```
220     # ***************
221
222     train_iterator = iter(train_loader)
223     val_iterator = iter(val_loader)
224     test_iterator = iter(test_loader)
225     for i in range(0,num_trials):
226         # print(f'--- Full Test Trial {i+1}')
227         try:
228             train_x, train_y = next(train_iterator)
229             val_x, val_y = next(val_iterator)
230             test_x, test_y = next(test_iterator)
231         except StopIteration:
232             train_iterator = iter(train_loader)
233             val_iterator = iter(val_loader)
234             test_iterator = iter(test_loader)
235
236             train_x, train_y = next(train_iterator)
237             val_x, val_y = next(val_iterator)
238             test_x, test_y = next(test_iterator)
239
240         # Model Predictions
241         if model_name == 'FFN':
242             train_out = model(torch.flatten(train_x,start_dim=2).squeeze
                    ().to(device).float())
243             val_out = model(torch.flatten(val_x,start_dim=2).squeeze().
                    to(device).float())
244             test_out = model(torch.flatten(test_x,start_dim=2).squeeze()
                    .to(device).float())
245             train_out = train_out.cpu().detach().numpy()
246             val_out = val_out.cpu().detach().numpy()
247             test_out = test_out.cpu().detach().numpy()
248
249         elif model_name == 'GRU':
250             h_tr = model.init_hidden(batch_size = train_length).to(
                    device)
251             h_val = model.init_hidden(batch_size = val_length).to(device
                    )
252             h_test = model.init_hidden(batch_size = test_length).to(
                    device)
253
254             train_out, _ = model(train_x.squeeze().to(device).float(),
                    h_tr)
255             val_out, _ = model(val_x.squeeze().to(device).float(),h_val)
256             test_out, _ = model(test_x.squeeze().to(device).float(),
                    h_test)
257             train_out = train_out[:,-1,0].cpu().detach().numpy()
258             val_out = val_out[:,-1,0].cpu().detach().numpy()
259             test_out = test_out[:,-1,0].cpu().detach().numpy()
260
261         elif model_name == 'Transformer':
```

```
262             train_out = model(train_x.squeeze().to(device).float())
263             val_out = model(val_x.squeeze().to(device).float())
264             test_out = model(test_x.squeeze().to(device).float())
265             train_out = train_out[:,-1,0].cpu().detach().numpy()
266             val_out = val_out[:,-1,0].cpu().detach().numpy()
267             test_out = test_out[:,-1,0].cpu().detach().numpy()
268 
269         elif model_name == 'DA-RNN':
270             train_out = model(train_x.squeeze().to(device).float()).cpu
                    ().detach().numpy()
271             val_out = model(val_x.squeeze().to(device).float()).cpu().
                    detach().numpy()
272             test_out = model(test_x.squeeze().to(device).float()).cpu().
                    detach().numpy()
273 
274         else:
275             print('Model Name is not one of the four options!!')
276             print('Must be FFN, GRU, Transformer, or DA-RNN')
277             model = []
278 
279         # True Vals
280         train_true = train_y.squeeze().float().detach().numpy()
281         val_true = val_y.squeeze().float().detach().numpy()
282         test_true = test_y.squeeze().float().detach().numpy()
283         true_vals = np.concatenate((train_true, val_true, test_true),
                axis=0)
284 
285         # Time vectors
286         time = np.linspace(start=30-len(true_vals)/fs,stop=30,num=len(
                true_vals))
287         time_tr = time[:train_length]
288         time_val = time[train_length:train_length+val_length]
289         time_test = time[train_length+val_length:]
290 
291         # Cut off prediction horizon
292         true_vals = true_vals[:-pred_hor]
293         test_out = test_out[:-pred_hor]
294         time = time[:-pred_hor]
295         time_test = time_test[:-pred_hor]
296 
297         # Plot
298         fig = plt.figure(figsize=(20,10))
299         plt.plot(time , true_vals, color='black', label='MoCap Targets',
                linewidth=4)
300         plt.plot(time_tr , train_out, color='blue', label='Train Pred.',
                linewidth=4)
301         plt.plot(time_val , val_out, color='green', label='Validation
                Pred.',linewidth=4)
302         plt.plot(time_test , test_out, color='red', label='Test Pred.',
                linewidth=4)
```

```
303         plt.xlabel('Time [s]',fontsize=20)
304         plt.ylabel('PAFP Torque [Nm/kg]',fontsize=20)
305         if model_name == 'DA-RNN':
306             plt.title(f'DA-GRU: Full Trial {i+1}',fontsize=30)
307         else:
308             plt.title(f'{model_name}: Full Trial {i+1}',fontsize=30)
309         plt.xlim(0,30)
310         plt.grid(True)
311         plt.legend(prop={"size":20}, loc='upper left')
312         plt.xticks(fontsize=20)
313         plt.yticks(fontsize=20)
314         plt.tight_layout()
315         plt.show()
316         # Save
317         filename = f'{model_name}-- Full Trial{i+1}.jpg' # f'{model_name
               }-- Full Trial{i+1}.png'
318         savepath = os.path.join(PATH, model_name , filename)
319         fig.savefig(savepath, bbox_inches='tight')
320
321
322     return percentFit, percentError
323
324 #%% Fitted Histograms
325
326 def fitted_histogram(target1, pred1, target2, pred2, target3, pred3,
       target4, pred4, PATH):
327
328     # with polynomial
329     fig = plt.figure(figsize=(20,10))
330     bins = 30
331
332     best_fit_line1, bins1 =  fit_line_calc(target1, pred1, bins, 'blue')
333     best_fit_line2, bins2 =  fit_line_calc(target2, pred2, bins, 'green'
          )
334     best_fit_line3, bins3 =  fit_line_calc(target3, pred3, bins, 'orange
          ')
335     best_fit_line4, bins4 =  fit_line_calc(target4, pred4, bins, 'red')
336
337     plt.plot(bins4, best_fit_line4, label='Analytical',color='red',
          linewidth=4)
338     plt.plot(bins1, best_fit_line1, label='FFN',color='blue', linewidth
          =4)
339     plt.plot(bins2, best_fit_line2, label='GRU',color='green', linewidth
          =4)
340     plt.plot(bins3, best_fit_line3, label='DA-RNN',color='orange',
          linewidth=4)
341     plt.legend(prop={"size":20}, loc='upper left')
342     plt.xticks(fontsize=20)
343     plt.yticks(fontsize=20)
344     plt.xlabel('Error [Nm/kg]',fontsize=20)
```

```
345     plt.ylabel('Probability',fontsize=20)
346     plt.title('Error Distribution for all Models',fontsize=30)
347     plt.grid(True)
348     plt.xticks(fontsize=20)
349     plt.yticks(fontsize=20)
350     plt.tight_layout()
351     plt.show()
352     # Save
353     filename = 'Error Histogram - All Models.jpg' # 'Error Histogram -
            All Models.png'
354     savepath = os.path.join(PATH, filename)
355     fig.savefig(savepath, bbox_inches='tight')
356 
357     return
358 
359 def nn_fitted_histogram(target1, pred1, target2, pred2, target3, pred3,
        PATH):
360     # just NNs
361     fig = plt.figure(figsize=(20,10))
362     bins = 30
363 
364     best_fit_line1, bins1 =  fit_line_calc(target1, pred1, bins, 'blue')
365     best_fit_line2, bins2 =  fit_line_calc(target2, pred2, bins, 'green'
            )
366     best_fit_line3, bins3 =  fit_line_calc(target3, pred3, bins, 'orange
            ')
367 
368     plt.plot(bins1, best_fit_line1, label='FFN',color='blue', linewidth
            =4)
369     plt.plot(bins2, best_fit_line2, label='GRU',color='green', linewidth
            =4)
370     plt.plot(bins3, best_fit_line3, label='DA-GRU',color='orange',
            linewidth=4)
371     plt.legend(prop={"size":20}, loc='upper left')
372     plt.xticks(fontsize=20)
373     plt.yticks(fontsize=20)
374     plt.xlabel('Error [Nm/kg]',fontsize=20)
375     plt.ylabel('Probability',fontsize=20)
376     plt.title('Error Distribution for all Models',fontsize=30)
377     plt.grid(True)
378     plt.xticks(fontsize=20)
379     plt.yticks(fontsize=20)
380     plt.tight_layout()
381     plt.show()
382     # Save
383     filename = 'Error Histogram - All NNs.jpg' # 'Error Histogram - All
            NNs.png'
384     savepath = os.path.join(PATH, filename)
385     fig.savefig(savepath, bbox_inches='tight')
386     return
```

```
387
388 def fit_line_calc(target, pred, bins, c):
389
390     Targets = np.transpose(np.array(target))
391     Predictions = np.transpose(np.array(pred))
392     T = np.ndarray.flatten(Targets)
393     P = np.ndarray.flatten(Predictions)
394     err = T-P
395
396     _, bins, _ = plt.hist(err, 20, density=1, alpha=0.5, color=c)
397
398     mu, sigma = scipy.stats.norm.fit(err)
399     best_fit_line = scipy.stats.norm.pdf(bins, mu, sigma)
400
401     return best_fit_line, bins
```

C.5 Neural Network Class Variable Definitions

This file includes the class object definitions of the neural networks which include the architecture of the model and forward prediction functionality.

```
"""
Created on Wed Apr 21 12:32:47 2021

Nueral Network Model Class Definitions

@author: cpras
"""

#%% Imports

import torch
import torch.nn as nn
import torch.nn.functional as F  # relu, tanh, etc.
import math

#%% Feedforward NN (FFN)

class FFN(nn.Module):
    def __init__(self,input_size, hidden_size, output_size, num_layers,
       sequence_length, dropout):
        super(FFN, self).__init__()
        self.dropout = nn.Dropout(dropout)
        self.sequence_length = sequence_length
        self.input_size = input_size * self.sequence_length
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.num_layers = num_layers

        self.act = F.relu
        self.final_layer = nn.Linear(self.hidden_size, self.output_size)
        self.input_layer = nn.Linear(self.input_size, self.hidden_size)
        self.hidden = []
        for i in range(self.num_layers):
            self.hidden.append(nn.Linear(self.hidden_size,self.
               hidden_size))
        self.hidden = nn.ModuleList(self.hidden)

    def forward(self,x):
        y = torch.flatten(x,start_dim=1)
        y = self.dropout(self.act(self.input_layer(y)))
        for i in range(self.num_layers):
            y = self.dropout(self.act(self.hidden[i](y)))
        out = self.final_layer(y)
        return out
```

```
43
44      # def __init__(self, input_size, hidden_size, output_size, dropout):
45      #     super(FFN, self).__init__()
46      #     self.fc1 = nn.Linear(input_size, hidden_size)
47      #     self.fc2 = nn.Linear(hidden_size, hidden_size)
48      #     self.out = nn.Linear(hidden_size, output_size)
49      #     self.dropout = nn.Dropout(dropout)
50
51      # def forward(self, x):
52      #     x = torch.flatten(x,start_dim=1)
53      #     y = self.dropout(F.relu(self.fc1(x)))
54      #     y = self.dropout(F.relu(self.fc2(y)))
55      #     y = self.out(y)  # no activation
56      #     return y
57
58  #%% Gated Recurrent Unit NN (GRU)
59
60  class GRU(nn.Module):
61      def __init__(self, input_size, hidden_size, output_size, num_layers,
            sequence_length, dropout, device):
62          super(GRU, self).__init__()
63          self.hidden_size = hidden_size
64          self.num_layers = num_layers
65          if self.num_layers > 1:
66              self.gru = nn.GRU(input_size, hidden_size,
67                                num_layers, batch_first=True, dropout=
                                      dropout)
68          else:
69              self.gru = nn.GRU(input_size, hidden_size,
70                                num_layers, batch_first=True)
71          self.fc = nn.Linear(hidden_size, output_size)
72          self.relu = nn.ReLU()
73          self.device = device
74
75      def forward(self, x, h0):
76          out, h0 = self.gru(x, h0)
77          out = self.fc(self.relu(out))
78          return out, h0
79
80      def init_hidden(self, batch_size):
81          weight = next(self.parameters()).data
82          hidden = weight.new(self.num_layers, batch_size, self.
                hidden_size).zero_().to(self.device)
83          return hidden
84
85  #%% Transformer Network (Transformer)
86
87  class PositionalEncoding(nn.Module):
88
89      def __init__(self, d_model, dropout, max_len=5000):
```

```
90          super(PositionalEncoding, self).__init__()
91          self.dropout = nn.Dropout(p=dropout)
92
93          pe = torch.zeros(max_len, d_model)
94          position = torch.arange(0, max_len, dtype=torch.float).unsqueeze
                (1)
95          div_term = torch.exp(torch.arange(0, d_model, 2).float() * (-
                math.log(10000.0) / d_model))
96          pe[:, 0::2] = torch.sin(position * div_term)
97          pe[:, 1::2] = torch.cos(position * div_term)
98          pe = pe.unsqueeze(0).transpose(0, 1)
99          #pe.requires_grad = False
100         self.register_buffer('pe', pe)
101
102     def forward(self, x):
103         x + self.pe[:x.size(0), :]
104         return self.dropout(x)
105
106
107 class TransformerModel(nn.Module):
108     def __init__(self,nout,ninp,nhead,nhid,num_layers,dropout):
109         super(TransformerModel, self).__init__()
110         self.model_type = 'Transformer'
111         self.src_mask = None
112
113         self.pos_encoder = PositionalEncoding(ninp,dropout)
114         self.encoder_layer = nn.TransformerEncoderLayer(d_model=ninp,
                nhead=nhead,
115                                                         dim_feedforward=
                                                            nhid, dropout
                                                            =dropout)
116         self.transformer_encoder = nn.TransformerEncoder(self.
                encoder_layer, num_layers=num_layers)
117         self.encoder = nn.Embedding(nout,ninp) # added
118         self.ninp = ninp
119         self.decoder = nn.Linear(ninp,nout)
120
121         self.init_weights()
122
123     def init_weights(self):
124         initrange = 0.1
125         self.encoder.weight.data.uniform_(-initrange, initrange) # added
126         self.decoder.bias.data.zero_()
127         self.decoder.weight.data.uniform_(-initrange, initrange)
128
129     def forward(self,src):
130         if self.src_mask is None or self.src_mask.size(0) != len(src):
131             device = src.device
132             mask = self._generate_square_subsequent_mask(len(src)).to(
                    device)
```

```
133             self.src_mask = mask
134
135         # src = self.encoder(src) * math.sqrt(self.ninp) # added
136         src = self.pos_encoder(src)
137         output = self.transformer_encoder(src,self.src_mask)
138         output = self.decoder(output) #seq2seq
139         return output
140
141     def _generate_square_subsequent_mask(self, sz):
142         mask = (torch.triu(torch.ones(sz, sz)) == 1).transpose(0, 1)
143         mask = mask.float().masked_fill(mask == 0, float('-inf')).
                masked_fill(mask == 1, float(0.0))
144         return mask
145
146 #%% Dual-Stage Attention-Based Gated Recurrent Unit (DA-RNN)
147
148 class InputAttentionEncoder(nn.Module):
149     def __init__(self, N, M, T, device, stateful=False):
150         """
151         :param: N: int
152             number of time serieses
153         :param: M:
154             number of LSTM units
155         :param: T:
156             number of timesteps
157         :param: stateful:
158             decides whether to initialize cell state of new time window
                    with values of the last cell state
159             of previous time window or to initialize it with zeros
160         """
161         super(self.__class__, self).__init__()
162         self.N = N
163         self.M = M
164         self.T = T
165         self.device = device
166
167         # self.encoder_lstm = nn.LSTMCell(input_size=self.N, hidden_size
                =self.M)
168         self.encoder_gru = nn.GRUCell(input_size=self.N, hidden_size=
                self.M)
169
170         #equation 8 matrices
171         # self.W_e = nn.Linear(2*self.M, self.T)
172         self.W_e = nn.Linear(self.M, self.T)
173         self.U_e = nn.Linear(self.T, self.T, bias=False)
174         self.v_e = nn.Linear(self.T, 1, bias=False)
175
176     def forward(self, inputs):
177         encoded_inputs = torch.zeros((inputs.size(0), self.T, self.M)).
                to(self.device)
```

```
178
179         #initiale hidden states
180         # s_tm1 = torch.zeros((inputs.size(0), self.M)).to(device)
181         h_tm1 = torch.zeros((inputs.size(0), self.M)).to(self.device)
182
183         for t in range(self.T):
184             #concatenate hidden states
185             # h_c_concat = torch.cat((h_tm1, s_tm1), dim=1)
186             h_c_concat = h_tm1
187
188             #attention weights for each k in N (equation 8)
189             x = self.W_e(h_c_concat).unsqueeze_(1).repeat(1, self.N, 1)
190             y = self.U_e(inputs.permute(0, 2, 1))
191             # y = self.U_e(inputs.permute(0, 1, 3, 2)) # CHANGED
192             z = torch.tanh(x + y)
193             e_k_t = torch.squeeze(self.v_e(z))
194
195             #normalize attention weights (equation 9)
196             alpha_k_t = F.softmax(e_k_t, dim=1)
197
198             #weight inputs (equation 10)
199             weighted_inputs = alpha_k_t * inputs[:, t, :]
200
201             # calculate next hidden states (equation 11)
202             # h_tm1, s_tm1 = self.encoder_lstm(weighted_inputs, (h_tm1,
                    s_tm1))
203             h_tm1 = self.encoder_gru(weighted_inputs, h_tm1)
204
205             encoded_inputs[:, t, :] = h_tm1
206         return encoded_inputs
207
208 class TemporalAttentionDecoder(nn.Module):
209     def __init__(self, M, P, T, device, stateful=False):
210         """
211         :param: M: int
212             number of encoder LSTM units
213         :param: P:
214             number of deocder LSTM units
215         :param: T:
216             number of timesteps
217         :param: stateful:
218             decides whether to initialize cell state of new time window
                    with values of the last cell state
219             of previous time window or to initialize it with zeros
220         """
221         super(self.__class__, self).__init__()
222         self.M = M
223         self.P = P
224         self.T = T
225         self.stateful = stateful
```

```
226         self.device = device
227 
228         # self.decoder_lstm = nn.LSTMCell(input_size=1, hidden_size=self
                .P)
229         self.decoder_gru = nn.GRUCell(input_size=1, hidden_size=self.P)
230 
231         #equation 12 matrices
232         self.W_d = nn.Linear(self.P, self.M)
233         self.U_d = nn.Linear(self.M, self.M, bias=False)
234         self.v_d = nn.Linear(self.M, 1, bias = False)
235 
236         #equation 15 matrix
237         self.w_tilda = nn.Linear(self.M, 1)
238 
239         #equation 22 matrices
240         self.W_y = nn.Linear(self.P + self.M, self.P)
241         self.v_y = nn.Linear(self.P, 1)
242 
243     def forward(self, encoded_inputs):
244 
245         #initializing hidden states
246         d_tm1 = torch.zeros((encoded_inputs.size(0), self.P)).to(self.
                device)
247         # s_prime_tm1 = torch.zeros((encoded_inputs.size(0), self.P)).to
                (device)
248 
249         for t in range(self.T):
250             #concatenate hidden states
251             # d_s_prime_concat = torch.cat((d_tm1, s_prime_tm1), dim=1)
252             d_s_prime_concat = d_tm1
253 
254             #print(d_s_prime_concat)
255             #temporal attention weights (equation 12)
256             x1 = self.W_d(d_s_prime_concat).unsqueeze_(1).repeat(1,
                    encoded_inputs.shape[1], 1)
257             y1 = self.U_d(encoded_inputs)
258             z1 = torch.tanh(x1 + y1)
259             l_i_t = self.v_d(z1)
260 
261             #normalized attention weights (equation 13)
262             beta_i_t = F.softmax(l_i_t, dim=1)
263 
264             #create context vector (equation_14)
265             c_t = torch.sum(beta_i_t * encoded_inputs, dim=1)
266 
267             # #concatenate c_t and y_t
268             # y_c_concat = torch.cat((c_t, y[:, t, :]), dim=1)
269             # #create y_tilda
270             # y_tilda_t = self.w_tilda(y_c_concat)
271 
```

```
272             y_tilda_t = self.w_tilda(c_t)
273
274             #calculate next hidden states (equation 16)
275             # d_tm1, s_prime_tm1 = self.decoder_lstm(y_tilda_t, (d_tm1,
                    s_prime_tm1))
276             d_tm1 = self.decoder_gru(y_tilda_t, d_tm1)
277
278         #concatenate context vector at step T and hidden state at step T
279         d_c_concat = torch.cat((d_tm1, c_t), dim=1)
280
281         #calculate output
282         y_Tp1 = self.v_y(self.W_y(d_c_concat))
283         return y_Tp1
284
285 class DARNN(nn.Module):
286     def __init__(self, N, M, P, T, device, stateful_encoder=False,
                stateful_decoder=False):
287         super(self.__class__, self).__init__()
288         self.device = device
289         self.encoder = InputAttentionEncoder(N, M, T, device,
                stateful_encoder).to(self.device)
290         self.decoder = TemporalAttentionDecoder(M, P, T, device,
                stateful_decoder).to(self.device)
291     def forward(self, X_history):
292         out = self.decoder(self.encoder(X_history))
293         return out
```

C.6 MATLAB-to-Python Data Processing Function Definitions

This file includes the functions required to load, process, and interpret the MATLAB data files.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 21 12:43:11 2021

Misc. Functions for Deep Learning Scripts

@author: cpras
"""
#%% Imports

import numpy as np
import scipy.io as spio

#%%
def loadmat(filename):
    '''
    this function should be called instead of direct spio.loadmat
    as it cures the problem of not properly recovering python
       dictionaries
    from mat files. It calls the function check keys to cure all entries
    which are still mat-objects

    from: `StackOverflow <http://stackoverflow.com/questions/7008608/
       scipy-io-loadmat-nested-structures-i-e-dictionaries>`_
    '''
    data = spio.loadmat(filename, struct_as_record=False, squeeze_me=
       True)
    return _check_keys(data)

def _check_keys(dict):
    '''
    checks if entries in dictionary are mat-objects. If yes
    todict is called to change them to nested dictionaries
    '''
    for key in dict:
        if isinstance(dict[key], spio.matlab.mio5_params.mat_struct):
            dict[key] = _todict(dict[key])
    return dict

def _todict(matobj):
    '''
    A recursive function which constructs from matobjects nested
       dictionaries
    '''
```

```
41      dict = {}
42      for strg in matobj._fieldnames:
43          elem = matobj.__dict__[strg]
44          if isinstance(elem, spio.matlab.mio5_params.mat_struct):
45              dict[strg] = _todict(elem)
46          else:
47              dict[strg] = elem
48      return dict
49
50  #Defining MAPE function
51  def MAPE(Y_actual,Y_Predicted):
52      mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
53      return mape
```

C.7 Miscellaneous Deep Learning Class Variable Definitions

This file includes class object definitions used in the neural network training procedure. The first class defines the structure of the dataset. The second class defines the early stopping protocol, which monitors the validation loss while training a PyTorch model and saves a checkpoint of the model each time the validation loss decreases.

```
"""
Created on Wed Apr 21 12:30:14 2021

Classes used across Deep Learning Scripts

@author: cpras
"""
#%% Dataset Class
from torch.utils.data import TensorDataset

class CustomTensorDataset(TensorDataset):
    """TensorDataset with support of transforms.
    """
    def __init__(self, tensors):
        assert all(tensors[0].size(0) == tensor.size(0) for tensor in
           tensors)
        self.tensors = tensors
        self.data = tensors[0]
        self.targets = tensors[1]


    def __getitem__(self, index):
        x = self.tensors[0][index]

        y = self.tensors[1][index]

        return x, y

#%% Early Stopping with Validation Data & Patience

import numpy as np
import torch

class EarlyStopping:
    """Early stops the training if validation loss doesn't improve after
        a given patience."""
    def __init__(self, patience=7, verbose=False, delta=1e-4, path='
       checkpoint.pt', trace_func=print):
        """
        Args:
```

```
38              patience (int): How long to wait after last time validation
                    loss improved.
39                              Default: 7
40              verbose (bool): If True, prints a message for each
                    validation loss improvement.
41                              Default: False
42              delta (float): Minimum change in the monitored quantity to
                    qualify as an improvement.
43                              Default: 0
44                              *** NOTE: default changed to 1e-4
45              path (str): Path for the checkpoint to be saved to.
46                              Default: 'checkpoint.pt'
47              trace_func (function): trace print function.
48                              Default: print
49          """
50          self.patience = patience
51          self.verbose = verbose
52          self.counter = 0
53          self.best_score = None
54          self.early_stop = False
55          self.val_loss_min = np.Inf
56          self.delta = delta
57          self.path = path
58          self.trace_func = trace_func
59      def __call__(self, val_loss, model):
60
61          score = -val_loss
62
63          if self.best_score is None:
64              self.best_score = score
65              self.save_checkpoint(val_loss, model)
66          elif score < self.best_score + self.delta:
67              self.counter += 1
68              self.trace_func(f'EarlyStopping counter: {self.counter} out
                    of {self.patience}')
69              if self.counter >= self.patience:
70                  self.early_stop = True
71          else:
72              self.best_score = score
73              self.save_checkpoint(val_loss, model)
74              self.counter = 0
75
76      def save_checkpoint(self, val_loss, model):
77          '''Saves model when validation loss decrease.'''
78          if self.verbose:
79              self.trace_func(f'Validation loss decreased ({self.
                    val_loss_min:.6f} --> {val_loss:.6f}).  Saving model ...'
                    )
80              print('\n')
81          torch.save(model.state_dict(), self.path)
```

```
82            self.val_loss_min = val_loss
```

C.8 Analytical Regression Model Training & Validation Evaluation Function

This script executes the training and k-folds validation for the analytical regression model, which is used as a baseline comparison for the DNN models.

```
"""
Created on Fri Oct 22 12:59:38 2021

@author: Chris Prasanna

Tools for simple polynomial fit (linear regression)
"""
#%% Imports

import numpy as np
from sklearn.model_selection import KFold
from scipy import optimize
from sklearn.metrics import mean_squared_error
from scipy.stats import pearsonr
import os

import matplotlib.pyplot as plt
from matplotlib import colors
from matplotlib.ticker import PercentFormatter
import matplotlib.ticker as tick

#%% Nominal Values
h = 0.134
d = 0.06
eta = 0.6
b = np.sqrt(d**2 + h**2)
theta0 = np.deg2rad(-4.5)
z1 = b**2 + d**2
z2 = -2*b*d
R1 = 19
R2 = 2*np.pi/0.004
k_tau = 0.00775
J_m = (4.09/1000)/(100^2)
J_g = (0.4/1000)/(100^2)
J_f = J_m / 10000
R_g = 3249/169
B = 0.01
mass = 85

#%% Functions
def f(x, a, b, c, d):
    # third-degree nonlinear (polynomial) model
    return a*x**3 + b*x**2 + c*x + d

```

```
45 def residual(p, x, y):
46     return y - f(x, *p)
47 
48 def fbeta(theta):
49     beta = np.arccos(d/b) + theta - theta0
50     return beta
51 
52 def fa(theta):
53     a = np.sqrt(b**2 + d**2 - 2*b*d*np.cos(fbeta(theta) + theta - theta0
       ));
54     return a
55 
56 def falpha(theta):
57     alpha = np.arcsin(np.divide(d*np.sin(fbeta(theta)),fa(theta)));
58     return alpha
59 
60 def fphi(theta):
61     phi = np.pi/2 - fbeta(theta) - falpha(theta)
62     return phi
63 
64 def transmission(theta):
65     R = d*np.cos(fphi(theta))*R1*R2*eta
66     return R
67 
68 def drivetrain_model(theta, current):
69     tau_a = (k_tau/mass)*np.multiply(transmission(theta),current)
70     return tau_a
71 
72 def plot_full(x, im, y, constants, best_model):
73 
74     PATH = constants['results path']
75     num_trials = constants['number of walking trials']
76     fs = constants['sub-sample freq']
77 
78     num_timestepsPerTrial = constants['number of timesteps per trial']
79     timesteps = num_timestepsPerTrial - 1
80     tr_percent = 0.70
81     val_percent = 0.15
82     test_percent = 0.15
83     train_length = int(np.round(timesteps*tr_percent))
84     test_length = int(np.floor(timesteps*test_percent))
85     val_length = timesteps - train_length - test_length
86 
87     x_train = x[:,:train_length]
88     x_val = x[:,train_length:train_length+val_length]
89     x_test = x[:,train_length+val_length:]
90     im_train = im[:,:train_length]
91     im_val = im[:,train_length:train_length+val_length]
92     im_test = im[:,train_length+val_length:]
93     y_train = y[:,:train_length]
```

```
94      y_val = y[:,train_length:train_length+val_length]
95      y_test = y[:,train_length+val_length:]
96
97      for i in range(0,num_trials):
98          # True Vals
99          train_true = y_train[i,:]
100         val_true = y_val[i,:]
101         test_true = y_test[i,:]
102         true_vals = np.concatenate((train_true, val_true, test_true),
                axis=0)
103
104         # Model outputs
105         train_out = f(x_train[i,:], *best_model) + drivetrain_model(
                x_train[i,:], im_train[i,:])
106         val_out = f(x_val[i,:], *best_model) + drivetrain_model(x_val[i
                ,:], im_val[i,:])
107         test_out = f(x_test[i,:], *best_model) + drivetrain_model(x_test
                [i,:], im_test[i,:])
108
109         # Time vectors
110         time = np.linspace(start=30-len(true_vals)/fs,stop=30,num=len(
                true_vals))
111         time_tr = time[:train_length]
112         time_val = time[train_length:train_length+val_length]
113         time_test = time[train_length+val_length:]
114
115         # Plot
116         fig = plt.figure(figsize=(20,10))
117         plt.plot(time , true_vals, color='black', label='MoCap Targets',
                linewidth=4)
118         plt.plot(time_tr , train_out, color='blue', label='Train Pred.',
                linewidth=4)
119         plt.plot(time_val , val_out, color='green', label='Validation
                Pred.',linewidth=4)
120         plt.plot(time_test , test_out, color='red', label='Test Pred.',
                linewidth=4)
121         plt.xlabel('Time [s]',fontsize=20)
122         plt.ylabel('PAFP Torque [Nm/kg]',fontsize=20)
123         plt.title(f'Analytical Regression Model: Full Trial {i+1}',
                fontsize=30)
124         plt.xlim(0,30)
125         plt.grid(True)
126         plt.legend(prop={"size":20}, loc='upper left')
127         plt.xticks(fontsize=20)
128         plt.yticks(fontsize=20)
129         plt.tight_layout()
130         plt.show()
131         # Save
132         filename = f'Analytical Regression Model -- Full Trial{i+1}.jpg'
                # f'Analytical Regression Model -- Full Trial{i+1}.png'
```

```
133         savepath = os.path.join(PATH, 'Polynomial' , filename)
134         fig.savefig(savepath, bbox_inches='tight')
135 
136     return
137 
138 def train_poly(constants):
139 
140     PATH = constants['results path']
141     num_trials = constants['number of walking trials']
142 
143     data = constants['dataset']
144     features = data['Features']
145     response_ = data['Responses']
146 
147     # Dataset splitting params
148     num_timestepsPerTrial = constants['number of timesteps per trial']
149     timesteps = num_timestepsPerTrial - 1
150     tr_percent = 0.70
151     val_percent = 0.15
152     test_percent = 0.15
153     train_length = int(np.round(timesteps*tr_percent))
154     test_length = int(np.floor(timesteps*test_percent))
155     val_length = timesteps - train_length - test_length
156 
157     # Get key features
158     ankle_angles_ = features[:,:,3] # CHECK INDEX TO MAKE SURE THIS IS
            CORRECT
159     current_ = features[:,:,0]
160 
161     # Split dataset
162     ankle_angles = ankle_angles_[:,:train_length+val_length]
163     current = current_[:,:train_length+val_length]
164     response = response_[:,:train_length+val_length]
165 
166     # Flatten
167     ankle_angles = ankle_angles.flatten()
168     current = current.flatten()
169     response = response.flatten()
170 
171     print('Nonlinear Least-Squares')
172 
173     # k-fold cross val params
174     k = 5
175     kf = KFold(n_splits=k, random_state=None, shuffle=True)
176 
177     # pre-allocate
178     mse = np.zeros(k)
179     p = np.zeros((k,4))
180 
181     # loop
```

```
182     ii = 0
183     for train_index, test_index in kf.split(response):
184         # print("TRAIN:", train_index, "TEST:", test_index)
185 
186         # Train
187         p0 = [1., 1., 1., 1.]
188         x = ankle_angles[train_index]
189         y = response[train_index] - drivetrain_model(x, current[
                train_index])
190         popt, pcov = optimize.leastsq(residual, p0, args=(x, y))
191 
192         # Val
193         xn = ankle_angles[test_index]
194         y_true = response[test_index]
195         y_pred = f(xn, *popt) + drivetrain_model(ankle_angles[test_index
                ], current[test_index])
196         mse[ii] = mean_squared_error(y_true, y_pred, squared=True)
197         p[ii,:] = popt
198         ii+=1
199 
200     best_model = p[np.argmin(mse),:]
201 
202     # Test
203     ankle_angles_test = ankle_angles_[:,train_length+val_length:]
204     current_test = current_[:,train_length+val_length:]
205     response_test = response_[:,train_length+val_length:]
206 
207     ankle_angles = ankle_angles_test.flatten()
208     current = current_test.flatten()
209     response = response_test.flatten()
210 
211     pred = f(ankle_angles, *best_model) + drivetrain_model(ankle_angles,
            current)
212     test_rmse = mean_squared_error(response, pred, squared=False)
213     test_pcc , _ = pearsonr(response, pred)
214 
215     # Visualize - Time Series
216     T = []
217     P = []
218     RMSE = []
219     PCC = []
220     ROM = []
221     RMSE_per = []
222     for i in range(0,num_trials):
223         y = response_test[i,:]
224         x = ankle_angles_test[i,:]
225         im = current_test[i,:]
226         t = np.linspace(start=0,stop=len(y)/30,num=len(y))
227 
228         yp = f(x, *best_model) + drivetrain_model(x, im)
```

```
229
230         # compare predictions to true label
231         rms = mean_squared_error(y, yp, squared=False)
232         RMSE.append(rms)
233         pcc , _ = pearsonr(y, yp)
234         PCC.append(pcc)
235         rom = np.ptp(y)
236         ROM.append(rom)
237         RMSE_per.append((rms/rom)*100)
238
239         # Plot
240         fig = plt.figure(figsize=(20,10))
241         plt.plot(t , y, label='MoCap', color='black',linewidth=4)
242         plt.plot(t , yp, label='Predicted', color='red',linewidth=4)
243         plt.xlabel('Time [s]',fontsize=20)
244         plt.ylabel('PAFP Torque [Nm/kg]',fontsize=20)
245         plt.xlim(0,t[-1]) # consistent scale
246         plt.grid(True)
247         plt.legend(prop={"size":20}, loc='upper left')
248         plt.title(f'Analytical Regression Model: Test Trial {i+1}',
               fontsize=30)
249         plt.xticks(fontsize=20)
250         plt.yticks(fontsize=20)
251         plt.tight_layout()
252         plt.show()
253         # Save
254         filename = f'Analytical -- Test{i+1}.jpg' # f'Analytical -- Test
               {i+1}.png'
255         savepath = os.path.join(PATH, 'Polynomial' , filename)
256         fig.savefig(savepath, bbox_inches='tight')
257
258         T.append(y)
259         P.append(yp)
260
261     # save results
262     filename = os.path.join(PATH, 'Polynomial', 'Optimization_Results.
           txt')
263     print('tau = (p1*x^3 + p2*x^2 + p3^x + p4) + (k/m)*R*i \n\n', file=
           open(filename, "a"))
264     print('Coefficients: ', file=open(filename, "a"))
265     print('\t\n'.join(map(str, best_model)), file=open(filename, "a"))
266     # print('\nRMSE: {:.6f} Nm/kg\n'.format(test_rmse), file=open(
           filename, "a"))
267     # print('PCC: {:.6f}\n'.format(test_pcc), file=open(filename, "a"))
268     print('\nRMSE: {:.6f} ({:.6f}) Nm/kg\n'.format(np.average(RMSE), np.
           std(RMSE)), file=open(filename, "a"))
269     print('% RMSE: {:.6f} ({:.6f}) Nm/kg\n'.format(np.average(RMSE_per),
            np.std(RMSE_per)), file=open(filename, "a"))
270     print('PCC: {:.6f} ({:.6f})\n'.format(np.average(PCC), np.std(PCC)),
            file=open(filename, "a"))
```

```
271
272     # Visualize Histogram
273     err = response - pred
274     meanErr = round(np.mean(err),4)
275     stdErr = round(np.std(err),4)
276     bins = 30
277
278     fig = plt.figure(figsize=(20,10))
279     ax = fig.add_subplot(111)
280     N, bins, patches = ax.hist(err, bins=bins, density=True, histtype='
            bar')
281     fracs = N / N.max()
282     norm = colors.Normalize(fracs.min(), fracs.max())
283     # Now, we'll loop through our objects and set the color of each
            accordingly
284     for thisfrac, thispatch in zip(fracs, patches):
285         color = plt.cm.viridis(norm(thisfrac))
286         thispatch.set_facecolor(color)
287
288     # Now we format the y-axis to display percentage
289     ax.yaxis.set_major_formatter(tick.PercentFormatter())
290
291     plt.xlabel('Error [Nm/kg]',fontsize=20)
292     plt.title(f'Analytical Regression Model Testing Errors: ' + str(
            meanErr) + ' +/- ' + str(stdErr) + ' Nm/kg',fontsize=30)
293     plt.grid(True)
294     plt.xticks(fontsize=20)
295     plt.yticks(fontsize=20)
296     plt.tight_layout()
297     plt.show()
298     # Save
299     filename = f'Analytical -- Error Histogram.jpg' # f'Analytical --
            Error Histogram.png'
300     savepath = os.path.join(PATH, 'Polynomial' , filename)
301     fig.savefig(savepath, bbox_inches='tight')
302
303     # Visualize Full Trials
304     plot_full(ankle_angles_, current_, response_, constants, best_model)
305
306     return best_model, test_rmse, response, pred, T, P
```

Appendix D

SUPPLEMENTARY HARDWARE & WEARABLE SENSORS SPECIFICATIONS

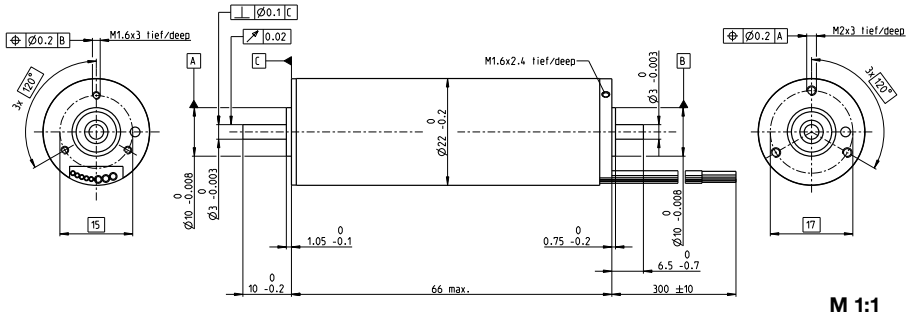
This appendix contains the specification sheets for the motor components, servo controller, wearable sensors, and instrumented treadmill hardware.

D.1 Maxon Motor Datasheet

maxon EC-4pole

EC-4pole 22 ∅22 mm, brushless, 120 Watt

High Power



Stock program
Standard program
Special program (on request)

Part Numbers

	311535	311536	311537	311538
Motor Data				
Values at nominal voltage				
1 Nominal voltage V	18	24	36	48
2 No load speed rpm	16800	16900	17800	16900
3 No load current mA	298	223	166	112
4 Nominal speed rpm	15700	15800	16800	15800
5 Nominal torque (max. continuous torque) mNm	54	54.6	54	54.5
6 Nominal current (max. continuous current) A	5.55	4.21	2.95	2.1
7 Stall torque mNm	874	954	1090	1020
8 Stall current A	86	70.4	56.8	37.7
9 Max. efficiency %	89	89	90	90
Characteristics				
10 Terminal resistance phase to phase Ω	0.209	0.341	0.634	1.27
11 Terminal inductance phase to phase mH	0.017	0.031	0.062	0.123
12 Torque constant mNm/A	10.2	13.5	19.2	27.1
13 Speed constant rpm/V	940	705	497	352
14 Speed/torque gradient rpm/mNm	19.4	17.7	16.4	16.6
15 Mechanical time constant ms	1.81	1.65	1.53	1.54
16 Rotor inertia gcm²	8.91	8.91	8.91	8.91

Specifications

Thermal data

17 Thermal resistance housing-ambient	10.7 K/W
18 Thermal resistance winding-housing	0.7 K/W
19 Thermal time constant winding	4.66 s
20 Thermal time constant motor	936 s
21 Ambient temperature	-20…+100°C
22 Max. winding temperature	+155°C

Mechanical data (preloaded ball bearings)

23 Max. speed		25000 rpm
24 Axial play at axial load	< 3.0 N	0 mm
	> 3.0 N	0.14 mm
25 Radial play		preloaded
26 Max. axial load (dynamic)		4 N
27 Max. force for press fits (static)		53 N
(static, shaft supported)		600 N
28 Max. radial load, 5 mm from flange		16 N

Other specifications

29 Number of pole pairs	2
30 Number of phases	3
31 Weight of motor	175 g

Values listed in the table are nominal.

Connection motor (Cable AWG 20)

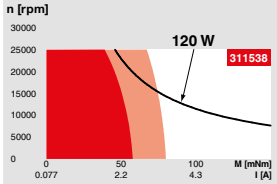
red	Motor winding 1
white	Motor winding 3
black	Motor winding 2

Connection sensors (Cable AWG 26)

red/grey	Hall sensor 1
black/grey	Hall sensor 2
white/grey	Hall sensor 3
green	V_{Hall} 3…24 VDC
blue	GND

Wiring diagram for Hall sensors see p. 45

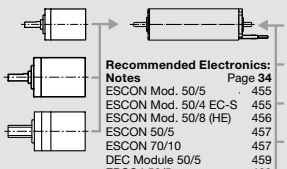
Operating Range Comments



Continuous operation
Continuous operation with reduced thermal resistance R_{th2} 50%
Intermittent operation

Assigned power rating

maxon Modular System Details on catalog page 34



Planetary Gearhead
∅22 mm
2.0 - 3.4 Nm
Page 343

Planetary Gearhead
∅32 mm
1.0 - 6.0 Nm
Page 353

Screw Drive
∅32 mm
Page 382–387

Recommended Electronics:

Notes	Page 34
ESCON Mod. 50/5	455
ESCON Mod. 50/4 EC-S	455
ESCON Mod. 50/8 (HE)	456
ESCON 50/5	457
ESCON 70/10	457
DEC Module 50/5	459
EPOS4 50/5	463
EPOS4 Mod./Comp. 50/5	463
EPOS2 P 24/5	470
MAXPOS 50/5	473

Encoder 16 EASY
128 - 1024 CPT, 3 channels
Page 418

Encoder 16 EASY XT
128 - 1024 CPT, 3 channels
Page 420

Encoder 16 EASY Absolute
4096 steps, Single Turn
Page 422

Encoder 16 EASY Absolute XT
4096 steps, Single Turn
Page 424

Encoder 16 RIO
1024 - 32768 CPT, 3 channels
Page 435

Encoder AEDL 5810
1024 - 5000 CPT, 3 channels
Page 438

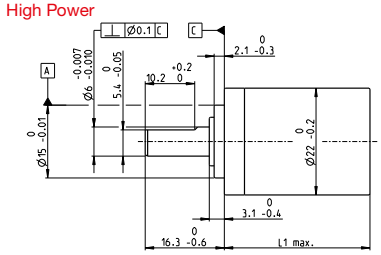
Encoder HEDL 5540
500 CPT, 3 channels
Page 446

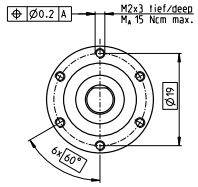
April 2019 edition / subject to change

D.2 Maxon Planetary Gearhead Datasheet

Planetary Gearhead GP 22 HP ∅22 mm, 2.0–3.4 Nm

High Power





M 1:1

Technical Data

Planetary Gearhead	straight teeth
Output shaft	stainless steel, hardened
Bearing at output	ball bearing
Radial play, 10 mm from flange	max. 0.2 mm
Axial play	max. 0.1 mm
Max. axial load (dynamic)	100 N
Max. force for press fits	100 N
Direction of rotation, drive to output	=
Max. continuous input speed	12 000 rpm
Recommended temperature range	-40...+100°C
Number of stages	1 2 3 4
Max. radial load, 10 mm from flange	55 N 85 N 100 N 110 N

maxon gear

Stock program
Standard program
Special program (on request)

Part Numbers	370683	370687	370690	370776	370780	370783	370792	370797	370802	370807
Gearhead Data (provisional)										
1 Reduction	3.8:1	14:1	20:1	53:1	76:1	104:1	198:1	316:1	410:1	590:1
2 Absolute reduction	15/4	225/16	81/4	3375/64	1215/16	87723/845	50625/256	2777895/8788	6561/16	59049/100
3 Max. motor shaft diameter mm	4	4	4	4	4	3.2	4	3.2	4	4
Part Numbers	370685	370688	370691	370778	370781	370784	370794	370799	370803	370808
1 Reduction	4.4:1	16:1	24:1	62:1	84:1	109:1	231:1	333:1	455:1	690:1
2 Absolute reduction	57/13	855/52	1539/65	12825/208	185193/2197	2187/20	192375/832	69255/208	5000211/10985	1121931/1625
3 Max. motor shaft diameter mm	3.2	3.2	3.2	3.2	3.2	4	3.2	3.2	3.2	3.2
Part Numbers	370686	370689	370692	370779	370782	370785	370795	370800	370805	370809
1 Reduction	5.4:1	19:1	29:1	72:1	89:1	128:1	270:1	370:1	479:1	850:1
2 Absolute reduction	27/5	3249/169	729/25	48735/676	4617/52	41553/325	731025/2704	10556001/28561	124659/260	531441/625
3 Max. motor shaft diameter mm	2.5	3.2	2.5	3.2	3.2	3.2	3.2	3.2	3.2	2.5
Part Numbers						370786	370796	370801	370806	
1 Reduction						157:1	285:1	389:1	561:1	
2 Absolute reduction						19663/125	18225/64	263169/676	236852/4225	
3 Max. motor shaft diameter mm						2.5	4	3.2	3.2	
4 Number of stages	1	2	2	3	3	3	4	4	4	4
5 Max. continuous torque Nm	2	2.4	2.4	3	3	3	3.4	3.4	3.4	3.4
6 Max. intermittent torque at gear output Nm	2.5	3	3	3.5	3.5	3.5	3.8	3.8	3.8	3.8
7 Max. efficiency %	84	70	70	59	59	59	49	49	49	49
8 Weight g	51	64	64	78	78	78	91	91	91	91
9 Average backlash no load °	1.0	1.2	1.2	1.6	1.6	1.6	2.0	2.0	2.0	2.0
10 Mass inertia gcm²	0.6	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
11 Gearhead length L1 mm	25.3	32.3	32.3	39.0	39.0	39.0	45.7	45.7	45.7	45.7

maxon Modular System

+ Motor	Page	+ Sensor/Brake	Page	Overall length [mm] = Motor length + gearhead length + (sensor/brake) + assembly parts									
EC-max 22, 12 W	222			57.4	64.4	64.4	71.1	71.1	71.1	77.8	77.8	77.8	77.8
EC-max 22, 12 W	222	MR	418	67.1	74.1	74.1	80.8	80.8	80.8	87.5	87.5	87.5	87.5
EC-max 22, 12 W	222	AB 20	478	93.0	100.0	100.0	106.7	106.7	106.7	113.4	113.4	113.4	113.4
EC-max 22, 25 W	223			73.9	80.9	80.9	87.6	87.6	87.6	94.3	94.3	94.3	94.3
EC-max 22, 25 W	223	MR	418	83.6	90.6	90.6	97.3	97.3	97.3	104.0	104.0	104.0	104.0
EC-max 22, 25 W	223	AB 20	478	109.5	116.5	116.5	123.2	123.2	123.2	129.9	129.9	129.9	129.9
EC-4pole 22, 90 W	231			74.0	81.0	81.0	87.7	87.7	87.7	94.4	94.4	94.4	94.4
EC-4pole 22, 90 W	231	16 EASY/Abs.	409/411	86.2	93.2	93.2	99.9	99.9	99.9	106.6	106.6	106.6	106.6
EC-4pole 22, 90 W	231	AEDL/HEDL	427/433	95.5	102.5	102.5	109.2	109.2	109.2	115.9	115.9	115.9	115.9
EC-4pole 22, 120 W	232			91.4	98.4	98.4	105.1	105.1	105.1	111.8	111.8	111.8	111.8
EC-4pole 22, 120 W	232	16 EASY/Abs.	409/411	103.6	110.6	110.6	117.3	117.3	117.3	124.0	124.0	124.0	124.0
EC-4pole 22, 120 W	232	AEDL/HEDL	427/433	112.9	119.9	119.9	126.6	126.6	126.6	133.3	133.3	133.3	133.3

D.3 Maxon Escon 70/10 Servo Controller Datasheet

maxon motor

maxon motor control	**ESCON Servo Controller**
Hardware Reference	**Edition November 2015**

ESCON 70/10

Servo Controller

P/N 422969

Hardware Reference

escon.maxonmotor.com

Document ID: rel5543

maxon motor ag Brünigstrasse 220 P.O.Box 263 CH-6072 Sachseln Phone +41 41 666 15 00 Fax +41 41 666 16 50 www.maxonmotor.com

2 Specifications

2.1 Technical Data

ESCON 70/10 (422969)		
Electrical Rating	Nominal operating voltage $+V_{CC}$	10…70 VDC
	Absolute operating voltage $+V_{CC\,min}$ / $+V_{CC\,max}$	8 VDC / 76 VDC
	Output voltage (max.)	0.95 x $+V_{CC}$
	Output current I_{cont} / I_{max} (<20 s)	10 A / 30 A
	Pulse Width Modulation frequency	53.6 kHz
	Sampling rate PI current controller	53.6 kHz
	Sampling rate PI speed controller	5.36 kHz
	Max. efficiency	98%
	Max. speed DC motor	limited by max. permissible speed (motor) and max. output voltage (controller)
	Max. speed EC motor	150'000 rpm (1 pole pair)
	Built-in motor choke	3 x 15 µH; 10 A
Inputs & Outputs	Analog Input 1 Analog Input 2	resolution 12-bit; −10…+10 V; differential
	Analog Output 1 Analog Output 2	resolution 12-bit; −4…+4 V; referenced to GND
	Digital Input 1 Digital Input 2	+2.4…+36 VDC (R_i = 38.5 kΩ)
	Digital Input/Output 3 Digital Input/Output 4	+2.4…+36 VDC (R_i = 38.5 kΩ) / max. 36 VDC (I_L <500 mA)
	Hall sensor signals	H1, H2, H3
	Encoder signals	A, A\, B, B\, (max. 1 MHz)
Voltage Outputs	Auxiliary output voltage	+5 VDC (I_L ≤10 mA)
	Hall sensor supply voltage	+5 VDC (I_L ≤30 mA)
	Encoder supply voltage	+5 VDC (I_L ≤70 mA)
Potentiometers	Potentiometer P1 (on board) Potentiometer P2 (on board)	240°; linear
Motor Connections	DC motor	+ Motor, − Motor
	EC motor	Motor winding 1, Motor winding 2, Motor winding 3
Interface	USB 2.0 / USB 3.0	full speed
Status Indicators	Operation	green LED
	Error	red LED

ESCON 70/10 (422969)			
Physical	Weight	approx. 259 g	
	Dimensions (L x W x H)	125 x 78.5 x 27 mm	
	Mounting holes	for M4 screws	
Environmental Conditions	Temperature	Operation	−30…+45 °C
		Extended range *1)	+45…+82 °C Derating ➔Figure 2-1
		Storage	−40…+85 °C
	Altitude *2)	Operation	0…10'000 m MSL
	Humidity	5…90% (condensation not permitted)	

*1) Operation within the extended range (temperature and altitude) is permitted. However, a respective derating (declination of output current I_{cont}) as to the stated values will apply.
*2) Operating altitude in meters above Mean Sea Level, MSL.

Table 2-4 Technical Data

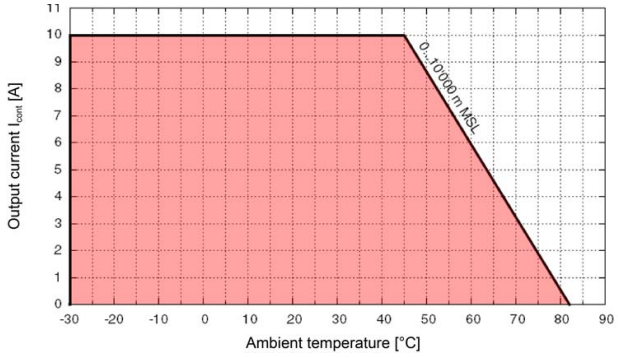


Figure 2-1 Derating Output Current

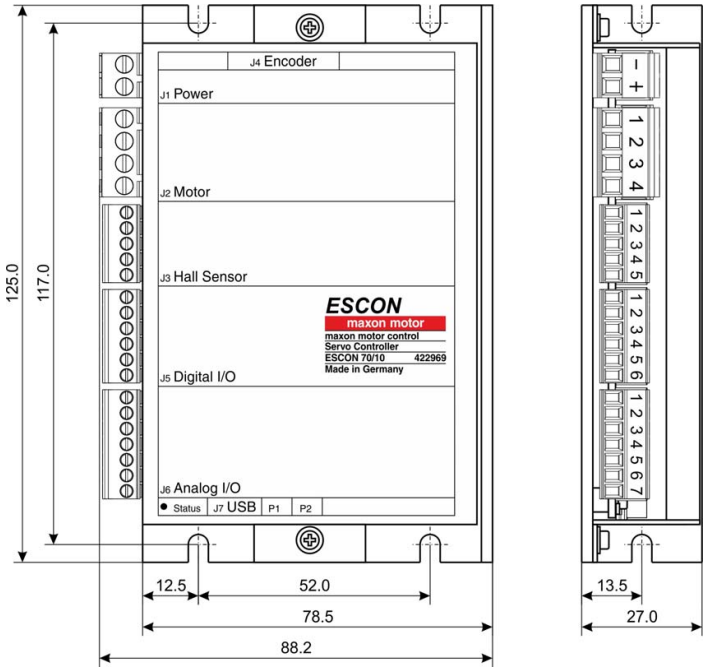


Figure 2-2 Dimensional Drawing [mm]

D.4 CUI AMT11 Encoder Datasheet

Additional Resources: Product Page | 3D Model

CUI DEVICES

date 10/08/2020
page 1 of 9

SERIES: AMT11 | **DESCRIPTION:** MODULAR INCREMENTAL ENCODER

FEATURES
- patented capacitive ASIC technology
- low power consumption
- incremental resolutions up to 4096 PPR
- resolutions programmable with AMT Viewpoint™ PC software
- differential line driver versions
- digitally set zero position
- compact modular package with locking hub for ease of installation
- radial and axial cable connections
- 7 different mounting hole options
- -40~125°C operating temperature

RoHS

ELECTRICAL

parameter	conditions/description	min	typ	max	units
power supply	VDD	4.5	5	5.5	V
start up time			200		ms
current consumption	with unloaded output		16		mA
single ended channels	output high level	VDD-0.1			V
	output low level			0.1	V
	output current (per channel)			15	mA
	rise/fall time		8		ns
differential RS-422 channels	output high level	3			V
	output low level			0.1	V
	output current (per channel)			25	mA
	rise/fall time	7	11	20	ns

INCREMENTAL CHARACTERISTICS

parameter	conditions/description	min	typ	max	units
channels	CMOS Voltage (S) A, B, Z Quadrature Line Driver (Q) A, $\overline{A}$, B, $\overline{B}$, Z, $\overline{Z}$				
waveform	CMOS voltage square wave				
phase difference	A leads B for CCW rotation (viewed from front)				
quadrature resolutions[1]	48, 96, 100, 125, 192, 200, 250, 256, 360, 384, 400, 500, 512, 768, 800, 1000, 1024, 1600, 2000, 2048, 2500, 4096				PPR
index[2]	one pulse per 360 degree rotation				
accuracy			0.2		degrees
quadrature duty cycle (at each resolution)	48, 96, 100, 125, 192, 256, 384	49	50	51	%
	200, 250, 360, 400, 768, 800	48	50	52	%
	500, 1000, 1600	46	50	54	%
	512, 1024 , 2048, 4096	50	50	50	%
	2000	44	50	56	%
	2500	43	50	57	%

Notes:
1. Resolution programmed with AMT Viewpoint™ PC software. Default resolution set to 2048 PPR. All resolutions are listed as pre-quadrature, meaning the final number of counts is PPR x 4.
2. Zero position alignment set with AMT One Touch Zero™ module, AMT Viewpoint™ PC software, or serial commands

MECHANICAL

parameter	conditions/description	min	typ	max	units
motor shaft length		9			mm
weight	weight varies by configuration		15.7		g
axial play				±0.3	mm
rotational speed (at each resolution)	48, 96, 100, 125, 192, 200, 250, 256, 384, 400, 500, 512, 800, 1000, 1024, 2048			8000	RPM
	360, 768, 1600, 2000, 4096			4000	RPM
	2500			2500	RPM

ENVIRONMENTAL

parameter	conditions/description	min	typ	max	units
operating temperature		-40		125	°C
humidity	non-condensing			85	%
vibration	10~500 Hz, 5 minute sweep, 2 hours on each XYZ			5	G
shock	3 pulses, 6 ms, 3 on each XYZ			200	G
RoHS	yes				

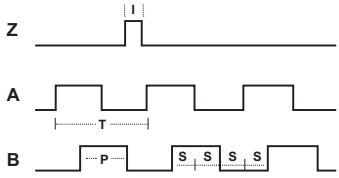
SERIAL INTERFACE

parameter	conditions/description	min	typ	max	units
protocol	serial UART				
controller	driven by onboard Microchip PIC18F25K80. See Microchip documentation for additional details.				
data rate	8 data bits, no parity, 1 stop bit, least significant bit first		115200		baud

WAVEFORMS

Figure 1

Quadrature signals with index showing counter-clockwise rotation



The following parameters are defined by the resolution selected for each encoder. The encoders resolution is listed as Pulses Per Revolution (PPR), which is the number of periods (or high pulses) over the encoders revolution.

Parameter	Description	Expression	Units	Notes
PPR	resolution		Pulses Per Revolution	This is the user selected value and the format all resolutions are listed in
CPR	counts	PPR x 4	Counts Per Revolution	This is the number of quadrature counts the encoder has
T	period	360/R	mechanical degrees	
P	pulse width	T/2	mechanical degrees	
S	A/B state width	T/4	mechanical degrees	This is the width of a quadrature state
I	index width	T/4	mechanical degrees	The width of a once per turn index is the state width for A & B lines

Note: For more information regarding PPR, CPR, or LPR (Lines Per Revolution) view https://www.cuidevices.com/blog/what-is-encoder-ppr-cpr-and-lpr

PART NUMBER KEY

For customers that prefer a specific AMT11 configuration, please reference the custom configuration key below.

AMT11XX - XXXX - XXXX - X

- Base Number
- Orientation:
  2 = Radial
  3 = Axial
- Output Signals:
  S = CMOS Voltage
  Q = Quadrature Line Driver
- Quadrature Resolution (PPR):

0048	0500
0096	0512
0100	0768
0125	0800
0192	1000
0200	1024
0250	1600
0256	2000
0360	2048
0384	2500
0400	4096

- Sleeve Bore Diameter:
  2000 = 2 mm
  3000 = 3 mm
  3175 = 3.175 mm (1/8")
  4000 = 4 mm
  4760 = 4.76 mm (3/16")
  5000 = 5 mm
  6000 = 6 mm
  6350 = 6.35 mm (1/4")
  8000 = 8 mm
- Mounting Base:
  S = Standard
  W= Wide

AMT11-V KITS

In order to provide maximum flexibility for our customers, the AMT11 series is provided in kit form standard. This allows the user to implement the encoder into a range of applications using one sku#, reducing engineering and inventory costs.

ORDERING GUIDE

AMT11XX-V

- Orientation:
  2 = Radial
  3 = Axial
- Output Signals:
  S = CMOS Voltage
  Q = Quadrature Line Driver

SLEEVES								
								
2mm	3mm	1/8 inch (3.175mm)	4mm	3/16 inch (4.76mm)	5mm	6mm	1/4 inch (6.35mm)	8mm
Light Sky Blue	Orange	Purple	Gray	Yellow	Green	Red	Snow	Blue

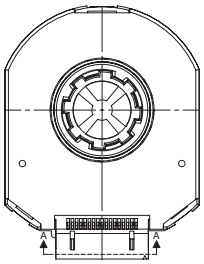
BASE	WIDE BASE	TOP COVER	SHAFT ADAPTER	TOOL A	TOOL C
					

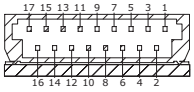
ENCODER INTERFACE

PINOUT CONNECTOR				
	Function			
#	AMT112S	AMT112Q	AMT113S	AMT113Q
1	TX_ENC+	TX_ENC+	RX_ENC+	RX_ENC+
2	RX_ENC+	RX_ENC+	TX_ENC+	TX_ENC+
3	N/A	N/A	N/A	N/A
4	GND[1]	GND[1]	GND[1]	GND[1]
5	N/A	N/A	N/A	N/A
6	+5 V	+5 V	+5 V	+5 V
7	N/A	N/A	N/A	N/A
8	B+	B+	B+	B+
9	N/A	B-	N/A	B-
10	A+	A+	A+	A+
11	N/A	A-	N/A	A-
12	Z+	Z+	Z+	Z+
13	N/A	Z-	N/A	Z-
14	MCLRB	MCLRB	MCLRB	MCLRB
15	N/A	N/A	N/A	N/A
16	N/A	N/A	N/A	N/A
17	N/A	N/A	N/A	N/A

Note: 1. Connect encoder GND to motor chassis as closely as possible. For additional grounding techniques contact CUI Application Support.

AMT112S & AMT112Q

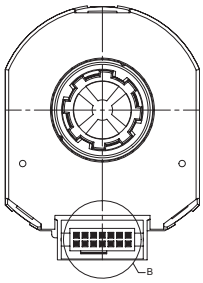


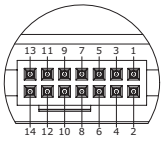


SECTION A-A
SCALE 4 : 1

Mating Connector:
JAE FI-W17S

AMT113S & AMT113Q





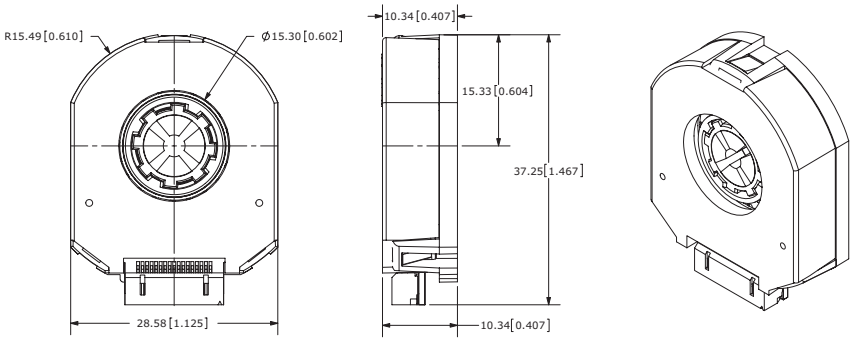
DETAIL B
SCALE 4 : 1

Mating Connector:
Samtec ISDF-07-D-L

MECHANICAL DRAWING

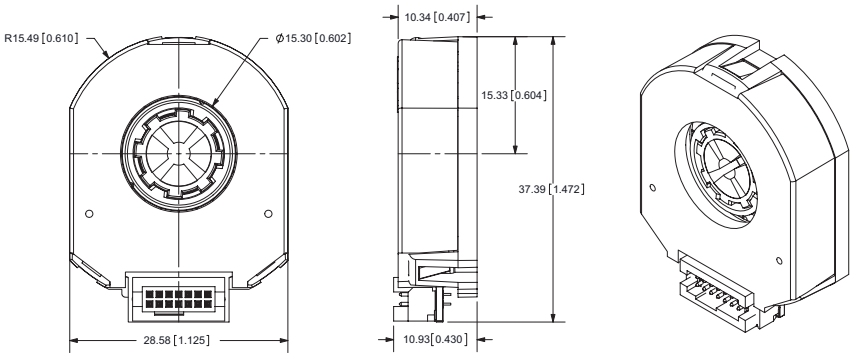
AMT112S & AMT112Q

units: mm[inch]
tolerance: ±0.1



AMT113S & AMT113Q

units: mm[inch]
tolerance: ±0.1

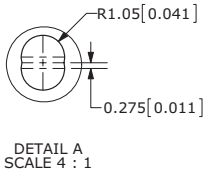


MECHANICAL DRAWING (CONTINUED)

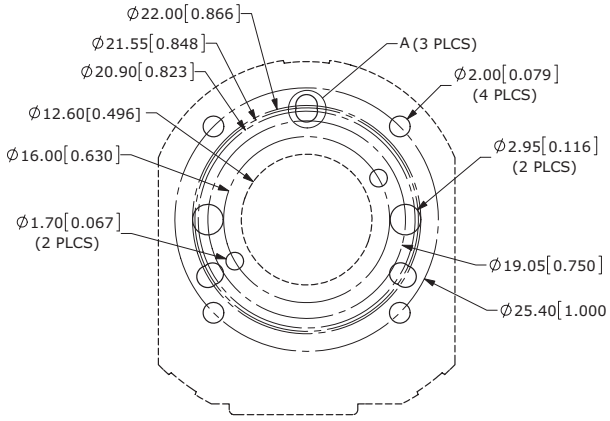
MOUNTING HOLE PATTERNS

STANDARD BASE

units: mm[inch]
tolerance: ±0.1

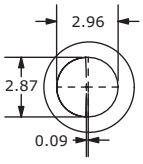


DETAIL A
SCALE 4 : 1

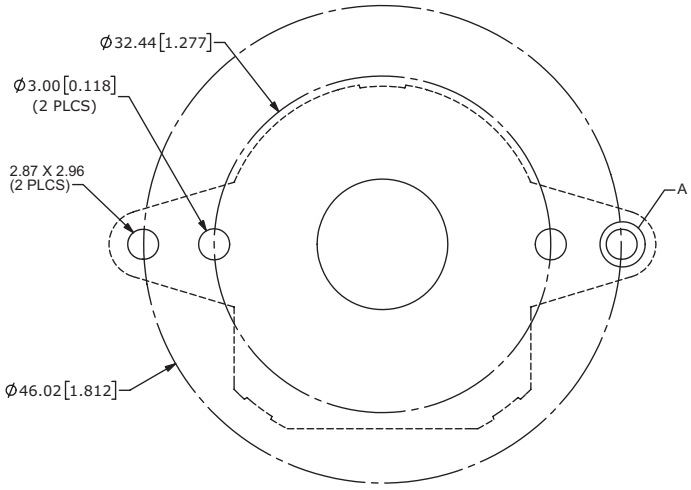


WIDE BASE

units: mm[inch]
tolerance: ±0.1

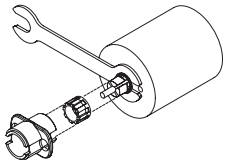


DETAIL A
SCALE 4 : 1



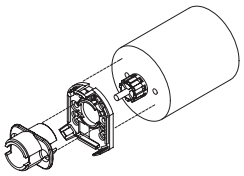
ASSEMBLY PROCEDURE

STEP 1



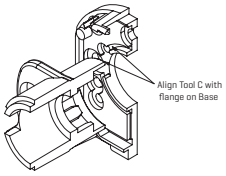
1. Insert Tool A as a spacer that defines the distance to the mounting surface.
2. Slide appropriate sized Sleeve over shaft all the way down to Tool A.
3. Slide Shaft Adaptor over Sleeve.
4. Use Tool C to press Shaft Adaptor over Sleeve (ensure Shaft Adapter and Tool C spline alignment) until flush with Tool A.

STEP 2



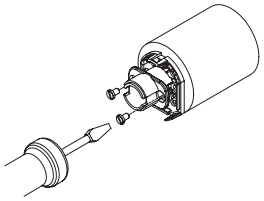
1. Remove Tools A and C.
2. Place Base on motor, with Tool C used as a centering tool.

STEP 3



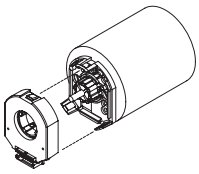
1. Align Tool C with flange on Base.
2. Slide Base and Tool C onto motor, centering onto the Shaft Adapter.

STEP 4



1. Fasten the Base on the motor (Tool C may need to be rotated to allow for some mounting configurations).
2. Remove Tool C.

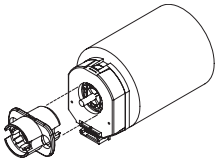
STEP 5



1. Snap the Top Cover onto the Base, carefully observing that the teeth of the Shaft Adaptor align with the grooves in the hub. *

* *We recommend no more than three cycles of mounting and removal of the AMT top cover base. Multiple cycles of mounting and removing the top cover can cause base fatigue over time and affect encoder performance.*

STEP 6



1. Make sure the snaps are fully engaged by pressing on the Hub with the reverse side of Tool C.
2. When assembly is finished, the Shaft Adaptor, Sleeve and Rotor Hub should all be flush with the Motor Shaft rotating freely.

APPLICATION NOTES

SERIAL INTERFACE

The AMT11 series encoder is designed to operate with a serial UART interface. This interface allows the encoder to be configured and programmed by the AMT Viewpoint™ application. Along with programming, the AMT Viewpoint™ application uses the serial interface for diagnostics and index alignment. Below are instructions on how to use the serial interface for position zeroing.

Table 1
Serial Commands

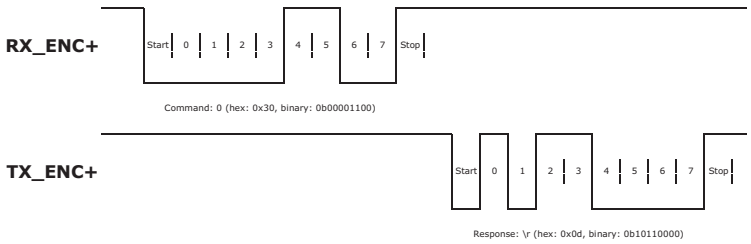
Command	Action	Use
0	This command sends an ascii '0' (hex value 0x30).	This zeros the encoder and sets the index at the current angular position. This position is stored in non-volatile memory and will remain present until a zero command is set again or encoder is reprogrammed via AMT Viewpoint™.
Q	This command sends an ascii 'Q' (hex value 0x51).	This command restarts the encoder as if it were power cycled.

Table 2
Serial Pins

Pin	Description	Connection
TX_ENC+	This is the pin that the encoder transmits serial data on.	Connect this pin to the receiver input of your serial/UART interface.
RX_ENC+	This is the pin that the encoder receives serial commands on.	Connect this pin to your serial/UART interface transmitter output.
MCLRB	This pin is used to force the encoder into reset for reprogramming via the AMT Viewpoint™ application.	Connection of this pin is not required for the above serial commands.

The serial interface operates at 115200 baud with 8 data bits, no parity, and 1 stop bit, and 1 start bit. This is the standard UART protocol. Data lines TX_ENC+ and RX_ENC+ are high when inactive.

Figure 2
Serial Timing Diagram



REVISION HISTORY

rev.	description	date
1.0	initial release	04/30/2014
1.01	updated datasheet	06/24/2014
1.02	updated datasheet	10/13/2015
1.03	added 360 & 2500 PPR resolutions, increased operating temperature to 125°C	12/18/2017
1.04	changed outer mounting holes to be oblong on wide base version	10/10/2018
1.05	brand update	11/21/2019
1.06	updated quadrature duty cycle details	10/08/2020

The revision history provided is for informational purposes only and is believed to be accurate.

D.5 LORD Microstrain 3DM-GX5-15 VRU/IMU

MicroStrain Sensing Product Datasheet

3DM-GX5-VRU

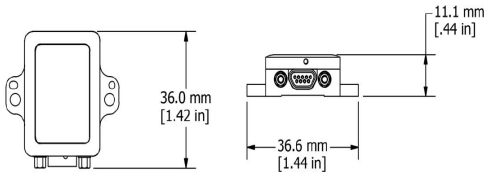
Vertical Reference Unit

The MicroStrain Sensing 3DM-GX5 family of high-performance, industrial-grade inertial sensors provides a wide range of triaxial inertial measurements, computed attitude, and navigation solutions.

In all models, the Inertial Measurement Unit (IMU) includes direct measurement of acceleration and angular rate, and is fully temperature-compensated and calibrated over the operating temperature. The use of Micro-Electro-Mechanical System (MEMS) technology allows for highly accurate, small, lightweight devices.

SensorConnect software is a user friendly program for device configuration. MIP Monitor (MicroStrain Internet Protocol) can also be used. Both packages provide for device configuration, live data monitoring, and recording. Alternatively, the MIP Data Communications Protocol is available for development of custom interfaces and easy OEM integration.

The sensor operates independent of computer platform, operating system, or coding language.



PRODUCT HIGHLIGHTS

- Triaxial accelerometer, gyroscope, magnetometer, temperature sensors achieve the optimal combination of measurement qualities
- Dual on-board processors run a new Auto-Adaptive Extended Kalman Filter (EKF) for outstanding dynamic attitude estimates
- Smallest, lightest, highest performance VR in its class

FEATURES AND BENEFITS

BEST IN CLASS PERFORMANCE

- Fully calibrated, temperature-compensated, and mathematically-aligned to an orthogonal coordinate system for highly accurate outputs
- Bias tracking, error estimation, threshold flags, and adaptive noise modeling allow for fine tuning to conditions in each application
- High-performance, low-drift gyros with low noise density and Vibrational Rectification Error
- Accelerometer noise as low as 20 ug/√Hz

EASE OF USE

- SensorConnect enables simple device configuration, live data monitoring, and recording
- The MSCL API allows easy integration with C++, Python, .NET, C#, Visual Basic, LabVIEW and MATLAB environments.Robust, forward compatible MIP packet protocol
- MIP open byte level communication protocol
- User-defined sensor-to-vehicle frame transformation

COST EFFECTIVE

- Out-of-the box solution reduces development time
- Volume discounts

APPLICATIONS

- Unmanned vehicle navigation
- Robotics
- Platform stabilization, artificial horizon
- Health and usage monitoring of vehicles

Parker LORD

ENGINEERING YOUR SUCCESS.

Vertical Reference Unit (VRU)

Specifications

General		
Integrated Sensors	Triaxial accelerometer, triaxial gyroscope, pressure altimeter, and temperature sensors	
Data Outputs	**Inertial Measurement Unit (IMU) outputs:** acceleration, angular rate, ambient pressure, Delta-theta, Delta-velocity **COMPUTED OUTPUTS** **Extended Kalman Filter (EKF):** filter status, attitude estimates (in Euler angles, quaternion,orientation matrix), bias compensated angular rate, pressure altitude, gravity-free linear acceleration, attitude uncertainties, gyroscope and accelerometer bias, scale factors and uncertainties, gravity models, and more. **Complementary Filter (CF):** attitude estimates (in Euler angles, quaternion, orientation matrix) stabilized, north and up vectors, GPS correlation timestamp	
Inertial Measurement Unit (IMU) Sensor Outputs		
	Accelerometer	**Gyroscope**
Measurement range	±8 g (standard) ±2 g, ±4 g, ±20 g, ±40 g (optional)	300°/sec (standard) ±75, ±150, ±900 (optional)
Non-linearity	±0.02 % fs	±0.02% fs
Resolution	0.02 mg (+/- 8 g)	<0.003°/sec (300 dps)
Bias instability	±0.04 mg	8°/hr
Initial bias error	±0.002 g	±0.04°/sec
Scale factor stability	0.03%	±0.05%
Noise density	20 µg/√Hz (2 g)	0.005°/sec/√Hz (300°/sec)
Alignment error	±0.05°	±0.05°
Bandwidth	225 Hz	250 Hz
Offset error over temperature	0.06% (typ)	0.04% (typ)
Gain error over temperature	0.03% (typ)	0.03% (typ)
Vibration induced noise	--	0.072°/s RMS/g RMS
Vibration rectification error (VRE)	--	0.001°/s/g² RMS
IMU filtering	Digital sigma-delta ADC sampled at 1kHz and 4kHz. 4kHz data averaged to 1kHz nominal sampling rate. Scaled into physical units at 1kHz. User adjustable IIR filter available for 1kHz data. Coning and sculling integrals computed at 1kHz.	
Sampling rate	1 kHz	4 kHz
IMU data output rate	1 Hz to 1000 Hz	
Pressure Altimeter		
Altitude Range	1260-260 mB (hPa) (-500 to 10,000m)	
Resolution	0.01 hPa RMS	
Relative Accuracy	±0.1 mB, over the range 800-1000mB @ T=25°C	
Sampling rate	25 Hz	

Computed Outputs	
Attitude accuracy	EKF outputs: ±0.25° RMS roll and pitch, (typ) CF outputs: ±0.5° roll and pitch (static, typ) and ±2.0° roll and pitch (dynamic, typ)
Attitude heading range	360° about all axes
Attitude resolution	< 0.01°
Attitude repeatability	0.2° (typ)
Calculation update rate	500 Hz
Computed data output rate	EKF outputs: 1 Hz to 500 Hz CF outputs: 1 Hz to 1000 Hz
Operating Parameters	
Communication	USB 2.0 (full speed) RS232 (9,600 bps to 921,600 bps, default 115,200)
Power source	+4 to + 36 V dc
Power consumption	500 mW (typ)
Operating temperature	-40°C to +85°C
Mechanical shock limit	500*g*/1ms absolute maximum survivability.*
MTBF	557,280 hours (Telcordia method, GM/35C)
Physical Specifications	
Dimensions	36.0 mm x 36.6 mm x 11.1 mm
Weight	16.5 grams
Enclosure material	Aluminum
Regulatory compliance	CE, REACH, ROHS
Integration	
Connectors	Data/power: Micro-D9
Software	SensorConnect and MIP Monitor software included; Windows XP/Vista/7/8/10 compatible
Data Communications Protocol (DCP)	Protocol compatibility across GX3, GX4, RQ1, GQ4, GX5 CX5 and CV5 product families
Software development kit (SDK)	MicroStrain Communication Library (MSCL) open source license includes full documentation and sample code.)

*Prolonged exposure to >2x full scale range can result in permanent damage. See manual for details

Parker Hannifin Corporation
MicroStrain Sensing
459 Hurricane Lane
Williston, VT 05495 · USA

phone: +1.802.862.6629
email: sensing_sales@LORD.com
sensing_support@LORD.com
www.microstrain.com
www.parker.com

D.6 AMTI Instrumented Force-Sensing Split-Belt Treadmill

AMTI
FORCE AND MOTION

www.AMTI.biz | sales@amtimail.com

AMTI FORCE-SENSING SIDE-BY-SIDE TREADMILL

- *Print-friendly version*
- **Request a Quote**

AMTI's side-by-side split-belt, force-sensing treadmill features two belts separated longitudinally by a 7mm gap. Each belt is 12.5 inches wide and can be run at a speed that is independent of the other. This configuration is ideal for certain pathological gait studies and treatment protocols such as in stroke rehabilitation.

A six-axis AMTI force plate is located under each belt to record the ground reaction forces (Fx, Fy, Fz) and moments (Mx, My, MZ) generated by the subject. The force plates are an integral part of the treadmill design and each force plate entirely supports its own treadmill/drive motor assembly.

High performance digital motor controllers and included software provide users with an intuitive interface for control of treadmill inclination, belt travel direction and belt speed up to 20 km/h. The extensive capabilities of the Instrumented Treadmill make it an ideal method for studying gait under level, uphill and downhill conditions.

The treadmill may be mounted on top of the laboratory floor and used with a set of attached stairs, or it may be mounted into a recessed pit.

KEY FEATURES

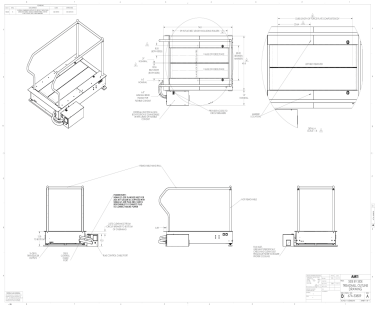
- *Side-by-side belt design allows independent kinetic data for each leg during the "double support" phase of walking*
- *Two integral composite force plates included (one under each belt)*
- *Speed: Two belts 0-20 km/h, independently adjustable in .06 km/h increments*
- *Inclination up to 25% grade*
- *Reversible belt direction for uphill and downhill walking and running*
- *Analog and digital outputs from each force plate supplied by included GEN5 amplifiers*
- *Removable side and front handrails*

Integrated Force Plate Specifications	
Vertical Force Plate Capacity	*8800 N*
Horizontal Force Plate Capacity	*4500 N*
Linearity	*+/- 0.2 % full scale output*
Hysteresis	*+/- 0.2 % full scale output*
Treadmill Dimensions	
Working Surface of Each Belt	*152.4 (L) X 31.75 (W) cm*
Total Working Surface	*152.4 (L) X 63.5 (W) cm*

Non-Inclined Height	*28 cm*
Overall Dimensions (Including Handrails)	*203 (L) X 112 (W) X 125 (H) cm*
Front Handrail Supports (Fixed)	*Two posts, 91 cm high and 91 cm apart*
Weight	*400 kg*
Power Requirements	
208 VAC, 3-phase, WYE connected 20-Amp twist lock receptacle.	

TECHNICAL DRAWINGS

Footprint Drawing (click on image to enlarge)



D.7 AMTI Gen 5 Force Plate Amplifier

AMTI introduces the next generation in strain gage amplifiers...

Gen 5: Advanced six-channel signal conditioner

The Gen 5 provides industry-leading performance and innovative features in an easy-to-use and cost effective package.

State-of-the-art signal conditioning

- Provides six unique, independently configurable data channels
- Fully calibrated and NIST-traceable
- Signal conditioning includes: 1 kHz anti-aliasing filter, oversampling and digital signal processing
- Built in cable length compensation
- Virtually eliminates crosstalk by applying a factory calibrated platform correction matrix
- Tested to medical-grade standards for safety, essential performance and electro-magnetic compatibility

Heart of new Smart Platform System

- Applies complete transducer profile stored onboard all new AMTI force plates and force sensors, including bridge resistance, transducer capacity and calibration matrix
- Allows users to order their platforms by simply walking across them in subjects' direction of travel

Intuitive and easy to use

- Fully software configurable
- All calibration and configuration settings stored internally
- Multiple Gen 5's automatically synchronize data sampling – no additional wiring required.
- Automatic balancing of strain gage bridges initiated by front-panel button or by software
- Supports both genlock data acquisition and external triggering

Digital and analog outputs

- Gen 5 can be connected to a PC or other data acquisition system via the digital (USB) or analog output
- Factory calibrated gain and offset correction constants applied to each output channel
- Each of the six analog output channels has an independent 16-bit DAC conditioned by a low-pass reconstruction filter and amplifier.
- Digital output stream consists of fully processed IEEE floating point numbers presented in their respective engineering units.

Highly configurable

- User-selectable excitation (2.5, 5.0, and 10.0 volts)
- User-selectable gains (500, 1000, 2000, and 4000)
- User-selectable zero set point

Flexible integration

- NetForce: complete data acquisition solution – features multi-channel real time acquisition and display, including center of pressure
- Gen 5 Setup Program: easy integration with analog systems and third-party software applications
- Gen 5 Software Development Kit: available for incorporating Gen 5 amplifiers into proprietary systems

AMTI
FORCE AND MOTION

www.AMTI.biz

Gen 5 specifications

Analog inputs	Six 4-arm strain gage bridges (350 Ohm minimum)
Bridge excitation	Channel independent, software configurable – 2.5, 5 or 10 VDC
Amplifier gains	Channel independent, software configurable – 500, 1000, 2000, 4000
Auto zero	Push button or software initiated
Anti-aliasing filter	1000 Hz low pass, 2-pole Butterworth
Analog output range	+/- 5 volts
Analog output reconstruction filter	1000 Hz low pass, 3-pole Butterworth
Analog output DAC	16 bit
Sample rate	Max: 2000 Hz/channel Min: 10 Hz/ channel
Synchronization	Genlock, external trigger, internal clock
Digital Signal Processor	16 bit
Digital data	IEEE 754 floating point
Digital resolution	14 bit
Power supply	External medical grade (included) Input: 120-240 VAC, 50/60 Hz Output: 15VDC @ 0.3 amps
Connectors	Digital output: USB 2.0 Sync/genlock: RCA phono Power: 5.5 mm x 2.1 mm plug Analog output: DB25S Transducer Input: 26-pin high density D type
System environmental operating conditions	0 to 125˚F (-18 to 52˚C) 0 to 70% RH, indoor/laboratory environment
Physical dimensions (WxLxH)	26 x 21 x 4 cm (10.25 x 8.25 x 1.72)
Weight	2 kg (4.5 lbs)

ISO 13485 certified

The Gen 5 is a medical-grade strain gage signal conditioner manufactured under the ISO 13485:2003 quality system.

CE

The amplifier meets applicable CE standards and has successfully demonstrated compliance to standards for medical electrical equipment regarding basic safety, essential performance and electro-magnetic compatibility:

AAMI – ES 60601-1

CSA – EN 60601-1

UL – IEC –EN 60601-1

AMTI
FORCE AND MOTION

www.AMTI.biz

GEN5-022514

Appendix E

INSTITUTIONAL REVIEW BOARD CONSENT FORM

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

LAY TITLE: Motion Analysis Lab Project Development

Researcher:

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

For study related questions call: (206) 764-2962

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

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

2. Description of the study including procedures to be used: The procedures in which you will be asked to participate are marked below with an "X." Depending on the number of tests you participate in, the total time will vary. You will not spend more than 4 hours in the lab at any one time. If you agree to wear the step-count device, you may wear the device for up to a month. All procedures are for research purposes and do not provide clinical treatment.

☐ **Novel Prosthetic or Orthotic Device.** You will be asked to wear one or several different prosthetic and/or orthotic device(s). You will be asked to wear each device until you feel confident and stable enough to perform the procedures checked below. If you are not able to obtain confidence and stability on any of the devices, we will stop the session.

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

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

☐ **Body Measurements.** We will take a standard set of body measurements (such as your height, weight, leg length, foot length, and other similar measurements) using calipers, a measuring tape, and a standard scale. The measurements will help us fit you with the study devices and equipment we use to collect data.

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

Check each one that applies:

_____ stand & balance
_____ walk
_____ run
_____ jump

_____ barefoot
_____ in shoes

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

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

Check each one that applies:

_____ walk
_____ run
_____ jump

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

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

Check each one that applies:

_____ walk
_____ run
_____ jump

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

☐ **Motion Measurement Using Optical Motion Detection System Procedure.** First, you will be asked to change into the provided tight-fitting shorts and T-shirt. We will then attach small reflective markers to your body using double-sided tape and ask you to:

Check each one that applies:

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

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

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

Body segments (one box per segment):

_____

_____

_____

_____

_____

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

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

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

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

Muscles and number of electrodes per muscle:

_____

_____

_____

_____

_____

_____

_____

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

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

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

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

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

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

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

The entire procedure will take up to 90 minutes.

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

☐ **Video and/or Photographic Recording Procedures.** During the procedures, we will take photographs and/or video recordings of you for procedure documentation, data analysis, and potential use in research publications and presentations. These images may include your entire body, but we will create an anonymized copy of them during data processing. Your face will be blurred out, and tattoos and other distinguishing marks will either be covered beforehand or blurred out to protect your identity. The original file will be securely stored or deleted. Only videos and photos that cannot identify you would be used in research publications and presentations.

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

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

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

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

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

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

to do this up to eight times with your feet in a set position. To do this, we will ask you to walk to a position on the floor where you will stand and place your feet in a specific position (for example, shoulder width apart or feet together).

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

☐ **Four-Square Step Test (FSST).** While wearing your prescribed prosthetic or orthotic device(s), and/or other novel devices as checked in this form, you may be asked to perform the Four-Square Step Test (FSST). The FSST involves stepping over 4 walking canes that are laid on the floor in a "t" shape to create 4 squares. You will be asked to step from one square to the others in a particular sequence. This will require stepping over each of the canes in forward, backward, and sideways directions. You will step over the canes several times during each test and may be asked to repeat the test multiple times. Before the test, we will ask you if you feel confident stepping over the canes, and we may ask you to demonstrate that you can lift your feet high enough to do the test. It takes about 2 minutes to do this test each time.

☐ **2- or 6- Minute Walk Test.** While wearing your prescribed prosthetic or orthotic device(s), and/or other novel devices as checked in this form, you may be asked to do a 2-minute or 6-minute walk test. You will be asked to walk back and forth in a hallway and "cover as much ground as possible over either 2 or 6 minutes. Walk continuously if possible, but do not be concerned if you need to slow down or stop to rest. The goal is to feel at the end of the test that more ground could not have been covered in the 2 or 6 minutes."

You may be asked to respond to the following question after completing your Walk Test: "On a scale from 6 to 20, with 6 being that you feel that you have not exerted yourself at all, and 20 being that you just worked as hard as you can possibly imagine, how intensely do you feel you have exerted yourself during the last 2 (or 6) minutes?"

☐ **Timed Up and Go.** While wearing your prescribed prosthetic or orthotic device(s), and/or other novel devices as checked in this form, you may be asked to perform the Timed Up and Go. For the Timed Up and Go, you will start seated in a standard armchair, with you back against the chair and your arms resting on the chair's arms. You will be asked to stand up and walk to a line that is 3 meters (9.8 feet) away, turn around at the line, walk back to the chair, and sit down. The test ends when you are seated again. You may use any assistive device (e.g. cane) that you normally use for walking. You will be asked to use a comfortable and safe walking speed. This test takes about 2 minutes each time. You may be asked to do this test a few times.

☐ **Pain Question.** You may be asked to respond to the following question before, during, and/or after completing any of the activities checked in this form: "On a scale from 0 to 10, with 0 being no pain at all, and 10 being the worst pain that you can imagine, please indicate the intensity of pain you are currently experiencing in your right/left foot or ankle". We may ask this question about other joints in your lower limbs.

☐ **Physical Therapy:** Under the guidance of a licensed physical therapist, you may be asked to do a physical therapy session lasting up to 2 hours. You will be able to take breaks as needed. Using standard physical therapy protocols and clinical judgement, the physical therapist will tailor specific tasks for you to do. Overall, the session will focus on increasing strength and range of motion in areas of your body that the physical therapist believes will improve your ability to use your prosthesis/orthosis. The session may also involve activities intended to improve your balance and/or the quality of your posture and gait. The physical therapist will monitor you closely so you do not over-exert yourself during the session. This may include assessing your blood pressure, heart rate, and/or oxygen consumption. You may be asked to do some of the activities checked in this form before, during, and/or after physical therapy.

☐ **User Experience Questions:** We will ask you open-ended questions about your experience using prosthetic, orthotic, or walking aid devices. For example, we may ask you what you like and do not like about a device, about how and when you use a device, about the comfort of a device, or similar questions. We may note your answers for our study records.

☐ **Foot and Ankle Ability Measure:** We will ask you to complete a questionnaire that asks you to rate how your foot and ankle impacts your ability to do a variety of different activities. We may ask you to do this questionnaire several times.

☐ **20-meter Shuttle Run:** We will take you to a hallway that is flat and free of obstacles. We will ask you to move to a marker that is 10 meters (about 33 feet) away and then return to the starting point as quickly as you can. We will record the amount of time it takes you to complete the activity. We may ask you to do this test up to three times per study visit. You can rest for 5minutes between each test.

☐ **Goal Setting-and Achievment Measurements:** We will ask you to tell us about some mobility goals that are important to you. For example, being able to walk with less ankle pain than you currently have or feel more confident walking on stairs. We will discuss the goals with you, and together we will refine them so they are specific, measureable, and achieiveable.

We will ask you to do your goal activity, such as measuring how long it takes you to walk 200 yards. Then we may ask you to do an intervention, like a physical therapy session and/or wear a joint brace as described in this consent form, and we will ask you to do your goal activity again. We may also ask you to do other tests/questionnaires checked in this consent form, before and after you do the intervention activity and/or wear a device.

☐ We will ask you to come back to the VA within 2 weeks to measure your goal activity again and repeat the other tests/questionnaires.

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

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

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

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

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

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

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

Residual Limb Volume and Shape Scanning. You may feel unsteady while standing still.

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

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

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

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

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

Physical Therapy. You may experience mild muscle soreness or fatigue during or shortly after the physical therapy session.

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

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

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

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

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

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

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

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

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

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

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

Photos and videos will be labeled with your study code number. The camera used for photos and videos, and the recording media (e.g., SD cards, optical disks) will be stored in a locked office at the VA Puget Sound. Any images that could identify you will be deleted or securely stored at the VA. The motion capture video files will contain your image but it will look like a stick figure. These files will also be coded. Photos and videos that do not contain identifiable information may also be stored on password-protected computers for future use in scientific presentations and publications.

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

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

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

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

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

You will be paid $20 per hour for participating in these experiments. You will receive payment by check. Checks will be mailed to you about 6-8 weeks after each visit or you can pick them up at VA Puget Sound in Seattle within the same timeframe. In order for these payments to be processed, you will be asked to give your full name, social security number, telephone number, and address. You may receive an Internal Revenue Service (IRS) Form 1099.

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

If you decide to withdraw, or if you are terminated from the study, a person from the study team will then need to meet with you to discuss the necessary steps that you may need to take to end your participation in the study.

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

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

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

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

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

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

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

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

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

Department of Veterans Affairs

VA PUGET SOUND HEALTH CARE SYSTEM (663)
RESEARCH CONSENT FORM

STUDY TITLE: Ambulation Research Protocol Development

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

I agree to participate in this research study as you have explained it in this document.

_____ _____
Subject Signature **Date**

_____
Print Name of Subject

_____ _____
Signature of Person Obtaining Consent **Date**

_____
Print Name of Person Obtaining Consent

Appendix F

HUMAN SUBJECT EXPERIMENTAL RESULTS: STATISTICAL OUTCOME TABLES

This appendix provides statistical information related to the human subject experiment results. A description of the statistical methods can be found within Section 5.2.5. Statistical results are discussed in Section 5.3. Statistical information is separated into two tables, one containing peak biomechanical and temporal outcomes (Table F.1), and the other containing asymmetry measurement outcomes (Table F.2). Each table contains mean outcome values across subjects for each condition. In addition, tables contain mean outcome differences between conditions across subjects.

F.1 Peak and Temporal Outcomes

Table F.1: Mean peak and temporal outcomes and mean pairwise differences by condition from linear regression of outcome on condition by foot interaction with study participant ID as a covariate

	Mean ± SE			F-test p*		Pairwise Differences: Mean ± SE, 95% CI, p-value		
Foot	**Prescribed**	**Passive**	**Active**	**raw**	**BH**	**Prescribed - Passive**	**Prescribed - Active**	**Passive - Active**
Peak Knee Abduction Moment								
Intact	0.422 ± 0.007	0.477 ± 0.007	0.415 ± 0.007	<1e-04	<1e-04	-0.055 ± 0.01, (-0.078, -0.031), <1e-04	0.007 ± 0.01, (-0.016, 0.03), 0.77	0.062 ± 0.01, (0.038, 0.085), <1e-04
Prosthesis	0.270 ± 0.007	0.255 ± 0.007	0.239 ± 0.007	0.0071	0.011	0.015 ± 0.01, (-0.008, 0.038), 0.27	0.031 ± 0.01, (0.008, 0.054), 0.0048	0.016 ± 0.01, (-0.008, 0.039), 0.25
Peak Hip Abduction Moment								
Intact	0.623 ± 0.008	0.537 ± 0.008	0.448 ± 0.008	<1e-04	<1e-04	0.086 ± 0.012, (0.059, 0.113), <1e-04	0.175 ± 0.012, (0.147, 0.202), <1e-04	0.088 ± 0.012, (0.061, 0.116), <1e-04
Prosthesis	0.727 ± 0.008	0.610 ± 0.008	0.644 ± 0.008	<1e-04	<1e-04	0.117 ± 0.011, (0.09, 0.144), <1e-04	0.083 ± 0.011, (0.056, 0.11), <1e-04	-0.034 ± 0.012, (-0.062, -0.007), 0.0093
Peak Ankle Plantarflexion								
Intact	9.5 ± 0.4	8.7 ± 0.4	10.1 ± 0.4	0.034	0.041	0.8 ± 0.6, (-0.5, 2.1), 0.29	-0.6 ± 0.6, (-2.0, 0.7), 0.47	-1.5 ± 0.6, (-2.8, -0.1), 0.026
Prosthesis	9.1 ± 0.4	6.1 ± 0.4	6.0 ± 0.4	<1e-04	<1e-04	3.0 ± 0.6, (1.7, 4.3), <1e-04	3.1 ± 0.6, (1.8, 4.4), <1e-04	0.1 ± 0.6, (-1.2, 1.4), 0.99
Peak Ankle Dorsiflexion								
Intact	14.5 ± 0.3	14.3 ± 0.4	13.3 ± 0.4	0.031	0.039	0.2 ± 0.5, (-0.9, 1.3), 0.91	1.2 ± 0.5, (0.1, 2.4), 0.035	1.0 ± 0.5, (-0.2, 2.2), 0.10
Prosthesis	8.6 ± 0.3	18.5 ± 0.3	14.7 ± 0.3	<1e-04	<1e-04	-10 ± 0.5, (-11.1, -8.8), <1e-04	-6.2 ± 0.5, (-7.3, -5), <1e-04	3.8 ± 0.5, (2.7, 5), <1e-04
Peak Knee Flexion								
Intact	48.9 ± 0.4	50.3 ± 0.4	51.2 ± 0.4	<1e-04	<1e-04	-1.4 ± 0.5, (-2.6, -0.2), 0.016	-2.3 ± 0.5, (-3.5, -1.1), <1e-04	-0.9 ± 0.5, (-2.1, 0.3), 0.21
Prosthesis	37.2 ± 0.4	44.2 ± 0.4	47.2 ± 0.4	<1e-04	<1e-04	-7.0 ± 0.5, (-8.2, -5.8), <1e-04	-10.0 ± 0.5, (-11.2, -8.8), <1e-04	-3.0 ± 0.5, (-4.2, -1.8), <1e-04
Peak Hip Extension								
Intact	7.8 ± 0.2	8.8 ± 0.2	7.7 ± 0.2	0.0024	0.0037	-1.0 ± 0.3, (-1.8, -0.2), 0.012	0.1 ± 0.3, (-0.7, 0.9), 0.91	1.1 ± 0.3, (0.3, 1.9), 0.0042
Prosthesis	10.6 ± 0.2	6.0 ± 0.2	5.5 ± 0.2	<1e-04	<1e-04	4.6 ± 0.3, (3.8, 5.4), <1e-04	5.1 ± 0.3, (4.3, 5.9), <1e-04	0.5 ± 0.3, (-0.3, 1.3), 0.31
Peak Ankle Moment								

Continued on next page

	Mean ± SE			F-test p*		Pairwise Differences: Mean ± SE, 95% CI, p-value		
Foot	**Prescribed**	**Passive**	**Active**	**raw**	**BH**	**Prescribed - Passive**	**Prescribed - Active**	**Passive - Active**
Intact	1.28 ± 0.02	1.35 ± 0.02	1.32 ± 0.02	0.027	0.036	-0.07 ± 0.03, (-0.14, -0.01), 0.021	-0.04 ± 0.03, (-0.11, 0.02), 0.23	0.03 ± 0.03, (-0.04, 0.09), 0.57
Prosthesis	1.55 ± 0.02	0.93 ± 0.02	1.00 ± 0.02	<1e-04	<1e-04	0.61 ± 0.03, (0.55, 0.68), <1e-04	0.54 ± 0.03, (0.48, 0.60), <1e-04	-0.07 ± 0.03, (-0.14, -0.01), 0.024
Peak Knee Flexion Moment								
Intact	0.296 ± 0.007	0.312 ± 0.007	0.326 ± 0.007	0.0084	0.012	-0.016 ± 0.01, (-0.039, 0.007), 0.23	-0.03 ± 0.01, (-0.053, -0.007), 0.0058	-0.014 ± 0.01, (-0.038, 0.009), 0.32
Prosthesis	0.083 ± 0.007	0.074 ± 0.007	0.076 ± 0.007	0.62	0.64	0.009 ± 0.01, (-0.014, 0.032), 0.63	0.007 ± 0.01, (-0.016, 0.03), 0.75	-0.002 ± 0.01, (-0.025, 0.021), 0.98
Peak Knee Extension Moment								
Intact	0.215 ± 0.011	0.316 ± 0.012	0.222 ± 0.012	<1e-04	<1e-04	-0.101 ± 0.017, (-0.14, -0.062), 1e-08	-0.007 ± 0.017, (-0.046, 0.032), 0.91	0.094 ± 0.017, (0.054, 0.134), <1e-04
Prosthesis	0.279 ± 0.011	0.214 ± 0.012	0.270 ± 0.012	2.00e-04	3.00e-04	0.065 ± 0.017, (0.026, 0.104), 3e-04	0.008 ± 0.017, (-0.031, 0.047), 0.87	-0.057 ± 0.017, (-0.096, -0.017), 0.0024
Peak Hip Flexion Moment								
Intact	0.513 ± 0.016	0.599 ± 0.017	0.492 ± 0.017	<1e-04	<1e-04	-0.086 ± 0.023, (-0.14, -0.032), 7e-04	0.022 ± 0.023, (-0.032, 0.076), 0.61	0.107 ± 0.023, (0.052, 0.162), <1e-04
Prosthesis	0.753 ± 0.016	0.711 ± 0.016	0.711 ± 0.016	0.098	0.11	0.042 ± 0.023, (-0.011, 0.096), 0.15	0.042 ± 0.023, (-0.011, 0.096), 0.15	0.000 ± 0.023, (-0.055, 0.054), 1.0
Peak Hip Extension Moment								
Intact	0.423 ± 0.016	0.447 ± 0.017	0.422 ± 0.017	0.49	0.52	-0.024 ± 0.024, (-0.079, 0.032), 0.57	0.002 ± 0.024, (-0.054, 0.057), 1.0	0.025 ± 0.024, (-0.031, 0.082), 0.54
Prosthesis	0.419 ± 0.016	0.407 ± 0.017	0.446 ± 0.017	0.26	0.28	0.011 ± 0.023, (-0.044, 0.066), 0.88	-0.027 ± 0.023, (-0.082, 0.028), 0.48	-0.038 ± 0.024, (-0.094, 0.018), 0.24
Peak GRF								
Intact	1.019 ± 0.007	1.171 ± 0.008	1.097 ± 0.008	<1e-04	<1e-04	-0.153 ± 0.010, (-0.178, -0.128), <1e-04	-0.078 ± 0.010, (-0.103, -0.053), <1e-04	0.075 ± 0.011, (0.05, 0.1), <1e-04
Prosthesis	1.029 ± 0.007	1.056 ± 0.007	1.045 ± 0.007	0.03	0.039	-0.028 ± 0.010, (-0.052, -0.003), 0.024	-0.016 ± 0.010, (-0.041, 0.008), 0.26	0.011 ± 0.011, (-0.014, 0.036), 0.54
Peak Positive Ankle Power								
Intact	1.38 ± 0.03	1.64 ± 0.03	1.61 ± 0.03	<1e-04	<1e-04	-0.26 ± 0.04, (-0.36, -0.16), <1e-04	-0.23 ± 0.04, (-0.33, -0.13), <1e-04	0.03 ± 0.04, (-0.07, 0.13), 0.78
Prosthesis	1.34 ± 0.03	0.47 ± 0.03	0.99 ± 0.03	<1e-04	<1e-04	0.87 ± 0.04, (0.77, 0.97), <1e-04	0.35 ± 0.04, (0.25, 0.45), <1e-04	-0.52 ± 0.04, (-0.62, -0.42), <1e-04
Peak Negative Ankle Power								
Intact	1.04 ± 0.04	0.88 ± 0.04	0.77 ± 0.04	<1e-04	<1e-04	0.17 ± 0.05, (0.05, 0.29), 0.0037	0.27 ± 0.05, (0.15, 0.39), <1e-04	0.10 ± 0.05, (-0.02, 0.23), 0.13
Prosthesis	1.07 ± 0.04	0.92 ± 0.04	0.67 ± 0.04	<1e-04	<1e-04	0.15 ± 0.05, (0.02, 0.27), 0.014	0.40 ± 0.05, (0.28, 0.52), <1e-04	0.26 ± 0.05, (0.14, 0.38), <1e-04
Peak Positive Knee Power								
Intact	0.35 ± 0.03	0.43 ± 0.03	0.36 ± 0.03	0.072	0.085	-0.09 ± 0.04, (-0.19, 0.01), 0.086	-0.01 ± 0.04, (-0.11, 0.09), 0.97	0.08 ± 0.04, (-0.02, 0.18), 0.15

Continued on next page

	Mean ± SE			F-test p*		Pairwise Differences: Mean ± SE, 95% CI, p-value		
Foot	**Prescribed**	**Passive**	**Active**	**raw**	**BH**	**Prescribed - Passive**	**Prescribed - Active**	**Passive - Active**
Prosthesis	0.11 ± 0.03	0.10 ± 0.03	0.10 ± 0.03	0.96	0.96	0.01 ± 0.04, (-0.09, 0.10), 0.98	0.01 ± 0.04, (-0.09, 0.11), 0.96	0.00 ± 0.04, (-0.09, 0.10), 1.0
Peak Negative Knee Power								
Intact	0.64 ± 0.03	0.59 ± 0.03	0.56 ± 0.03	0.092	0.11	0.05 ± 0.04, (-0.04, 0.14), 0.39	0.08 ± 0.04, (-0.01, 0.17), 0.077	0.03 ± 0.04, (-0.06, 0.12), 0.67
Prosthesis	1.16 ± 0.03	0.78 ± 0.03	0.85 ± 0.03	<1e-04	<1e-04	0.38 ± 0.04, (0.29, 0.47), <1e-04	0.31 ± 0.04, (0.23, 0.4), <1e-04	-0.07 ± 0.04, (-0.15, 0.02), 0.20
Peak Positive Hip Power								
Intact	0.71 ± 0.03	0.66 ± 0.03	0.60 ± 0.03	0.025	0.034	0.04 ± 0.04, (-0.05, 0.14), 0.49	0.11 ± 0.04, (0.01, 0.20), 0.019	0.06 ± 0.04, (-0.03, 0.16), 0.27
Prosthesis	0.77 ± 0.03	0.55 ± 0.03	0.66 ± 0.03	<1e-04	<1e-04	0.22 ± 0.04, (0.13, 0.31), <1e-04	0.11 ± 0.04, (0.02, 0.20), 0.015	-0.11 ± 0.04, (-0.21, -0.02), 0.012
Peak Negative Hip Power								
Intact	0.30 ± 0.03	0.42 ± 0.03	0.28 ± 0.03	8.00e-04	0.0013	-0.11 ± 0.04, (-0.20, -0.03), 0.0075	0.02 ± 0.04, (-0.07, 0.11), 0.83	0.13 ± 0.04, (0.04, 0.22), 0.0013
Prosthesis	0.43 ± 0.03	0.46 ± 0.03	0.69 ± 0.03	<1e-04	<1e-04	-0.03 ± 0.04, (-0.11, 0.06), 0.75	-0.26 ± 0.04, (-0.35, -0.17), <1e-04	-0.23 ± 0.04, (-0.32, -0.15), <1e-04
Step Period								
Intact	1.19 ± 0.01	1.28 ± 0.01	1.27 ± 0.01	<1e-04	<1e-04	-0.09 ± 0.02, (-0.13, -0.05), <1e-04	-0.09 ± 0.02, (-0.13, -0.05), <1e-04	0.00 ± 0.02, (0.04, 0.04), 0.98
Prosthesis	1.19 ± 0.01	1.25 ± 0.01	1.27 ± 0.01	<1e-04	<1e-04	-0.06 ± 0.02, (-0.10, -0.02), 0.0013	-0.08 ± 0.02, (-0.12, 0.04), <1e-04	-0.02 ± 0.02, (-0.06, 0.02), 0.31
Stance								
Intact	65.0 ± 0.2	66.8 ± 0.3	66.4 ± 0.3	<1e-04	<1e-04	-1.8 ± 0.4, (-2.6, -1.0), <1e-04	-1.4 ± 0.4, (-2.2, -0.6), 2e-04	0.4 ± 0.4, (-0.4, 1.3), 0.48
Prosthesis	62.4 ± 0.2	59.9 ± 0.3	60.9 ± 0.3	<1e-04	<1e-04	2.5 ± 0.3, (1.7, 3.3), <1e-04	1.4 ± 0.3, (0.6, 2.3), 1e-04	-1.0 ± 0.4, (-1.9, -0.2), 0.0096

*The significance of the association between outcome and condition. BH stands for the Benjamini-Hochberg correction, which adjusts p-values to maintain a false discovery rate of 5%.

F.2 Asymmetry Measurement Outcomes

Table F.2: Mean asymmetry outcomes and mean pairwise differences by condition from linear regression of outcome on condition with study participant ID and foot as covariates

	Mean ± SE			F-test p*		Pairwise Differences: Mean ± SE, 95% CI, p-value		
Outcome	**Prescribed**	**Passive**	**Active**	**raw**	**BH**	**Prescribed - Passive**	**Prescribed - Active**	**Passive - Active**
Support Moment	5.42 ± 0.08	4.91 ± 0.09	4.89 ± 0.09	<1e-04	<1e-04	0.51 ± 0.12, (0.23, 0.80), 1e-04	0.54 ± 0.12, (0.25, 0.83), <1e-04	0.02 ± 0.12, (-0.27, 0.32), 0.98
Ankle Angles	0.96 ± 0.02	0.87 ± 0.02	0.65 ± 0.02	<1e-04	<1e-04	0.09 ± 0.03, (0.01, 0.17), 0.021	0.30 ± 0.03, (0.22, 0.38), <1e-04	0.21 ± 0.03, (0.13, 0.29), <1e-04
Ankle Moments	2.77 ± 0.08	3.61 ± 0.08	3.63 ± 0.08	<1e-04	<1e-04	-0.84 ± 0.12, (-1.12, -0.57), <1e-04	-0.87 ± 0.12, (-1.14, -0.59), <1e-04	-0.02 ± 0.12, (-0.30, 0.26), 0.98
Ankle Power	3.2 ± 0.15	4.22 ± 0.16	3.66 ± 0.16	<1e-04	<1e-04	-1.03 ± 0.22, (-1.54, -0.51), <1e-04	-0.47 ± 0.22, (-0.98, 0.04), 0.080	0.56 ± 0.22, (0.04, 1.08), 0.033
Knee Angles	1.38 ± 0.04	1.38 ± 0.04	1.25 ± 0.04	0.021	0.021	-0.01 ± 0.06, (-0.14, 0.12), 0.99	0.13 ± 0.06, (0, 0.26), 0.045	0.14 ± 0.06, (0.01, 0.27), 0.039
Knee Moments	2.09 ± 0.06	2.29 ± 0.06	2.73 ± 0.06	<1e-04	<1e-04	-0.20 ± 0.08, (-0.39, -0.01), 0.038	-0.63 ± 0.08, (-0.82, -0.44), <1e-04	-0.44 ± 0.08, (-0.63, -0.24), <1e-04
Knee Power	3.25 ± 0.11	3.76 ± 0.12	3.81 ± 0.12	9.00e-04	0.001	-0.51 ± 0.16, (-0.9, -0.13), 0.0053	-0.55 ± 0.16, (-0.94, -0.17), 0.0023	-0.04 ± 0.17, (-0.43, 0.35), 0.97
Hip Angles	0.59 ± 0.02	0.64 ± 0.02	0.75 ± 0.02	<1e-04	<1e-04	-0.06 ± 0.03, (-0.11, 0), 0.073	-0.16 ± 0.03, (-0.22, -0.10), <1e-04	-0.10 ± 0.03, (-0.16, -0.04), 3e-04
Hip Moment	1.58 ± 0.04	2.20 ± 0.04	2.47 ± 0.04	<1e-04	<1e-04	-0.62 ± 0.06, (-0.77, -0.48), <1e-04	-0.89 ± 0.06, (-1.04, -0.75), <1e-04	-0.27 ± 0.06, (-0.42, -0.12), <1e-04
Hip Power	1.81 ± 0.07	2.99 ± 0.07	3.66 ± 0.07	<1e-04	<1e-04	-1.18 ± 0.09, (-1.40, -0.96), <1e-04	-1.86 ± 0.09, (-2.08, -1.63), <1e-04	-0.67 ± 0.10, (-0.90, -0.45), <1e-04
GRF	0.96 ± 0.05	2.25 ± 0.05	1.70 ± 0.05	<1e-04	<1e-04	-1.29 ± 0.07, (-1.45, -1.13), <1e-04	-0.74 ± 0.07, (-0.90, -0.58), <1e-04	0.55 ± 0.07, (0.39, 0.72), <1e-04

*The significance of the condition variable. BH stands for the Benjamini-Hochberg correction, which adjusts p-values to maintain a false discovery rate of 5%.

Appendix G

HUMAN SUBJECT EXPERIMENTAL RESULTS: CATEGORICAL SCATTER PLOTS

This appendix provides categorical scatter plots for the results discussed in Section 5.3. Each marker within the scatter plots represents the outcome for a collected gait cycle. Scatter plots display outcomes for each subject across all three experimental condition. Figures G.1 – G.18 show peak biomechanical outcomes, Figures G.19 and G.20 show temporal outcomes, and Figures G.21 – G.31 show asymmetry measurement outcomes.

G.1 Peak Outcomes

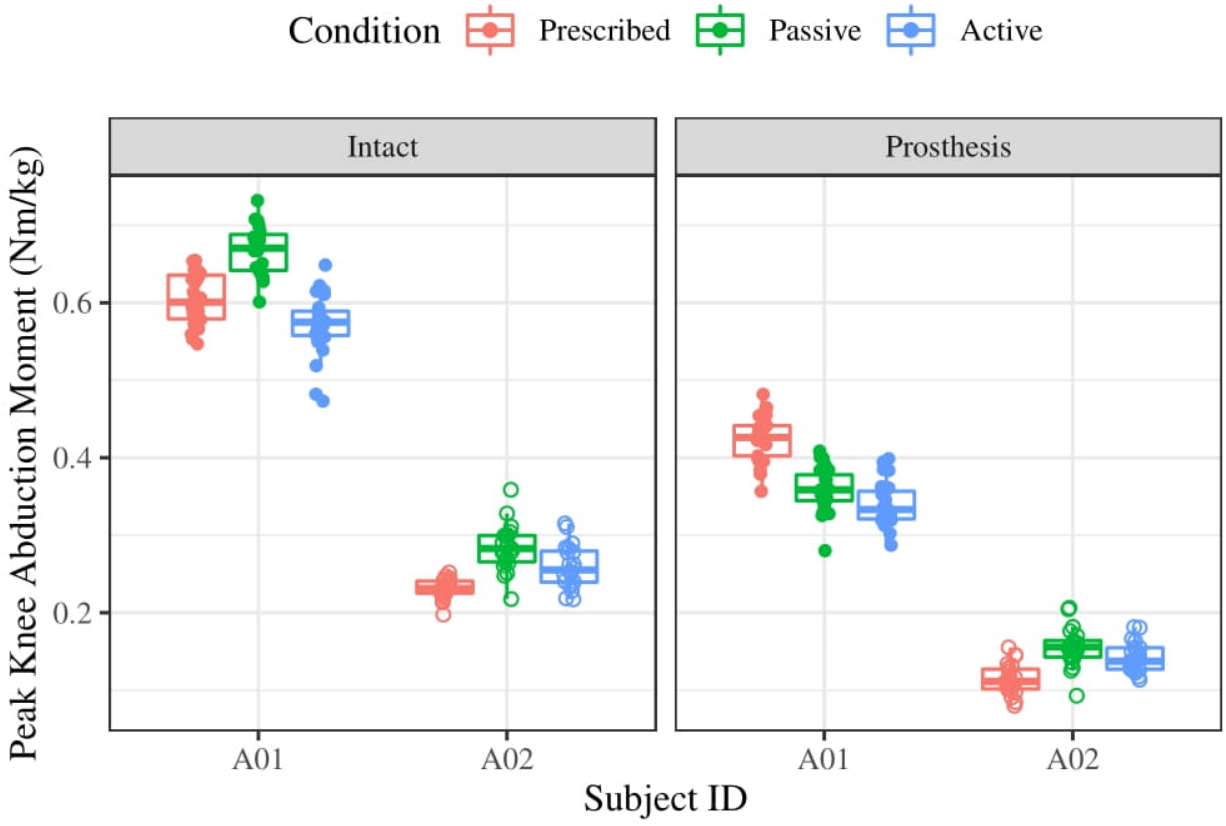


Figure G.1: Peak knee abduction moment outcomes for each subject and the three experimental conditions.

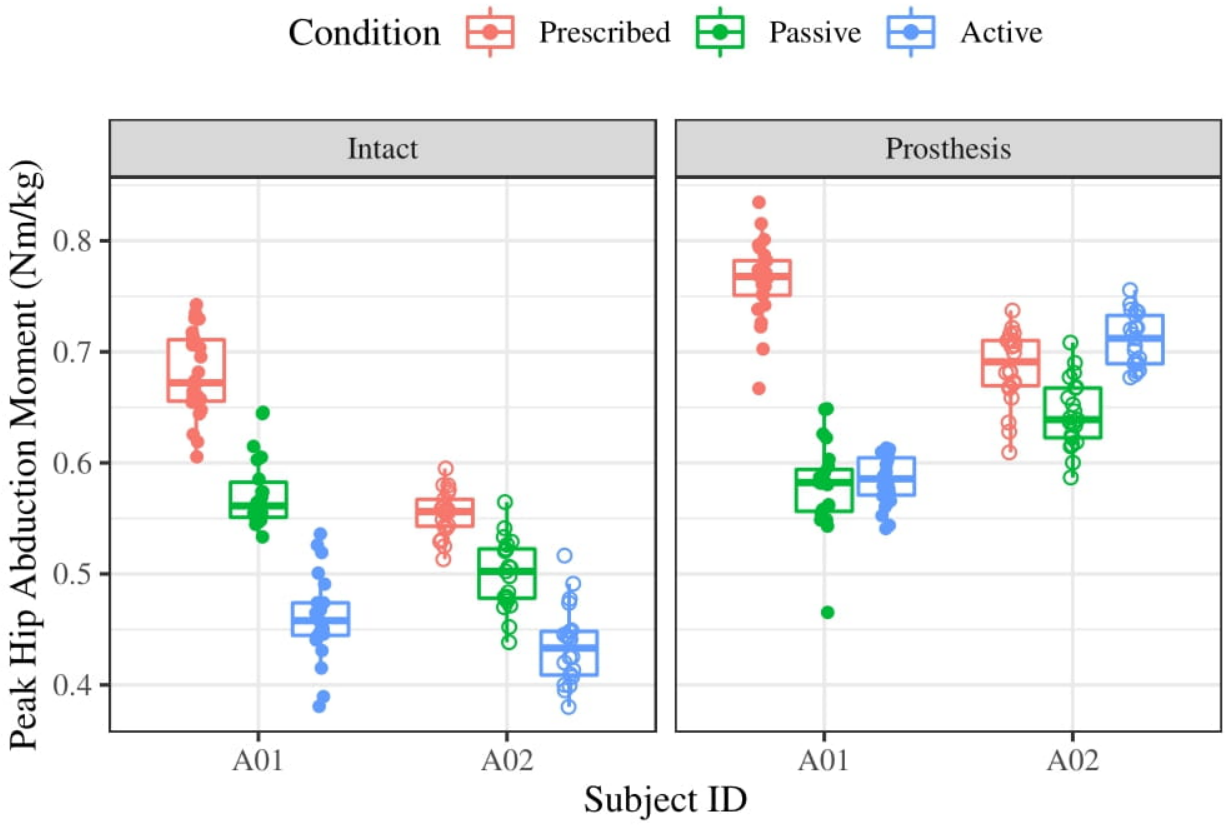


Figure G.2: Peak hip abduction moment outcomes for each subject and the three experimental conditions.

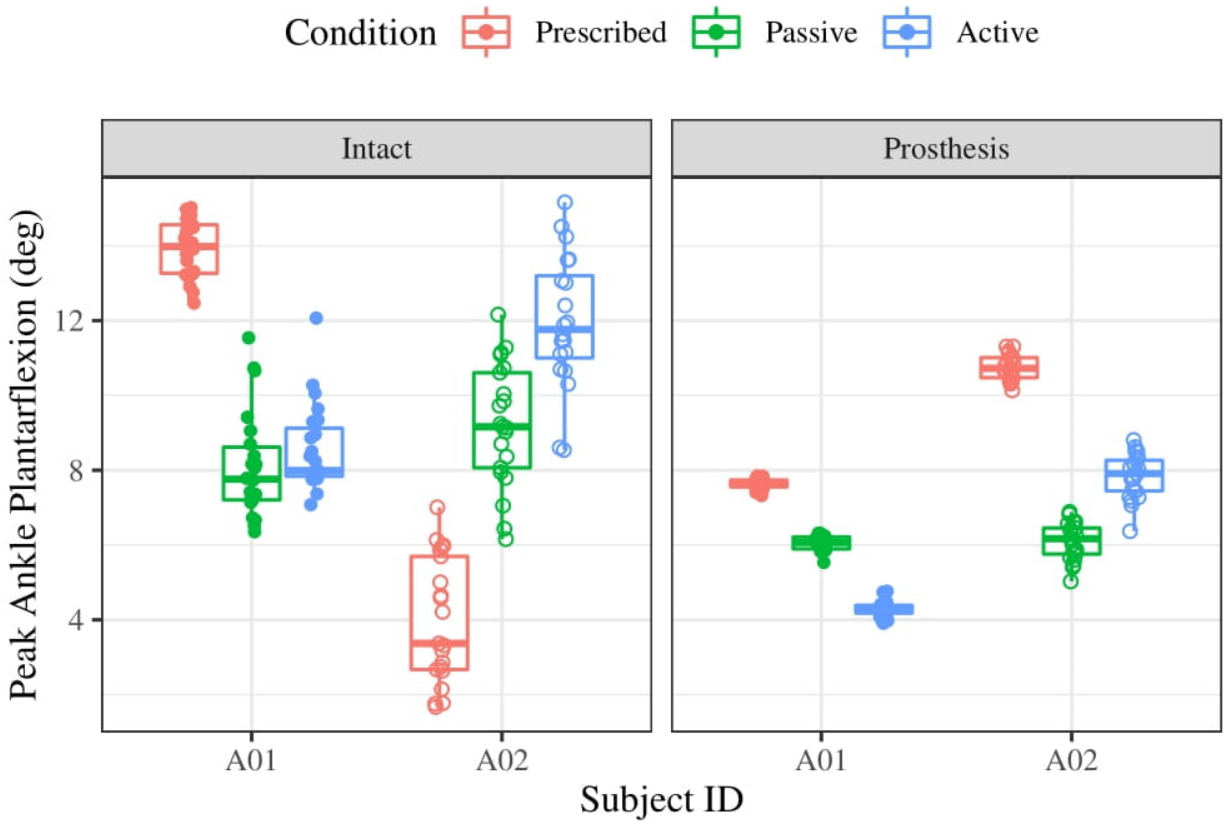


Figure G.3: Peak ankle plantarflexion angle outcomes for each subject and the three experimental conditions.

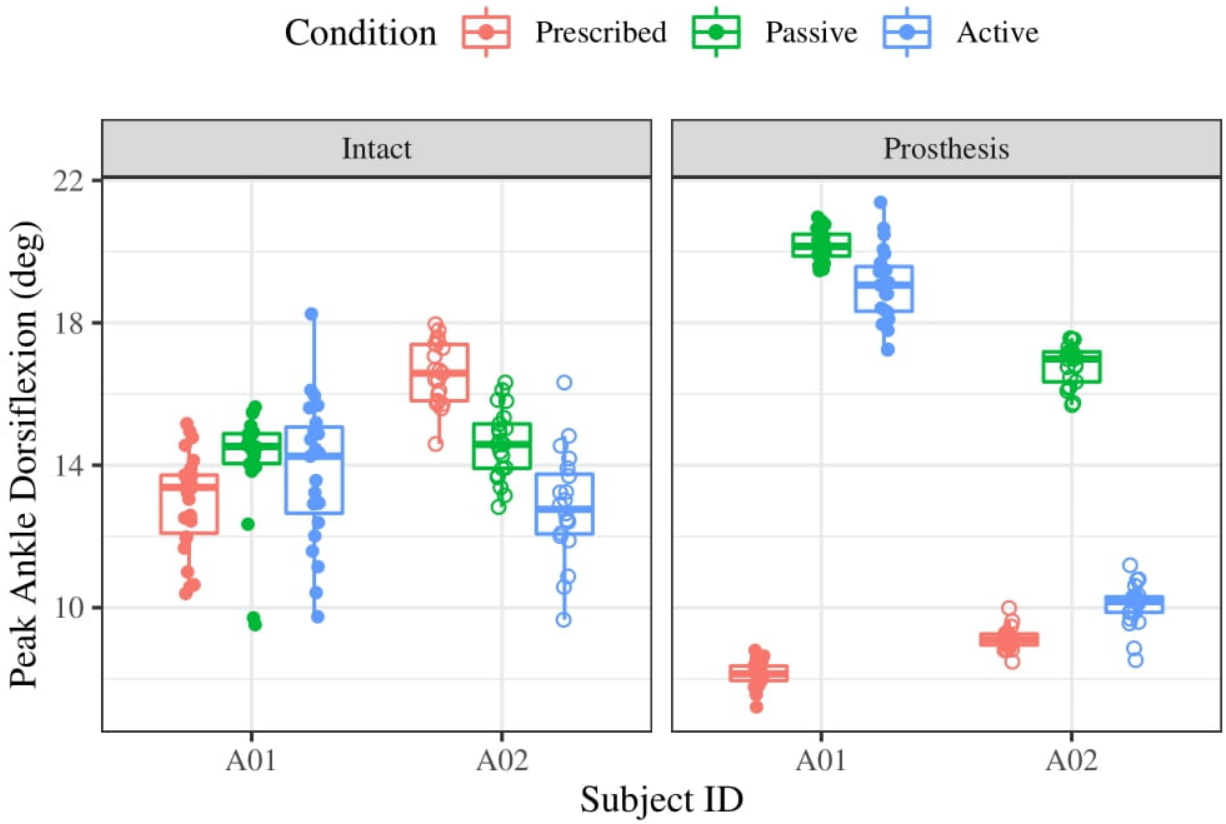


Figure G.4: Peak ankle dorsiflexion angle outcomes for each subject and the three experimental conditions.

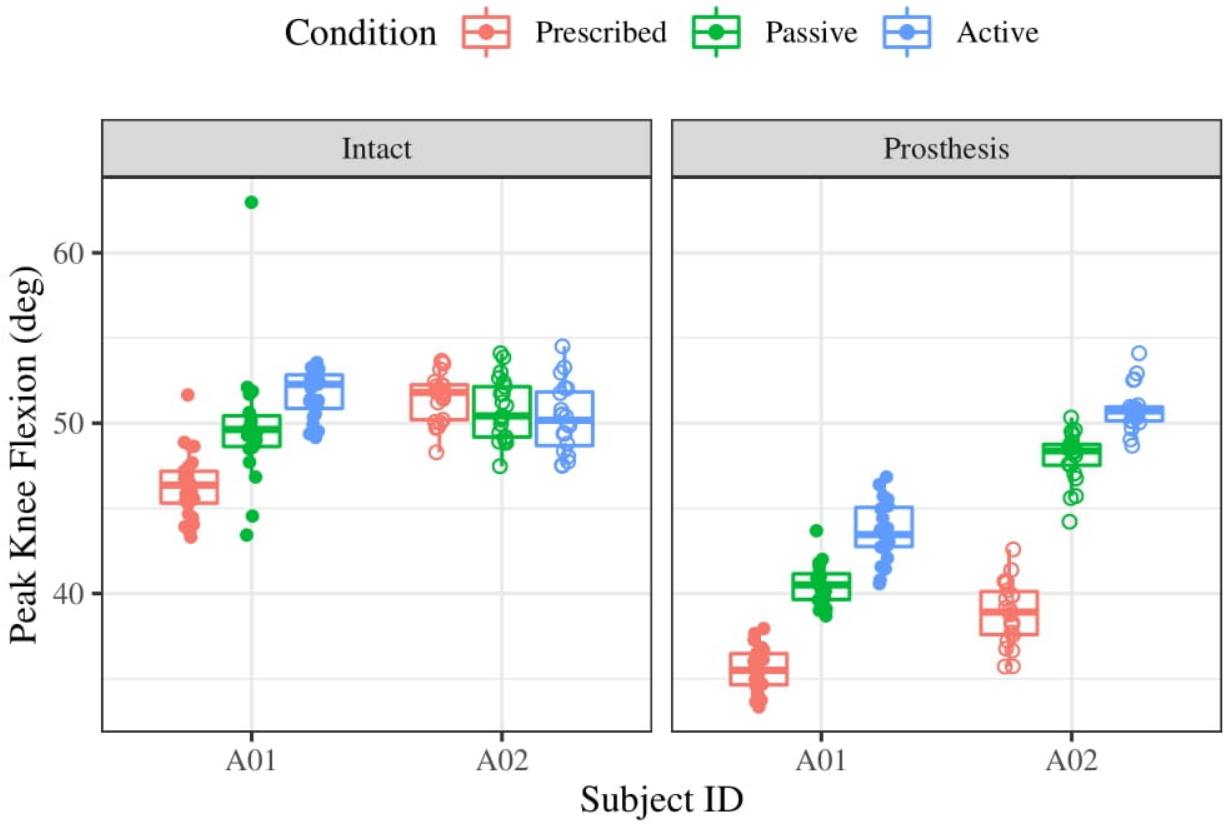


Figure G.5: Peak knee flexion angle outcomes for each subject and the three experimental conditions.

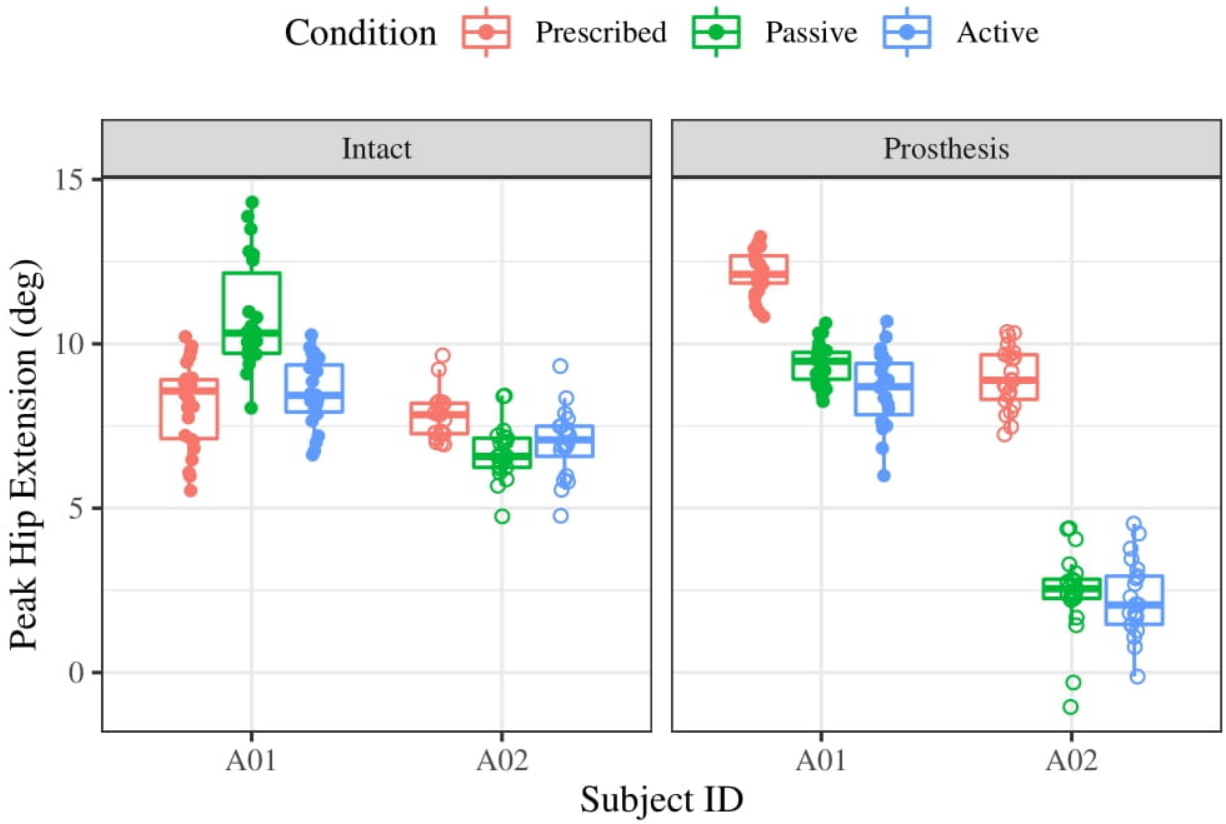


Figure G.6: Peak hip extension angle outcomes for each subject and the three experimental conditions.

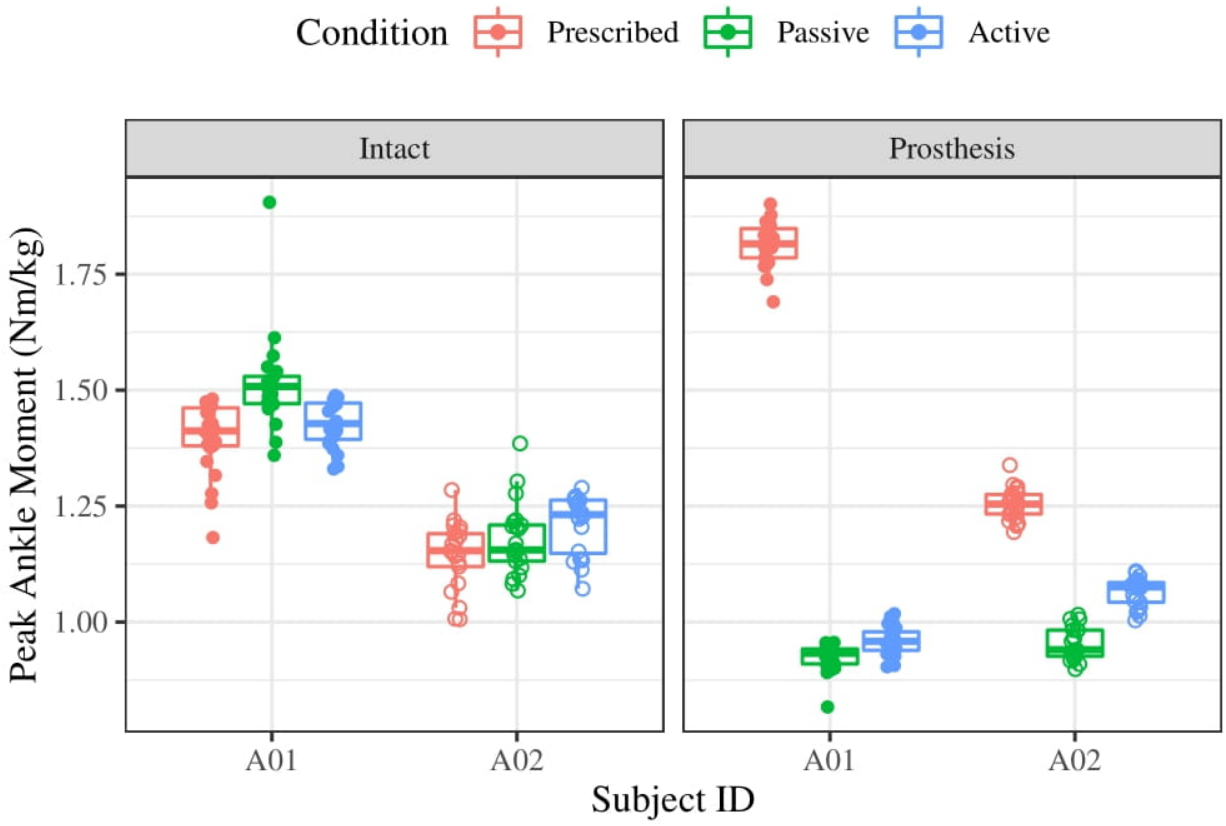


Figure G.7: Peak ankle moment outcomes for each subject and the three experimental conditions.

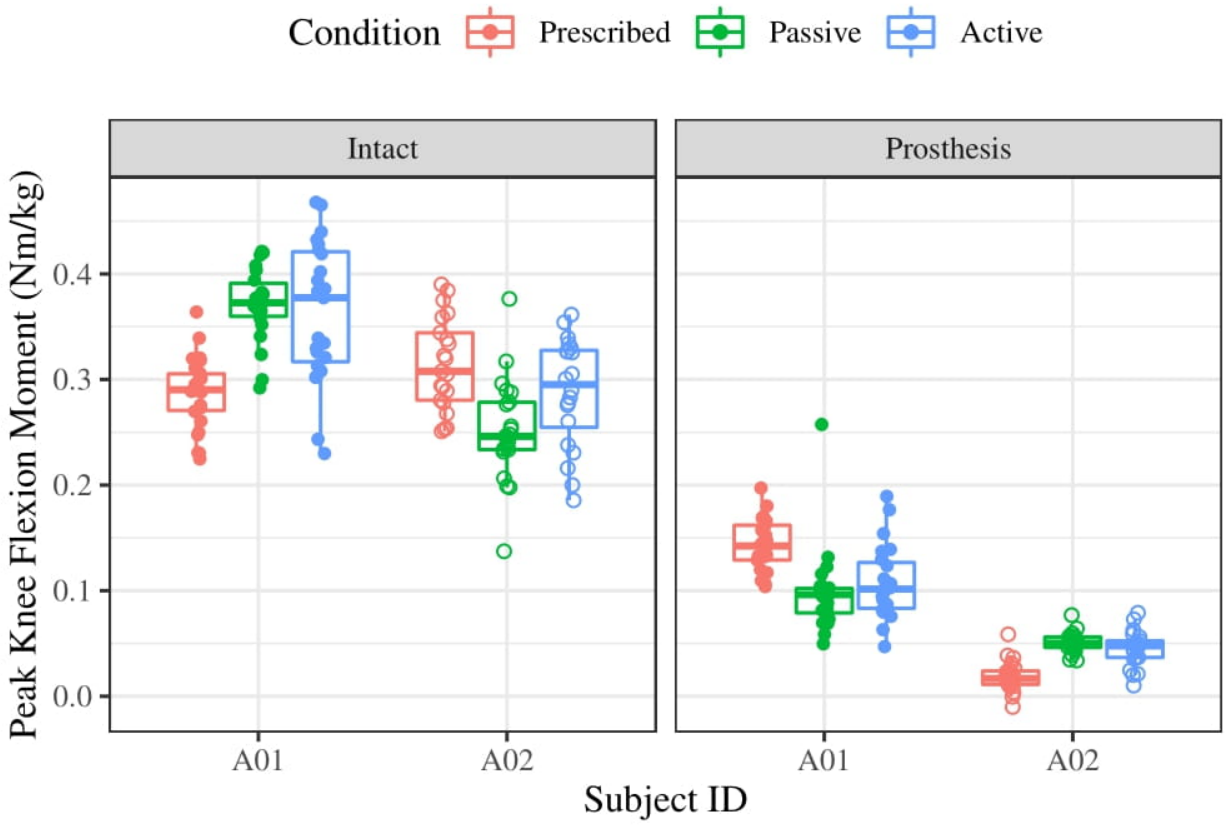


Figure G.8: Peak knee flexion moment outcomes for each subject and the three experimental conditions.

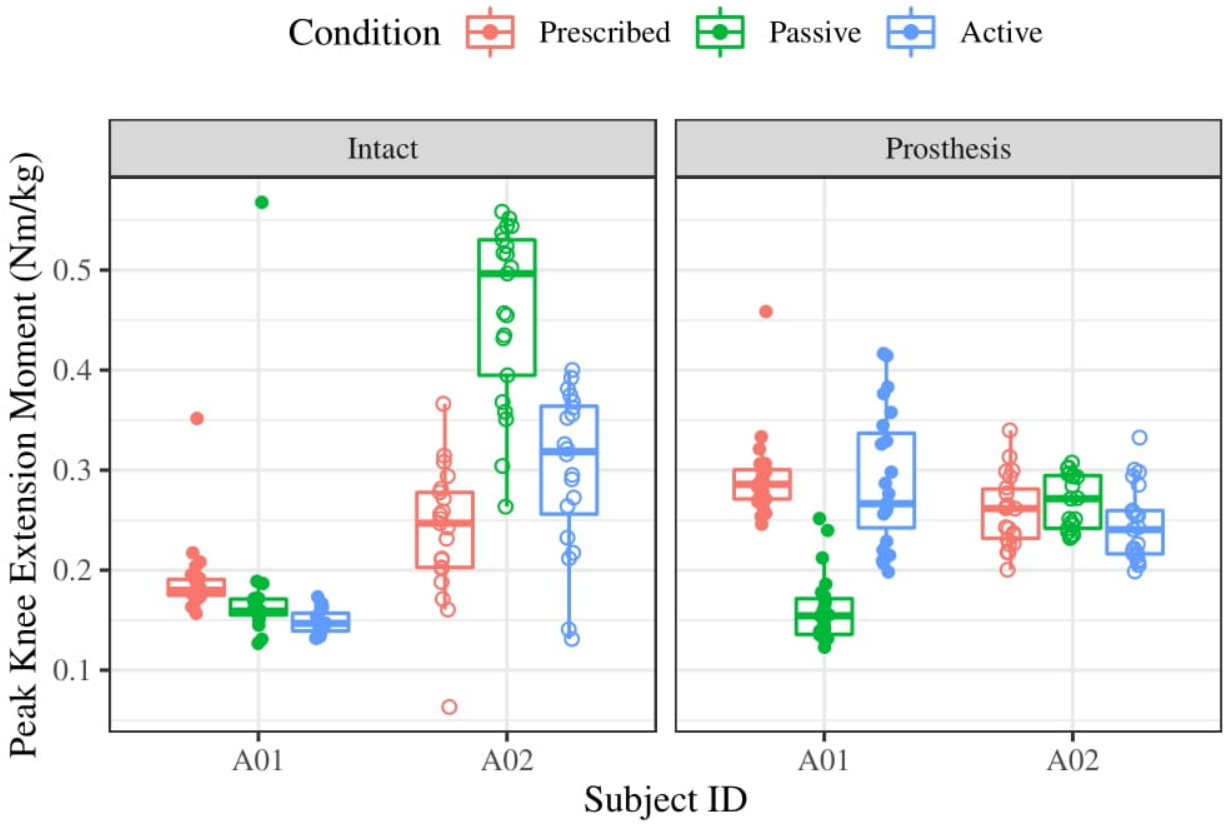


Figure G.9: Peak knee extension moment outcomes for each subject and the three experimental conditions.

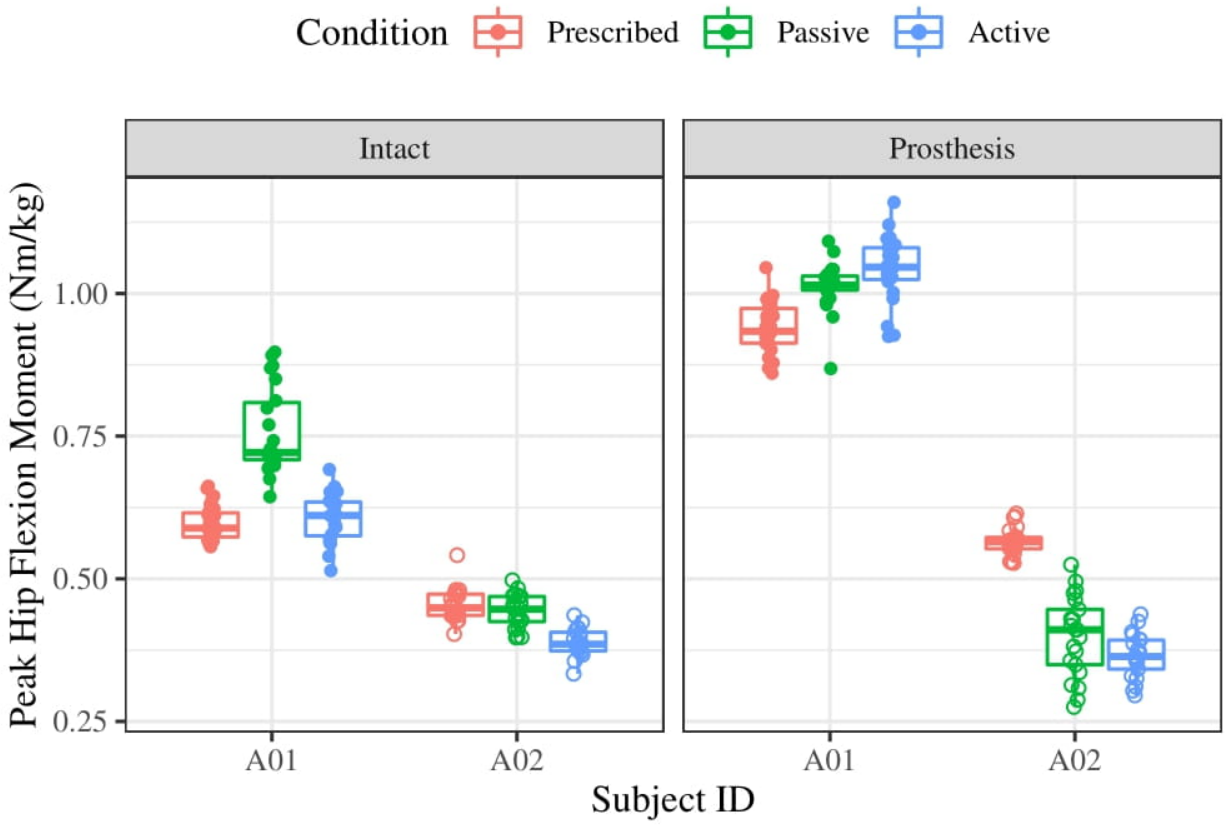


Figure G.10: Peak hip flexion moment outcomes for each subject and the three experimental conditions.

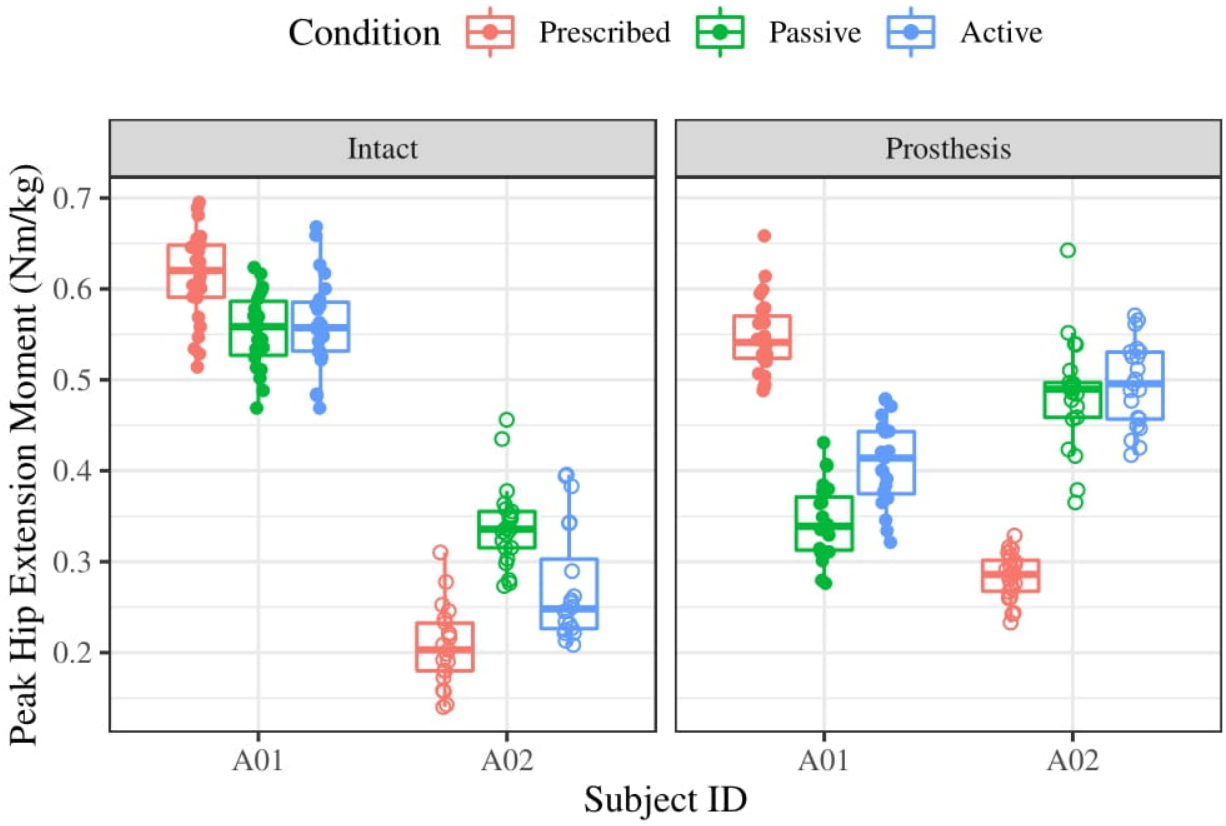


Figure G.11: Peak hip extension moment outcomes for each subject and the three experimental conditions.

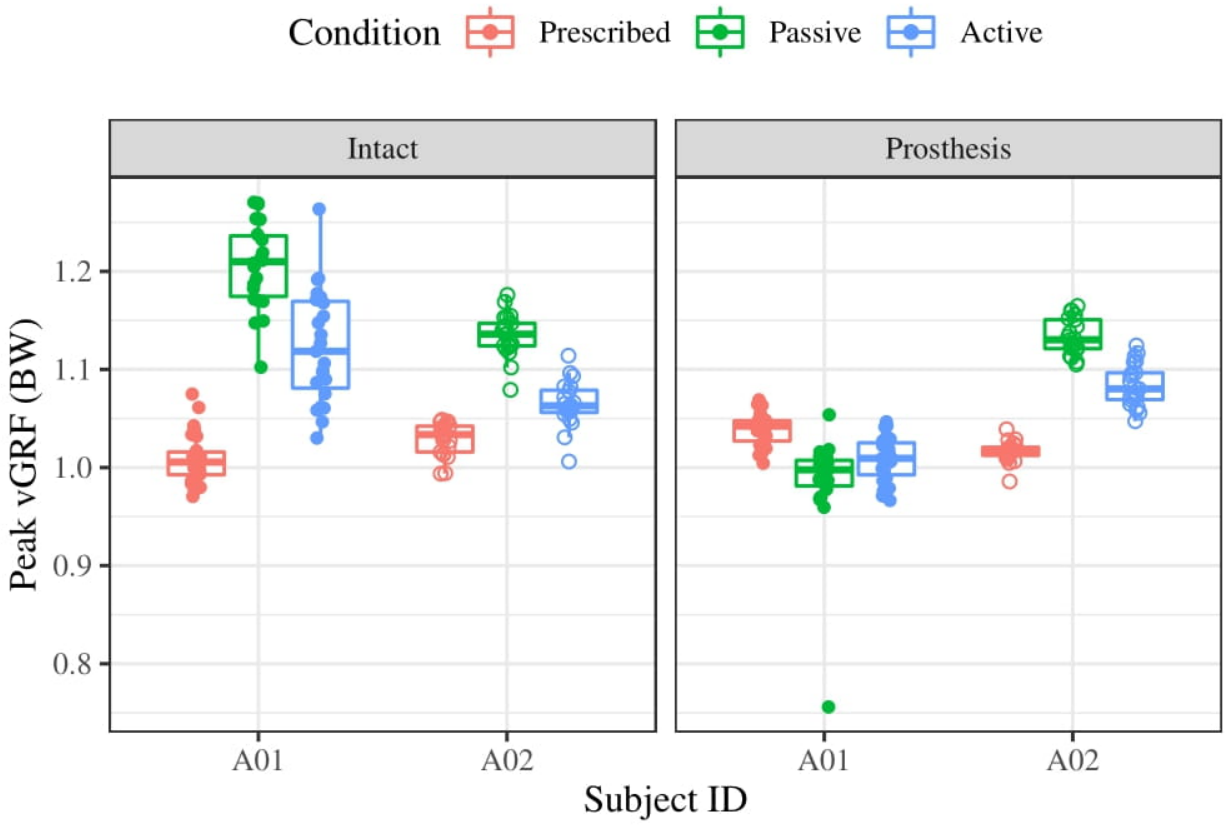


Figure G.12: Peak vertical ground reaction force outcomes for each subject and the three experimental conditions.

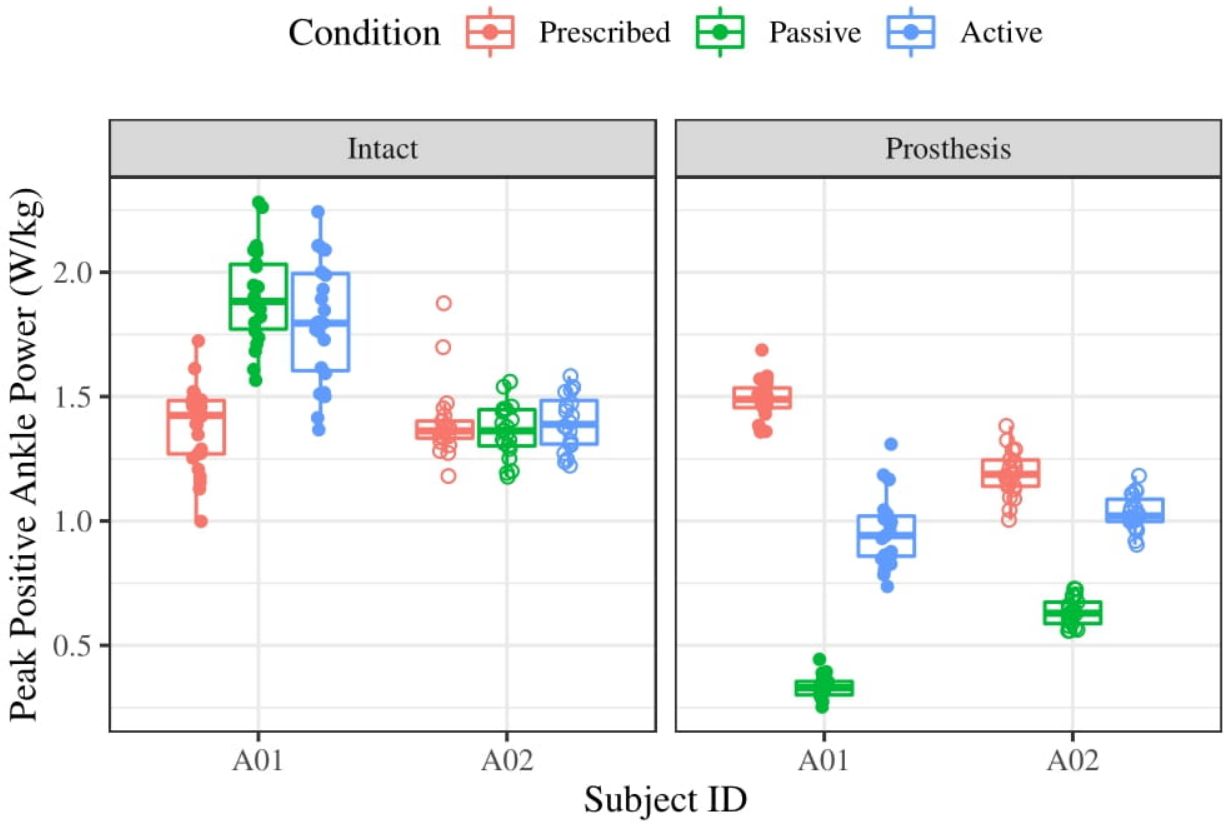


Figure G.13: Peak positive ankle power outcomes for each subject and the three experimental conditions.

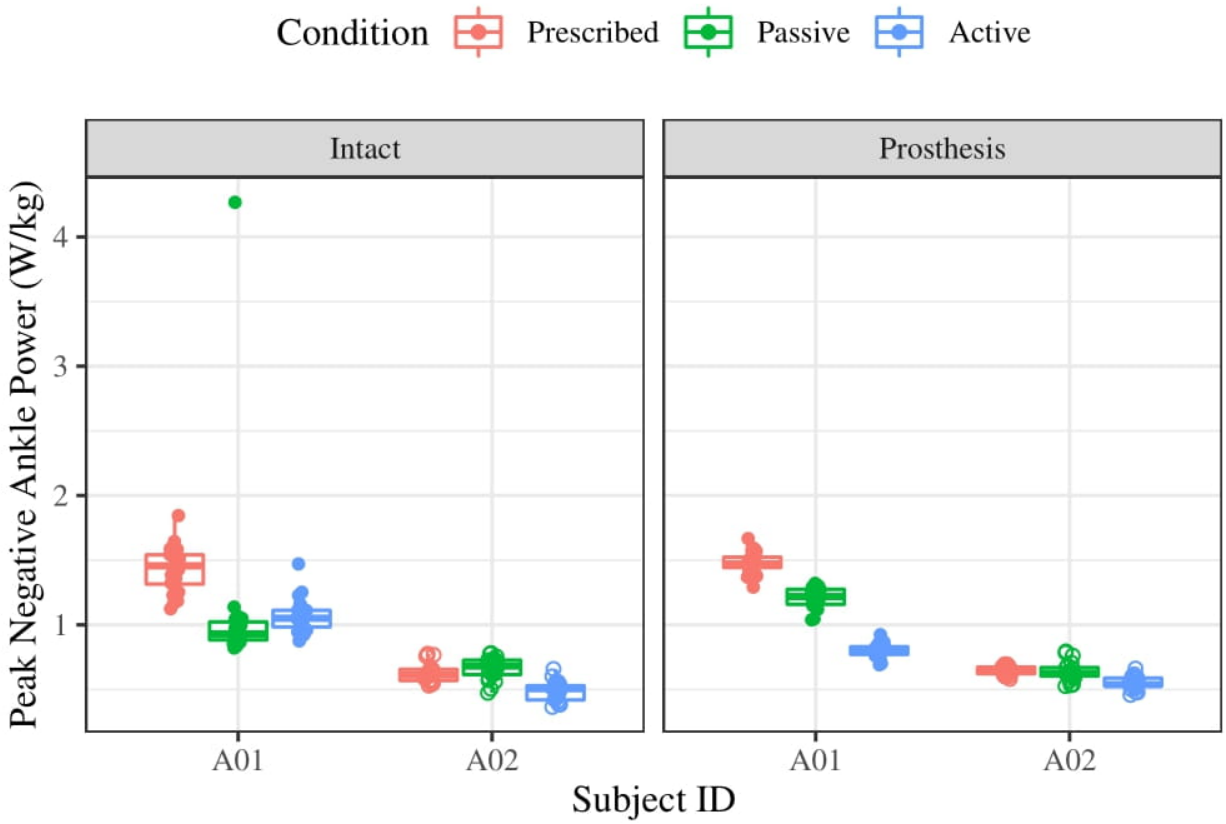


Figure G.14: Peak negative ankle power outcomes for each subject and the three experimental conditions.

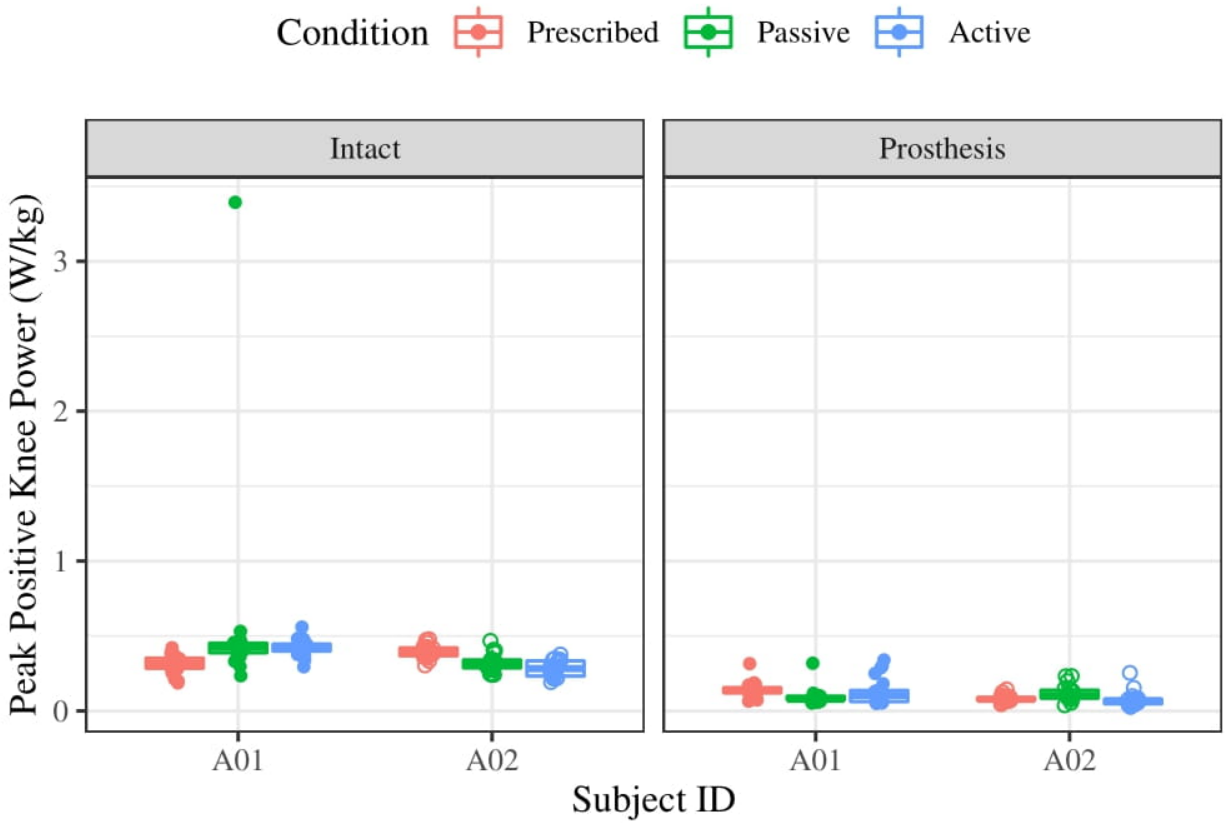


Figure G.15: Peak positive knee power outcomes for each subject and the three experimental conditions.

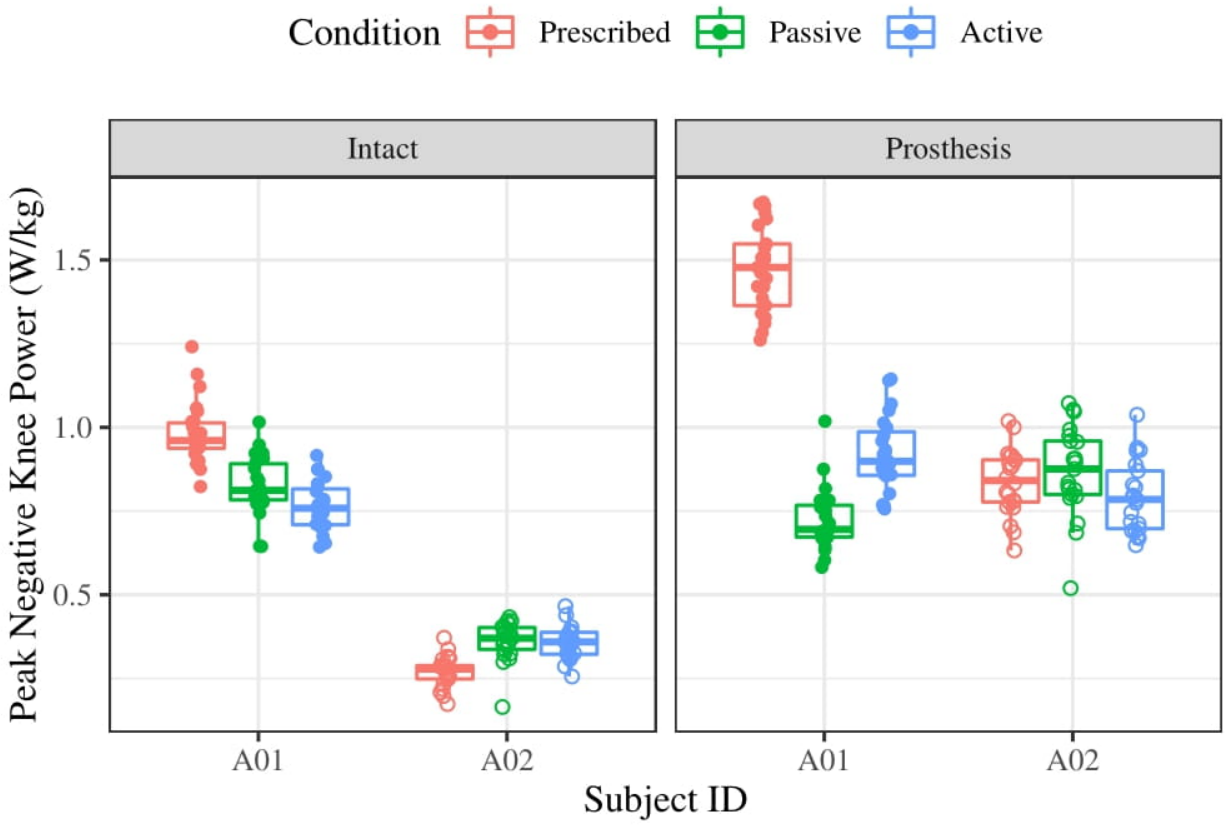


Figure G.16: Peak negative knee power outcomes for each subject and the three experimental conditions.

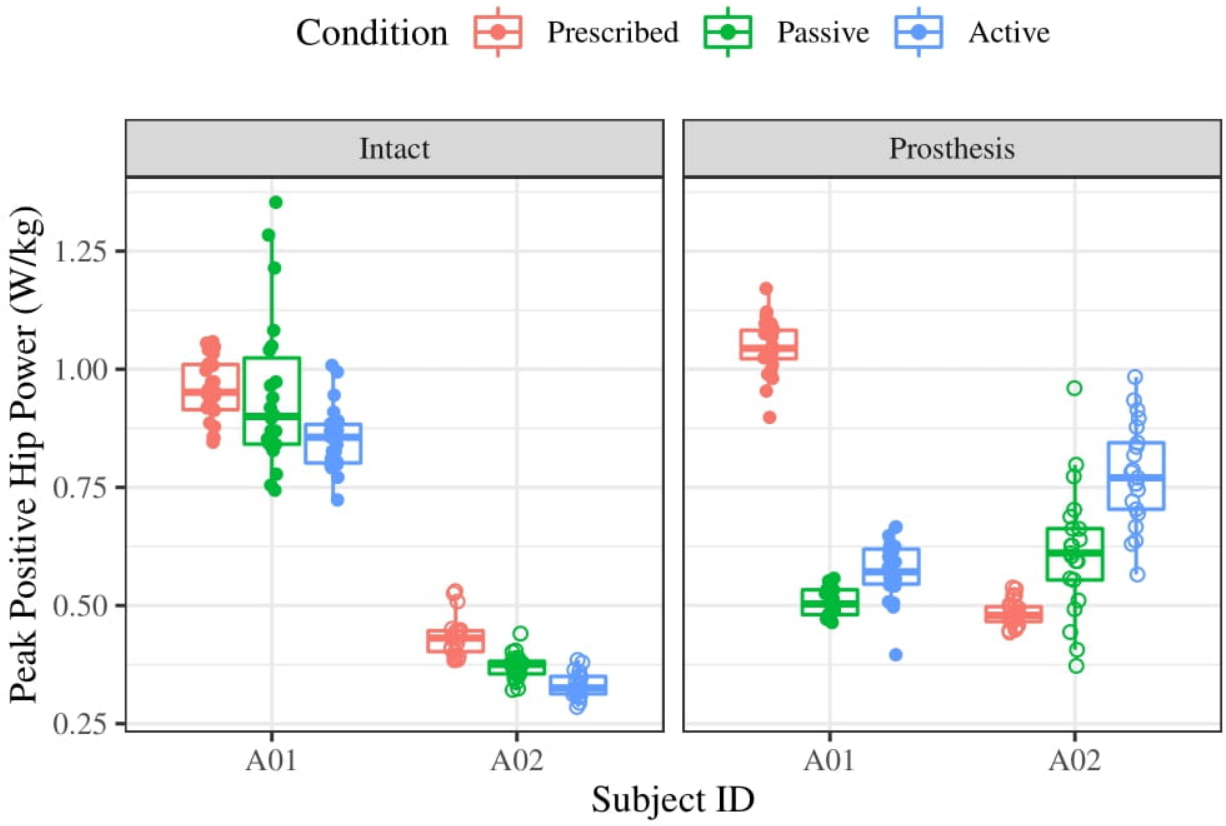


Figure G.17: Peak positive hip power outcomes for each subject and the three experimental conditions.

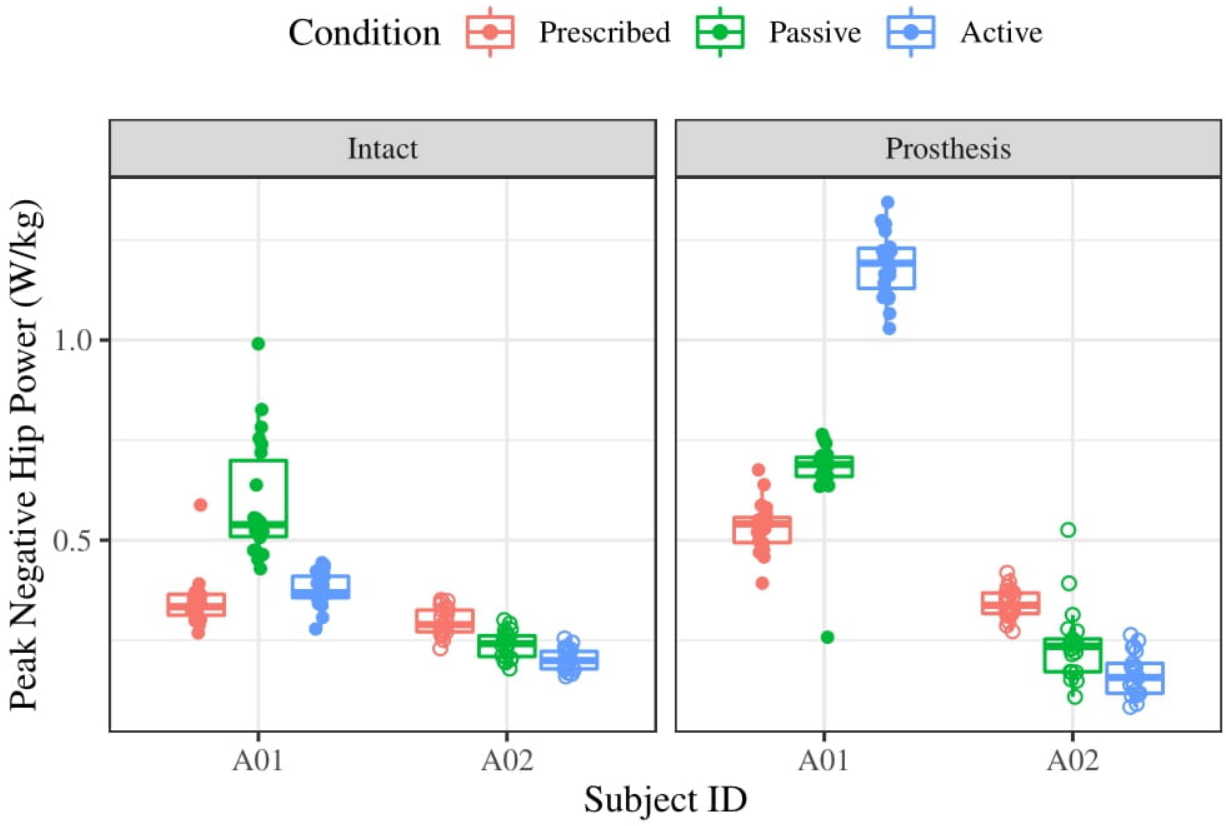


Figure G.18: Peak negative hip power outcomes for each subject and the three experimental conditions.

G.2 Temporal Outcomes

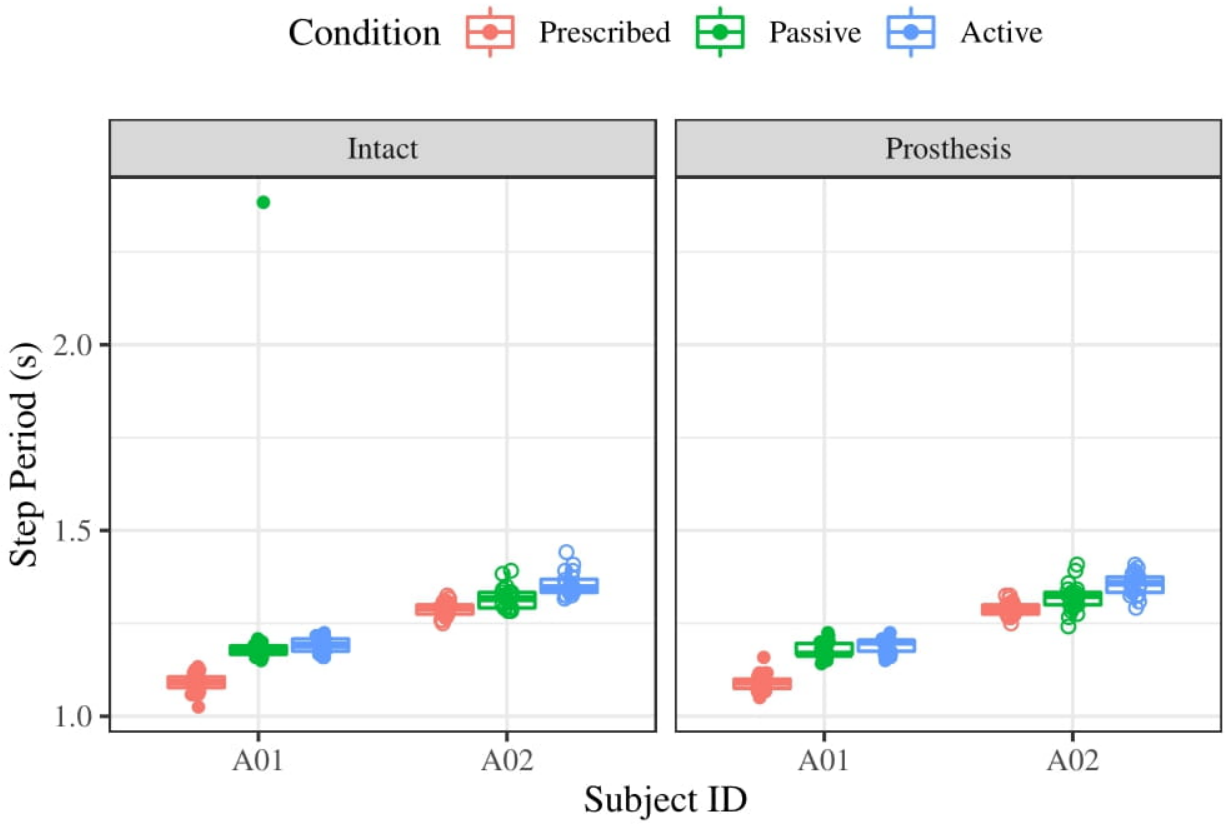


Figure G.19: Step period time outcomes for each subject and the three experimental conditions.

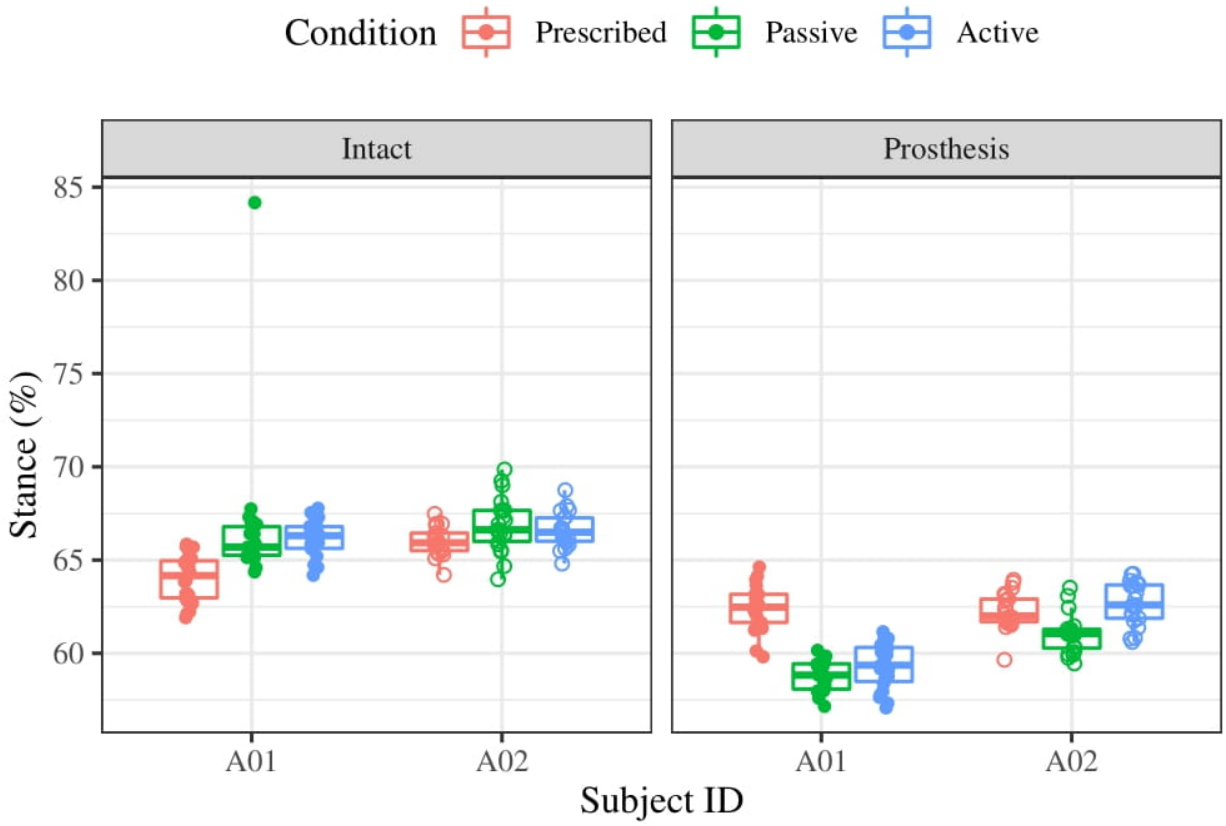


Figure G.20: Stance period outcomes for each subject and the three experimental conditions.

G.3 Asymmetry Measurement Outcomes

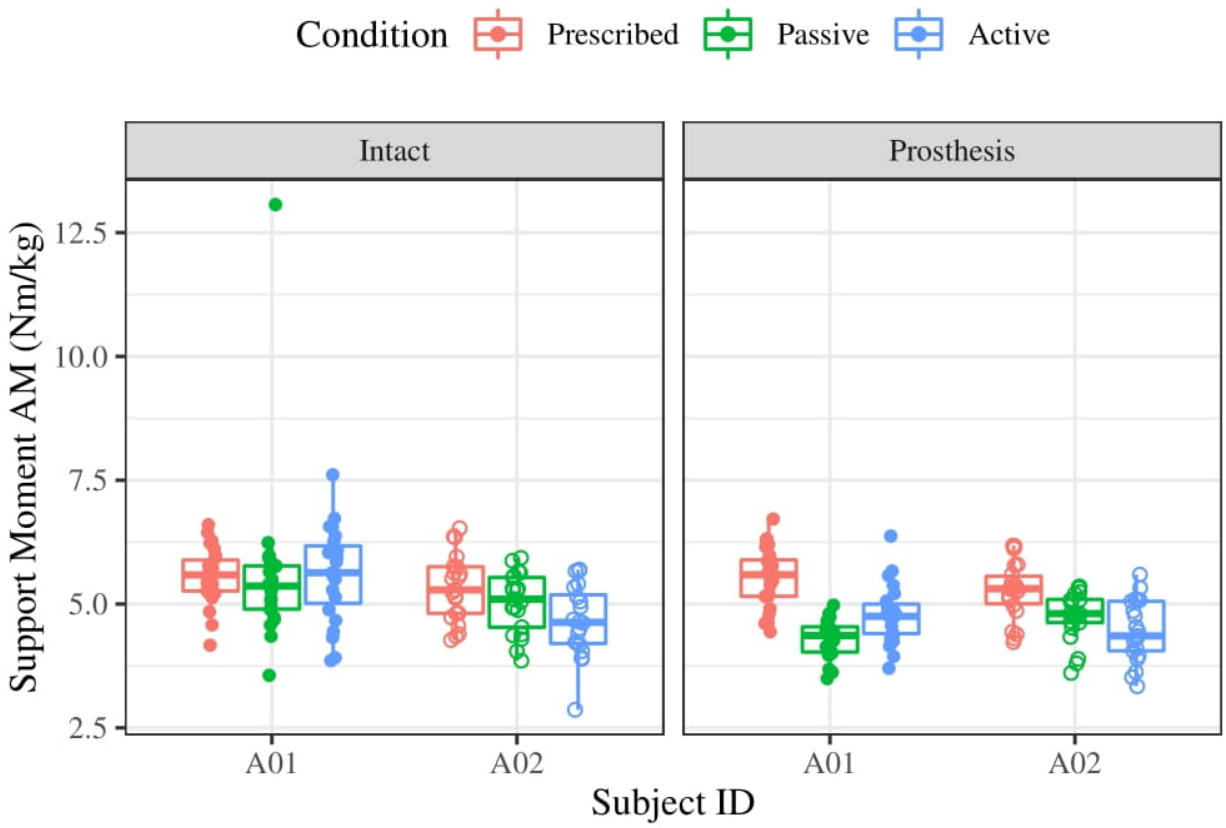


Figure G.21: Support moment asymmetry measurement outcomes for each subject and the three experimental conditions.

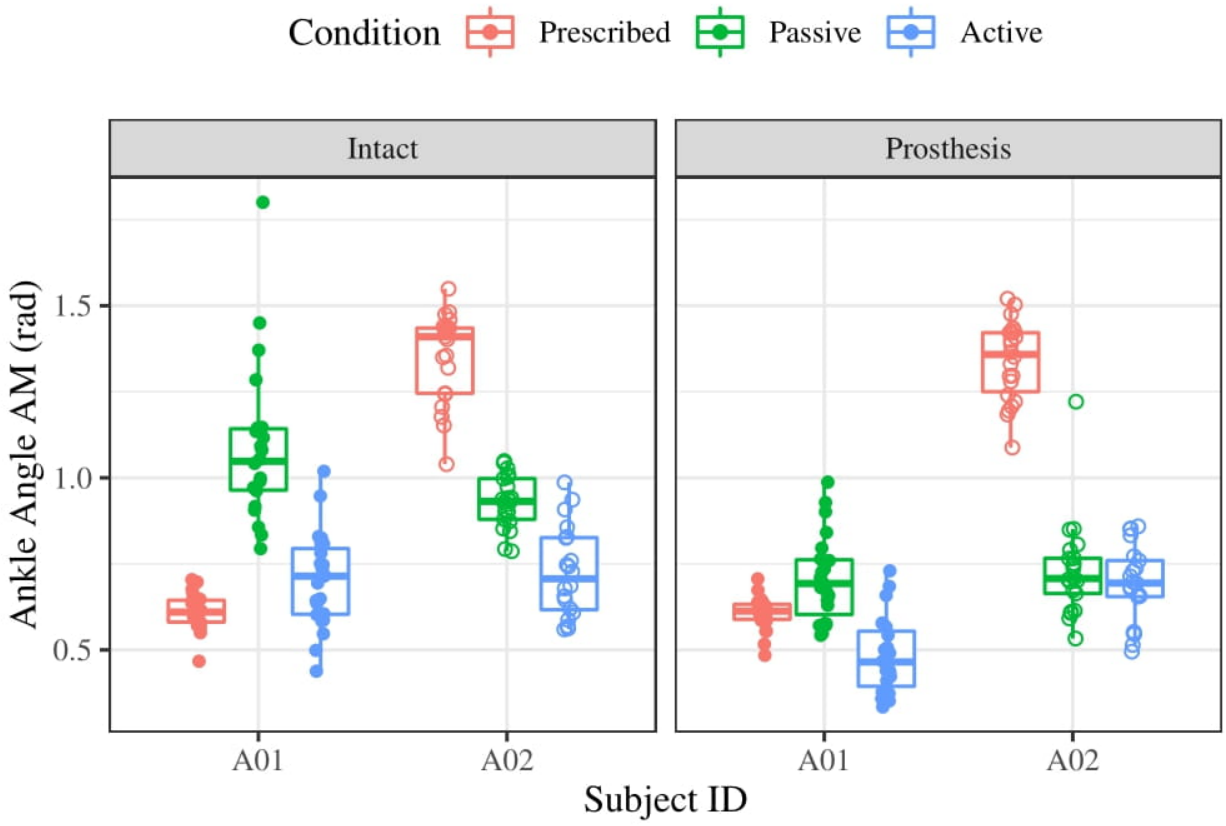


Figure G.22: Ankle angle asymmetry measurement outcomes for each subject and the three experimental conditions.

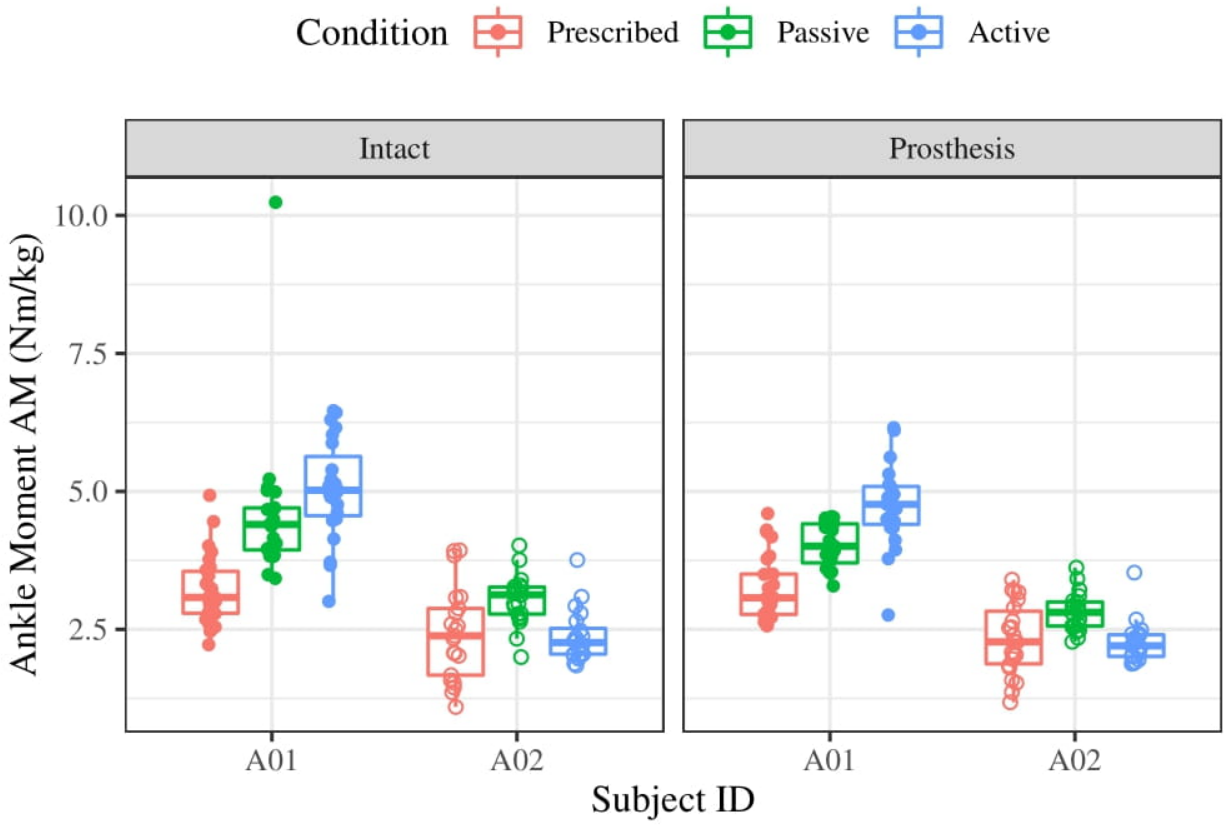


Figure G.23: Ankle moment asymmetry measurement outcomes for each subject and the three experimental conditions.

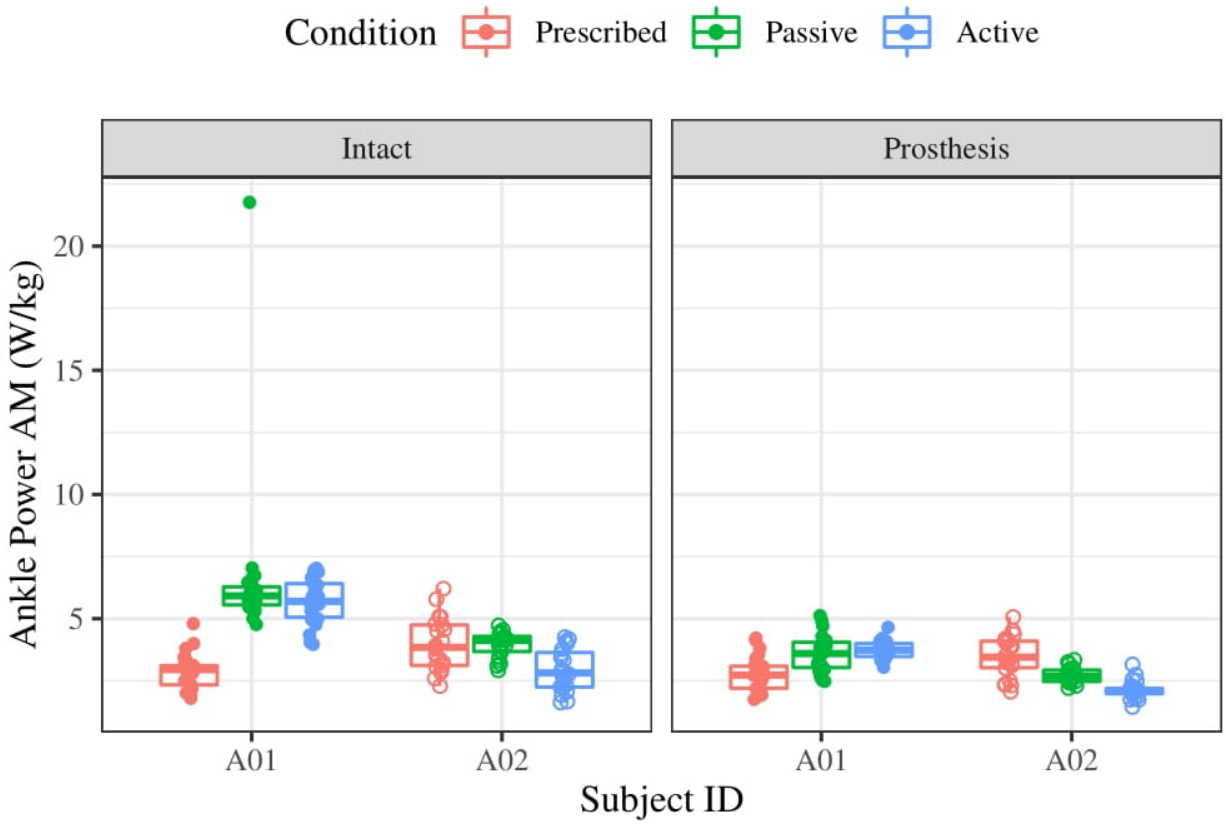


Figure G.24: Ankle power asymmetry measurement outcomes for each subject and the three experimental conditions.

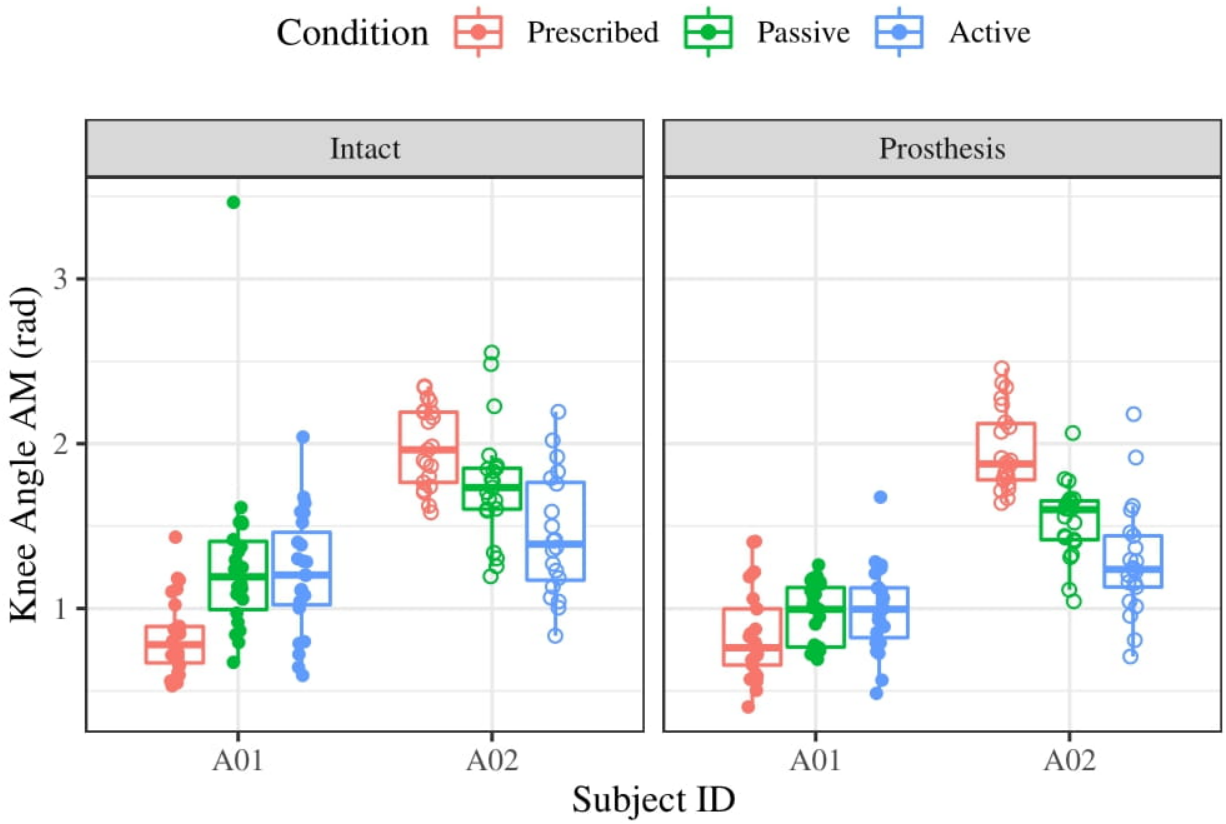


Figure G.25: Knee angle asymmetry measurement outcomes for each subject and the three experimental conditions.

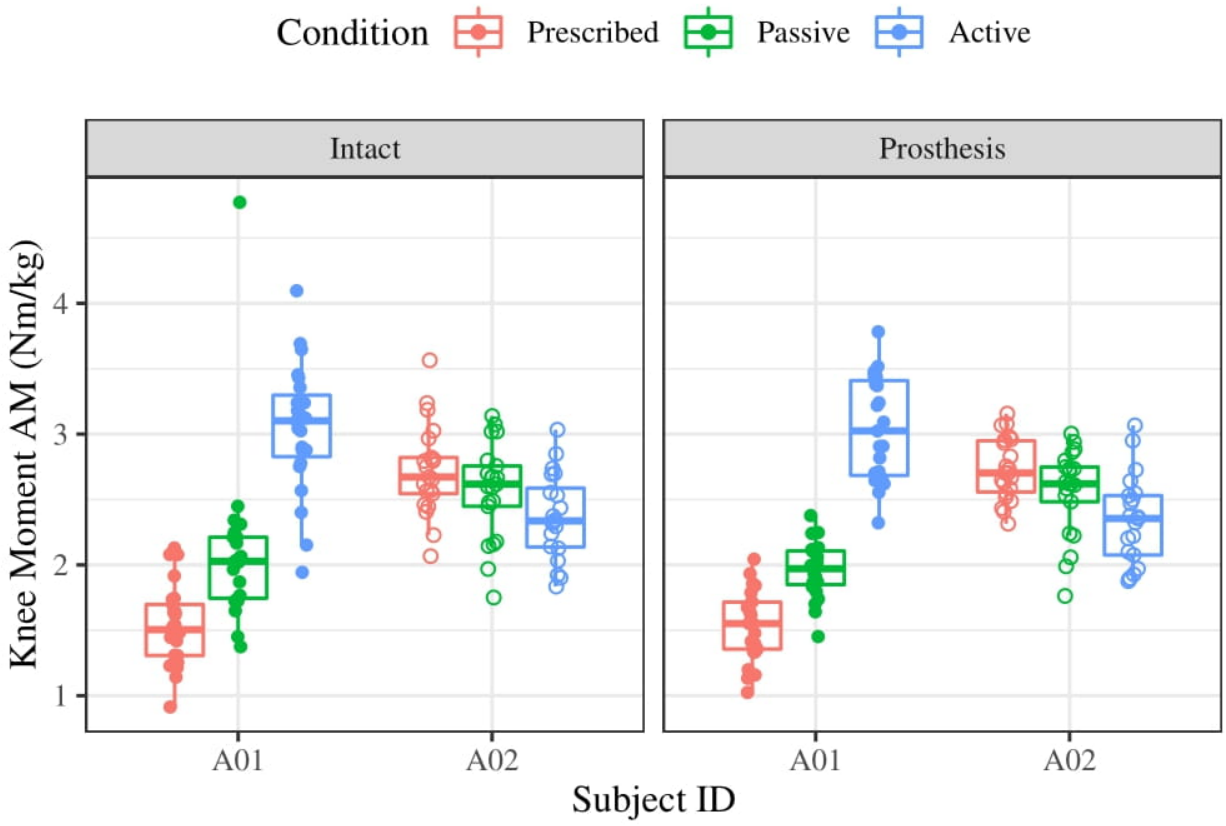


Figure G.26: Knee moment asymmetry measurement outcomes for each subject and the three experimental conditions.

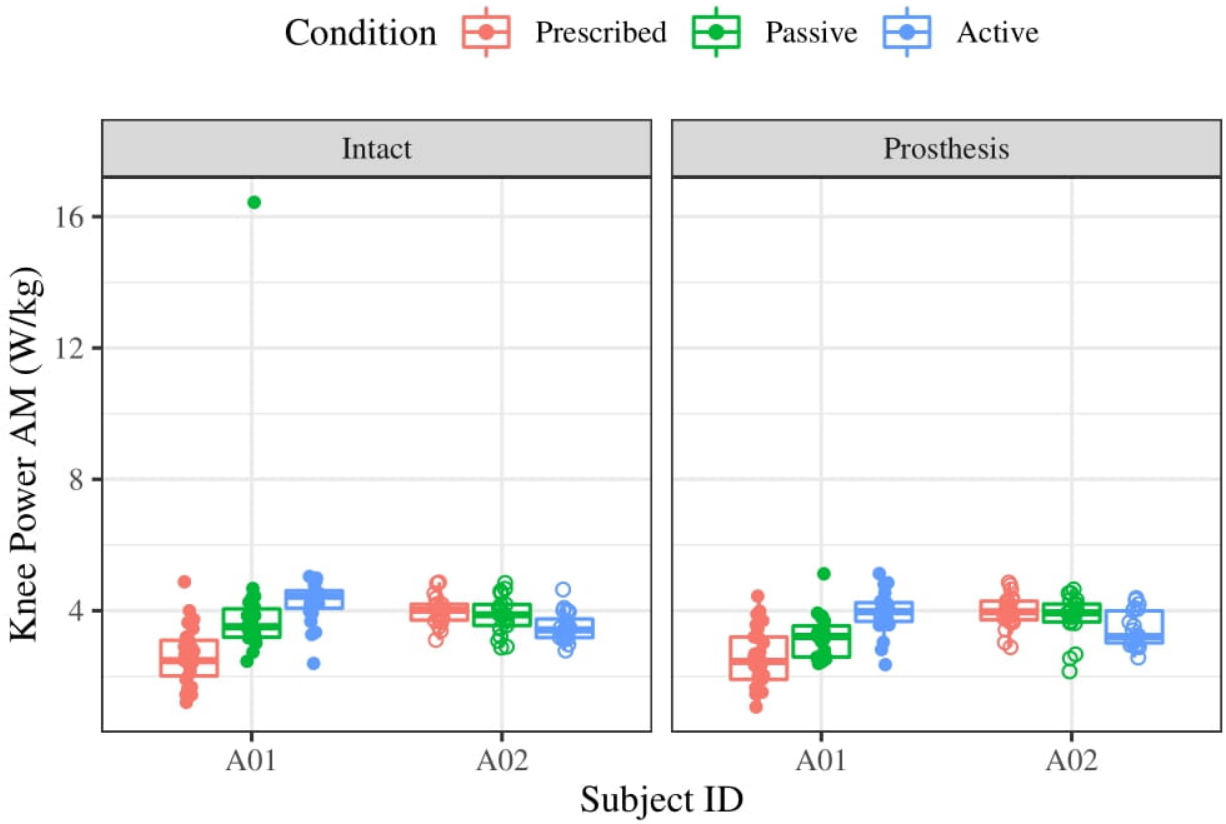


Figure G.27: Knee power asymmetry measurement outcomes for each subject and the three experimental conditions.

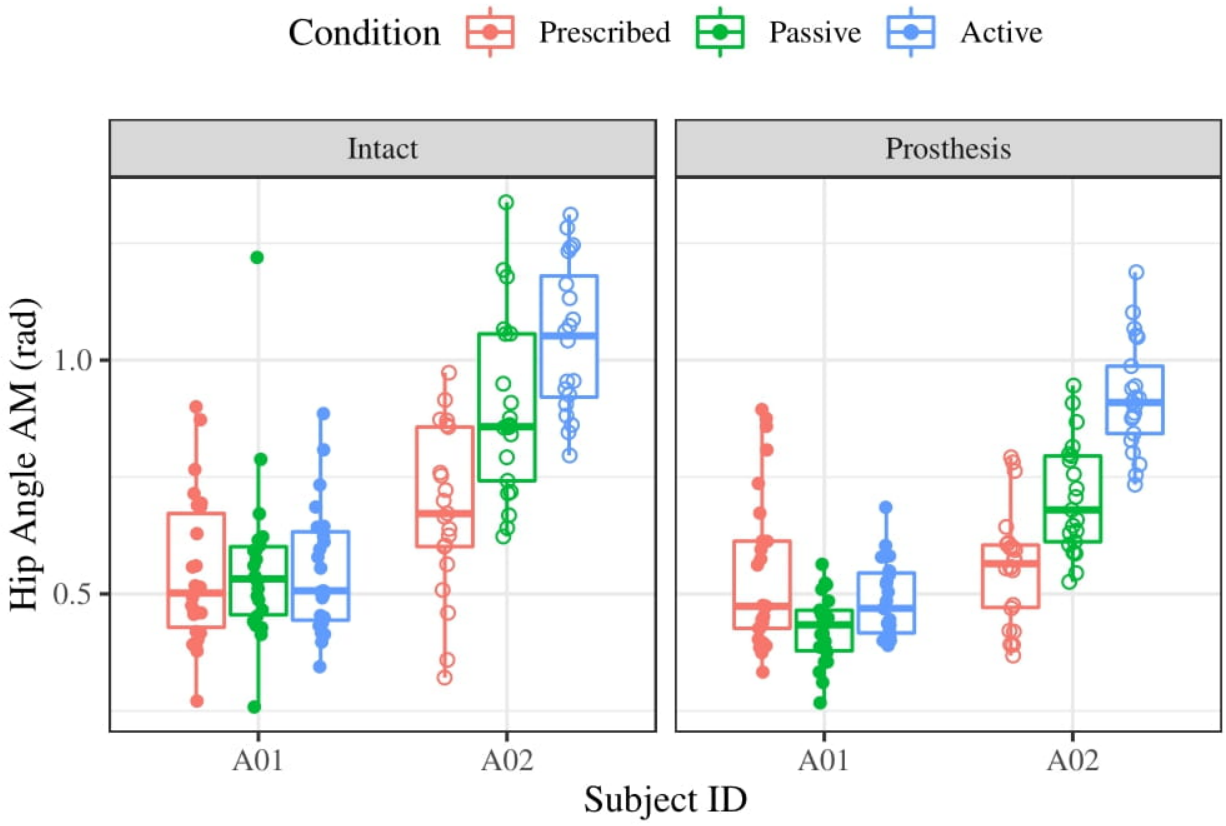


Figure G.28: Hip angle asymmetry measurement outcomes for each subject and the three experimental conditions.

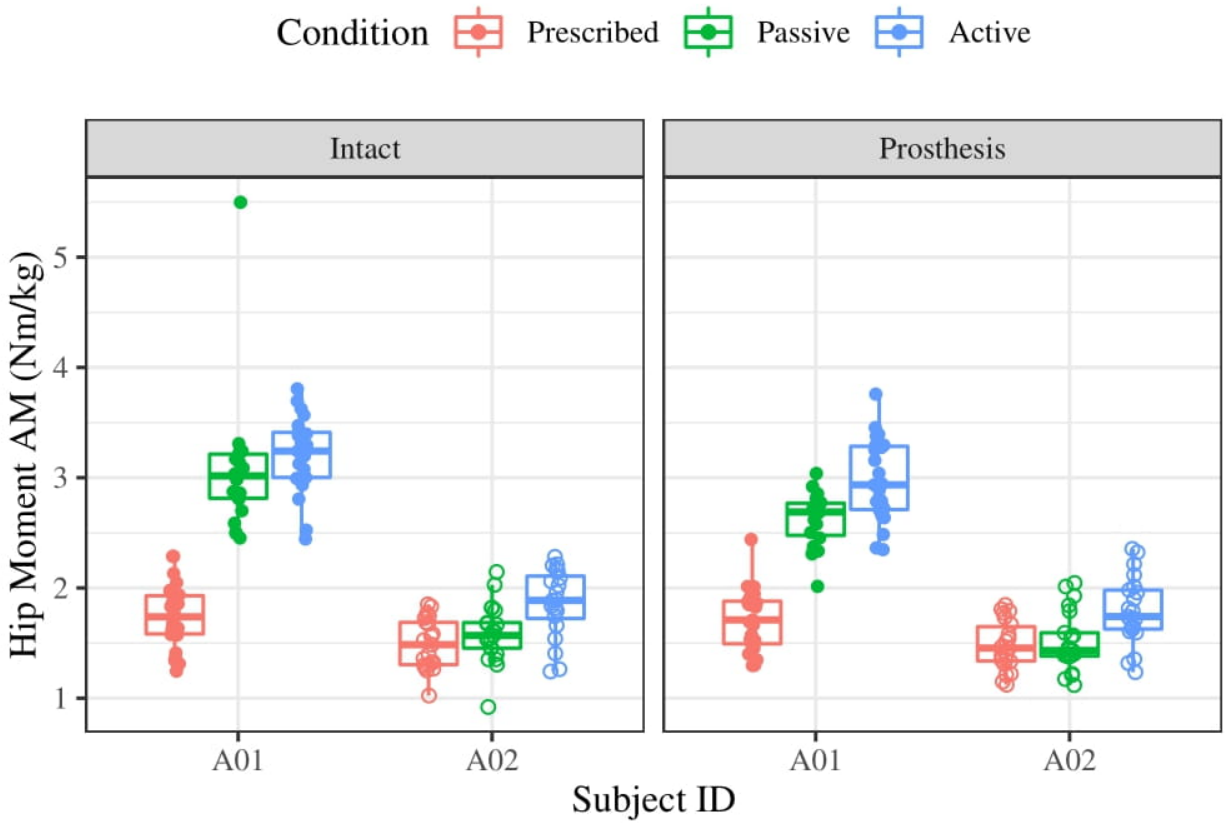


Figure G.29: Hip moment asymmetry measurement outcomes for each subject and the three experimental conditions.

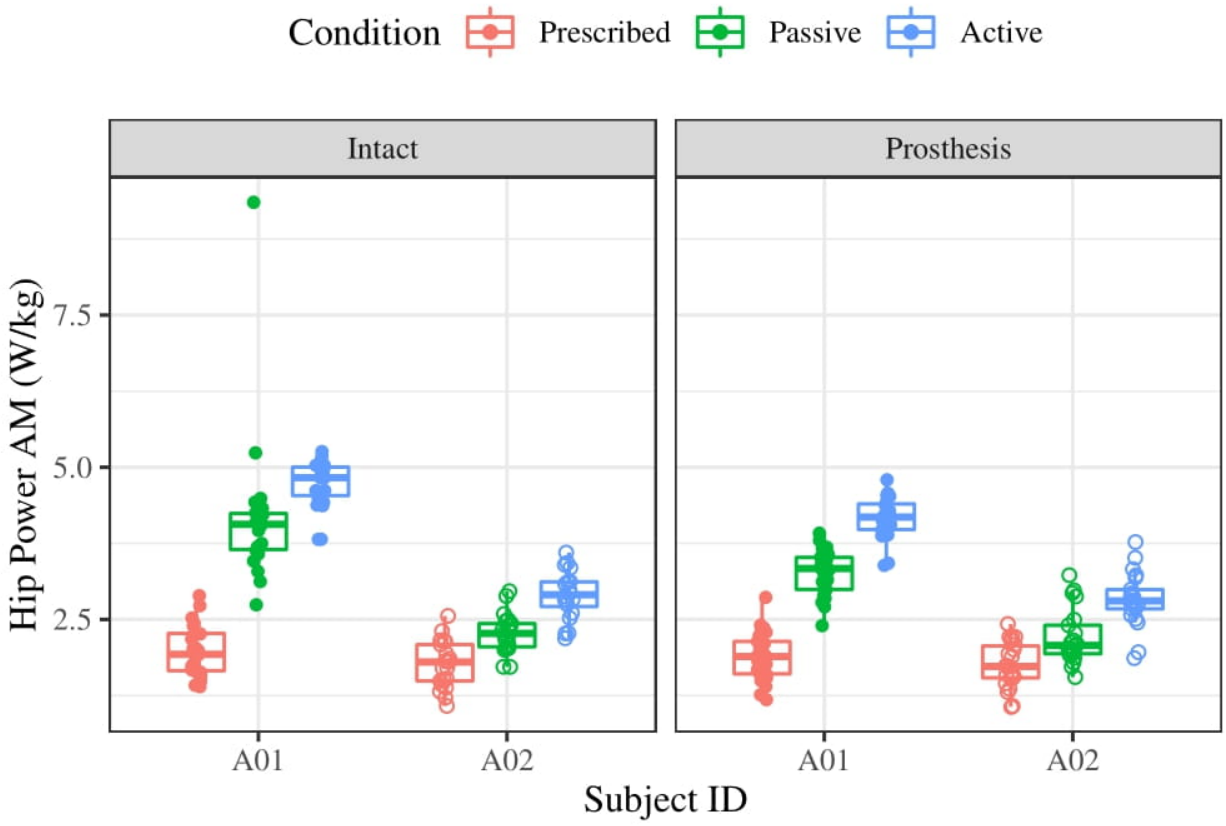


Figure G.30: Hip power asymmetry measurement outcomes for each subject and the three experimental conditions.

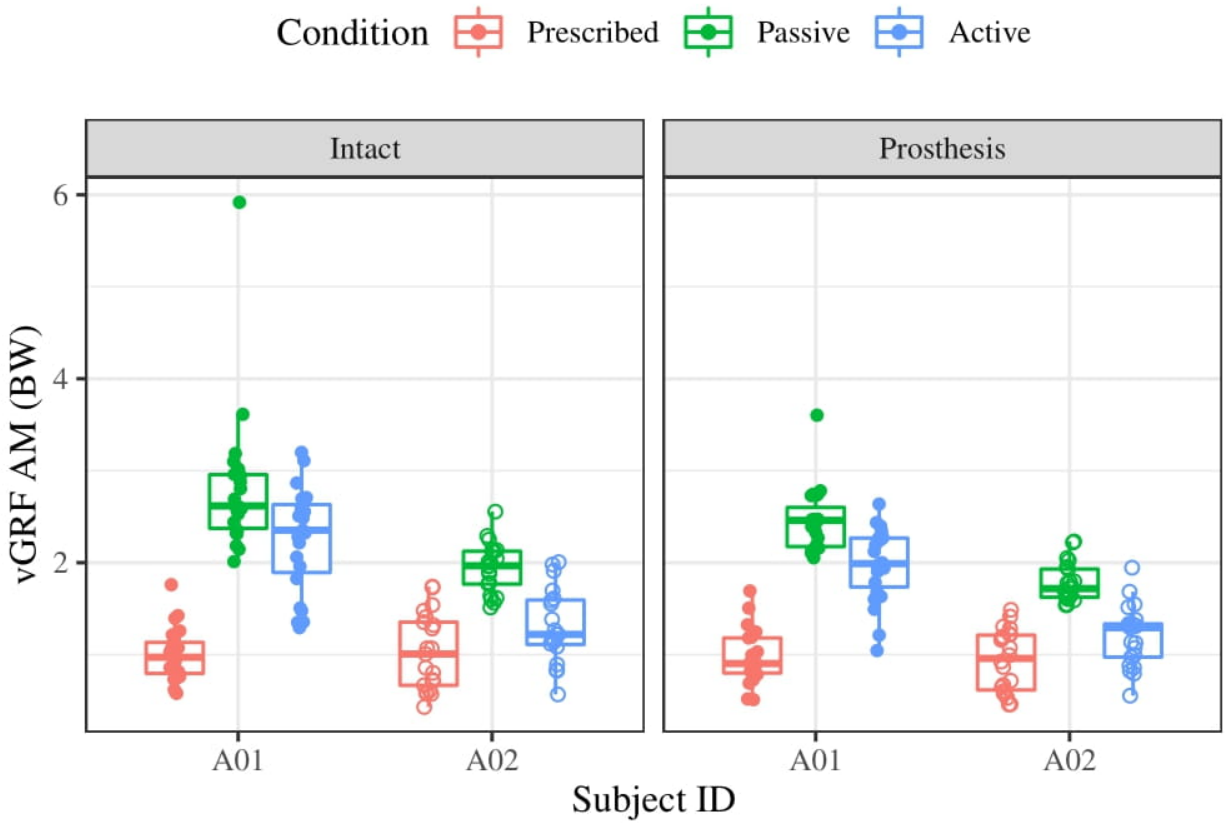


Figure G.31: Vertical ground reaction force asymmetry measurement outcomes for each subject and the three experimental conditions.

Appendix H

ONE-SAMPLE AHEAD TIME SERIES PREDICTION RESULTS

This appendix contains the full time series and test results for one-sample ahead model predictions. The four models and training procedures are discussed in Chapter 6.

Figure H.1: Prediction error histogram for all DNN models across all test time series data.

H.1 Feedforward Neural Network

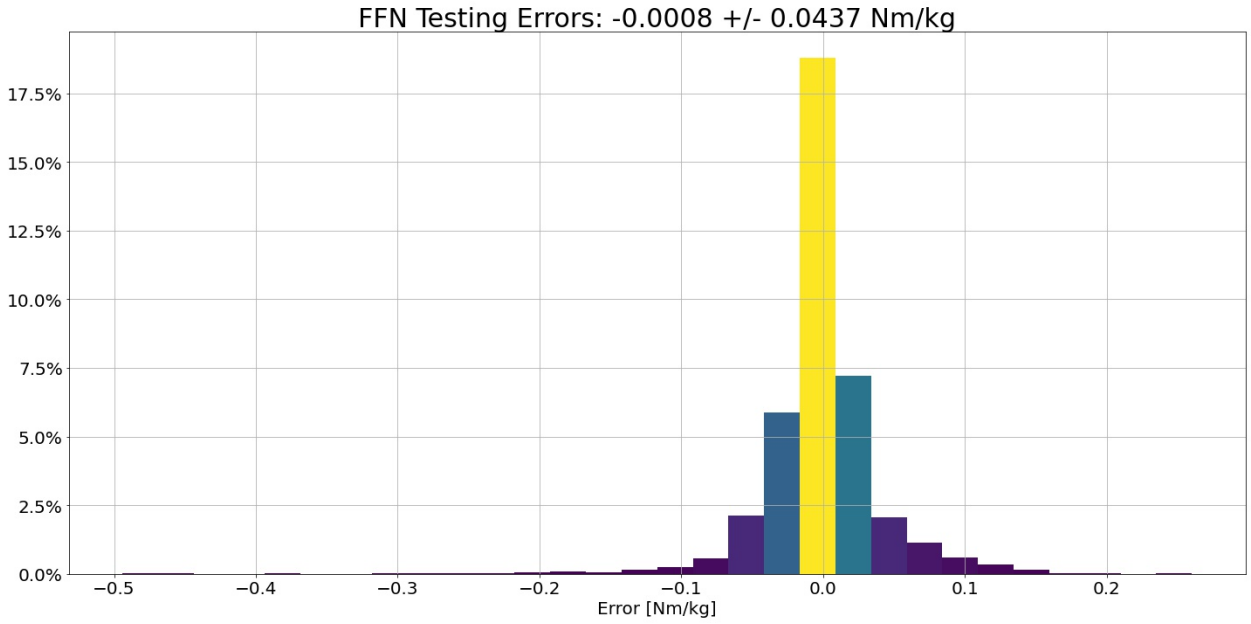


Figure H.2: FFN prediction error histogram for all test time series data.

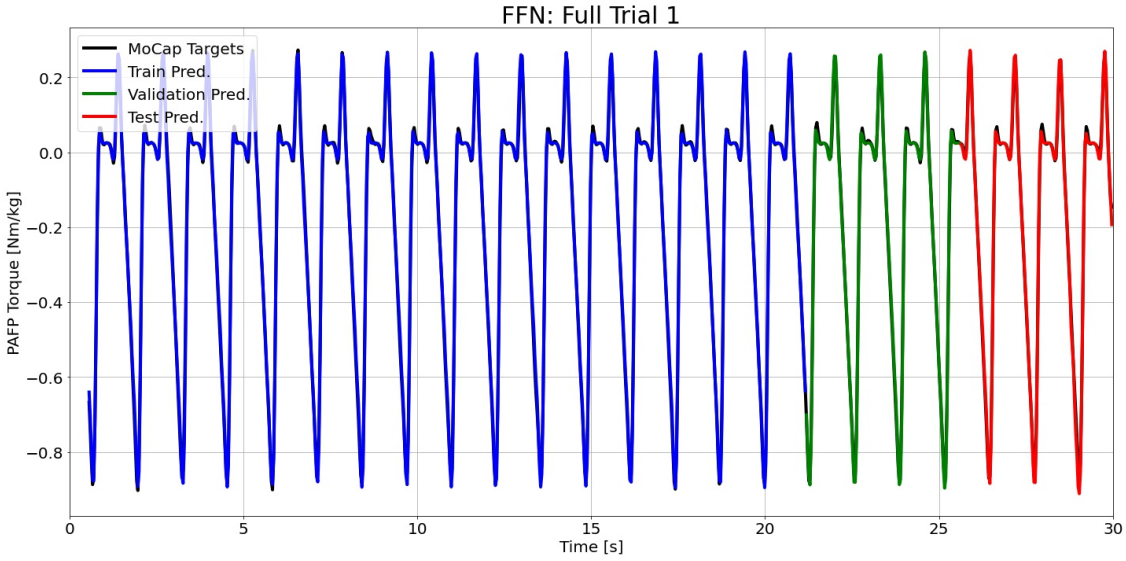


Figure H.3: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 1.

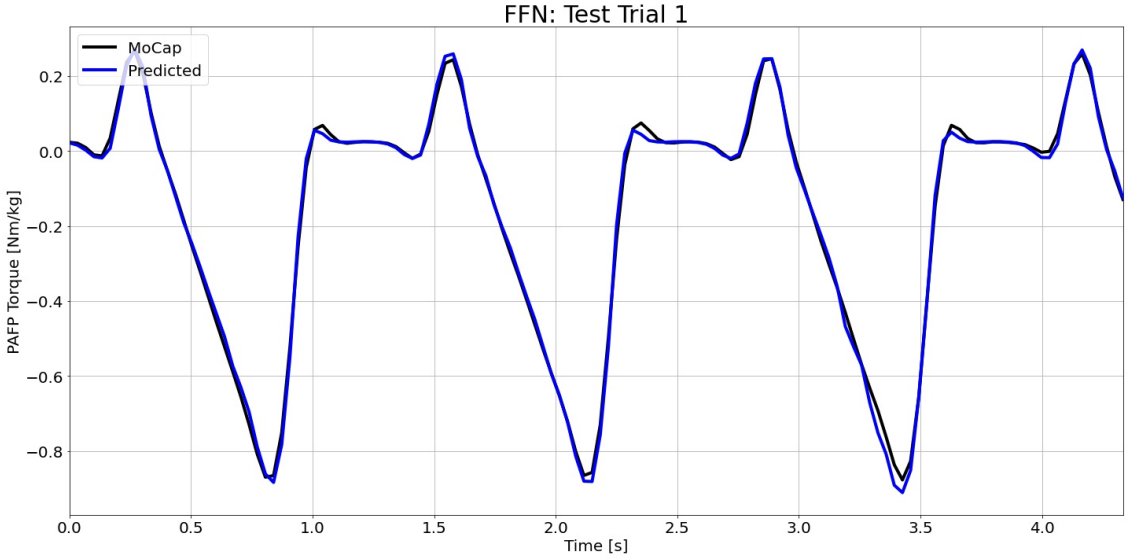


Figure H.4: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 1.

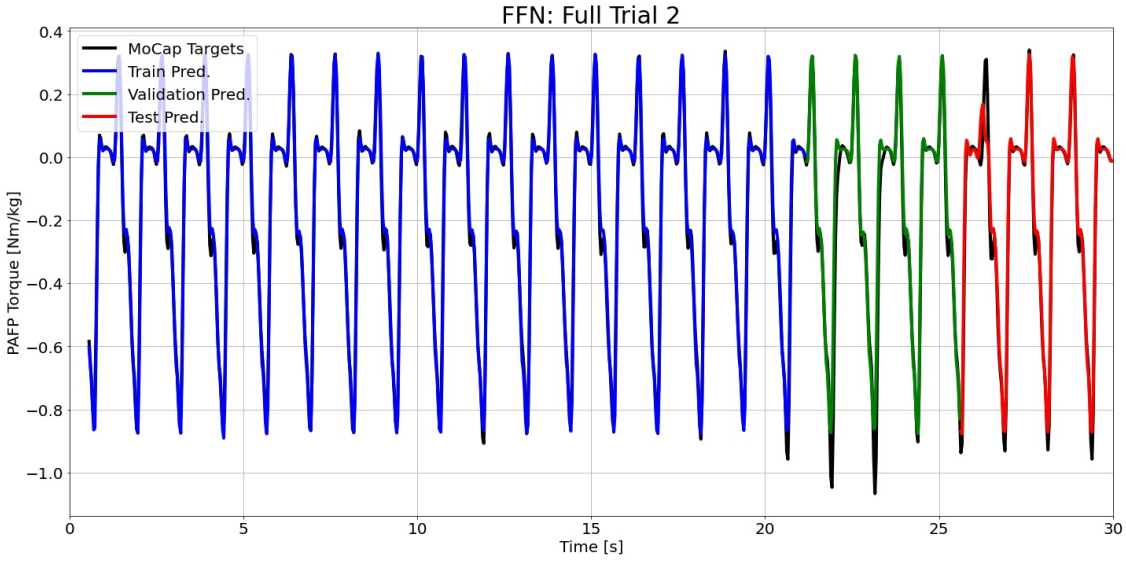


Figure H.5: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 2.

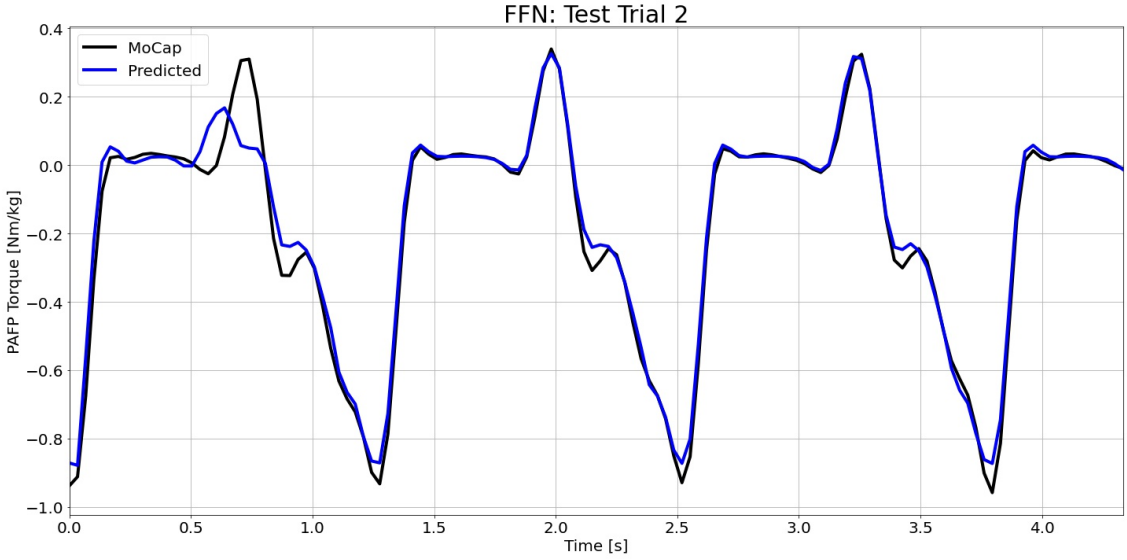


Figure H.6: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 2.

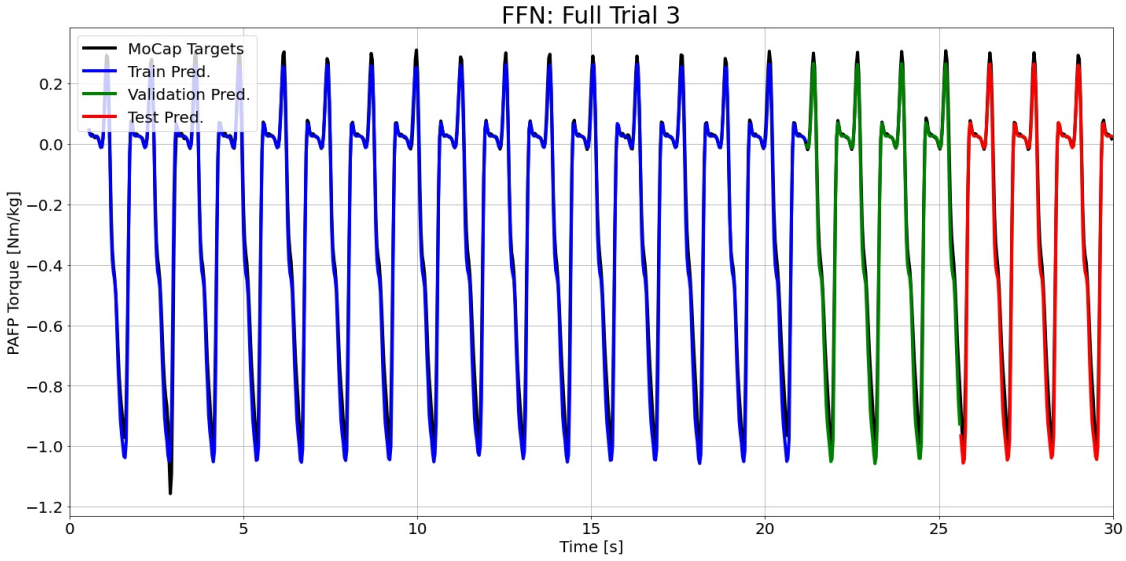


Figure H.7: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 3.

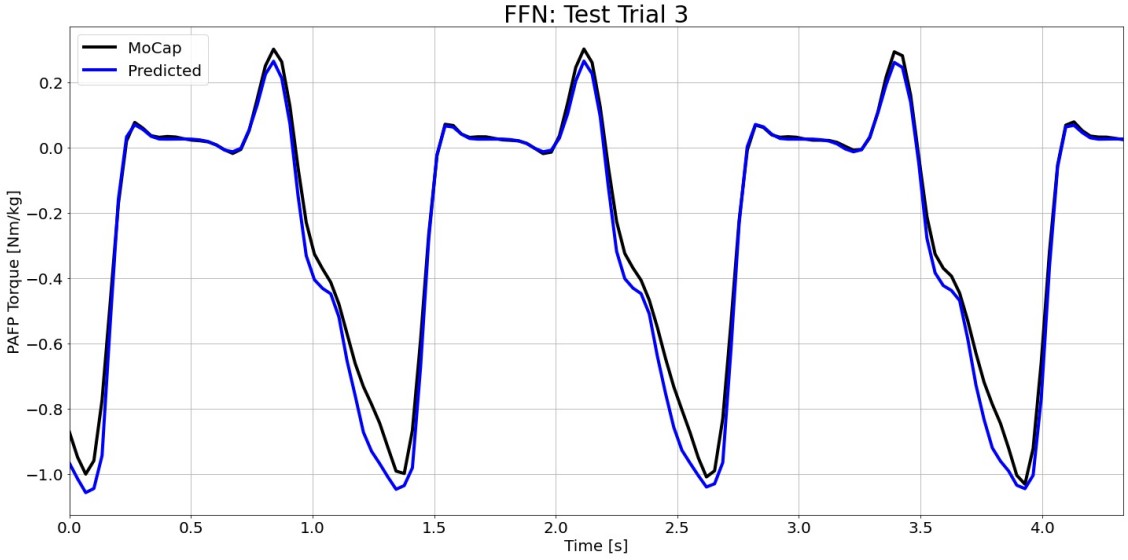


Figure H.8: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 3.

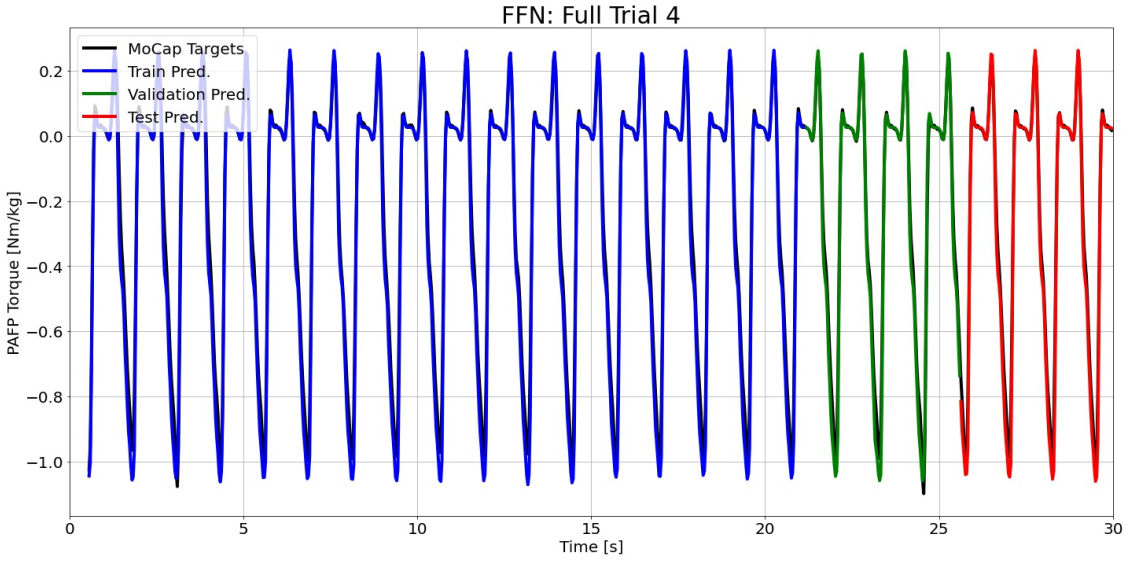


Figure H.9: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 4.

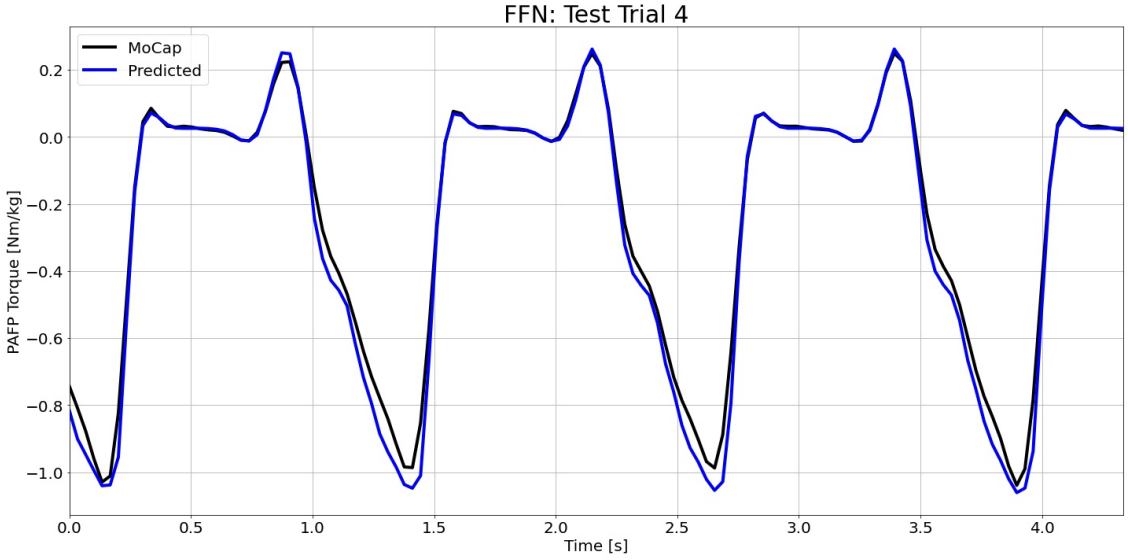


Figure H.10: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 4.

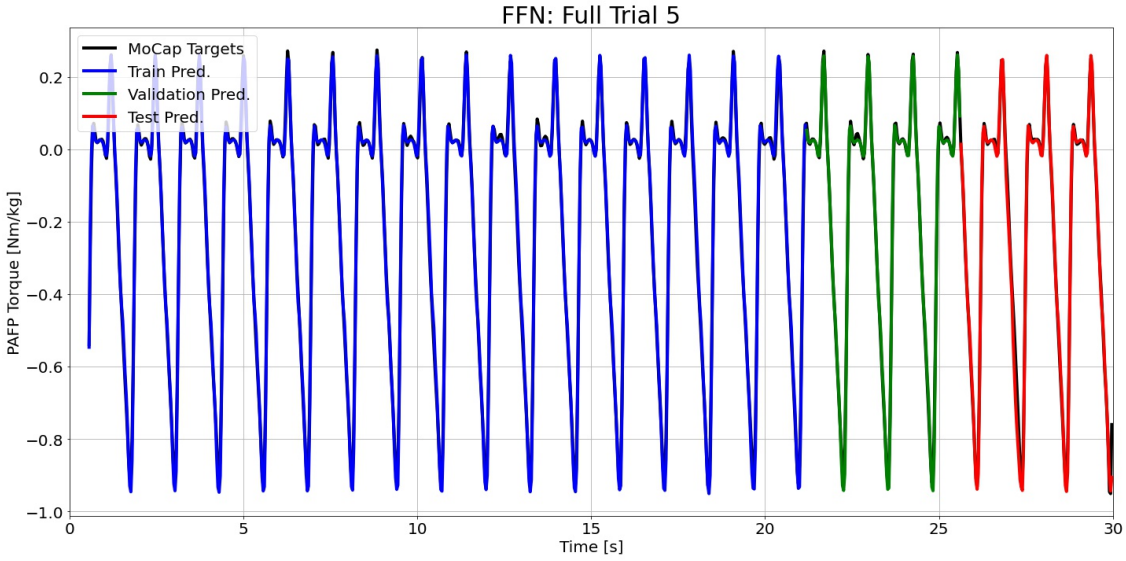


Figure H.11: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 5.

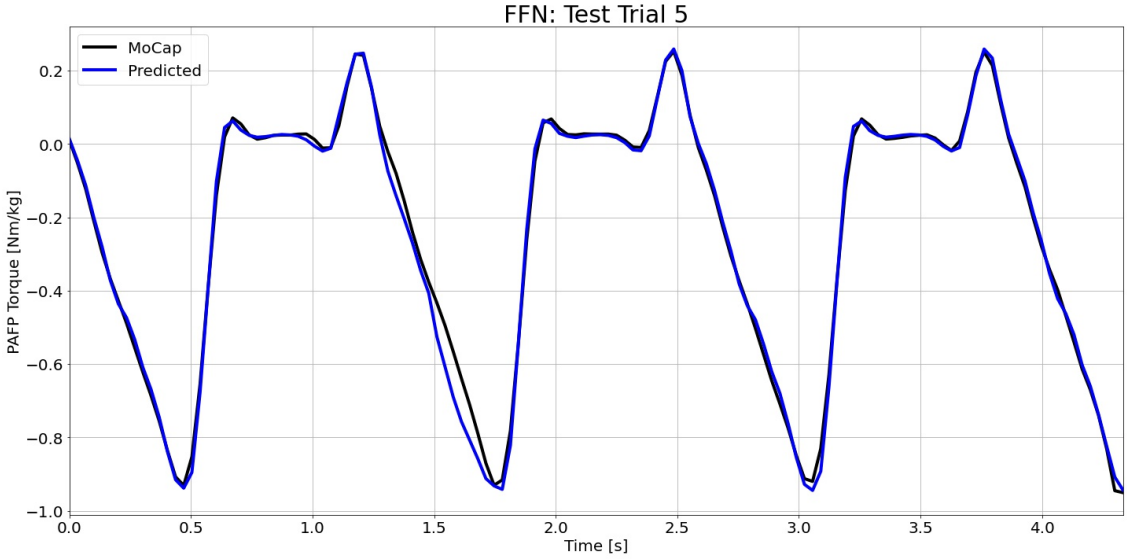


Figure H.12: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 5.

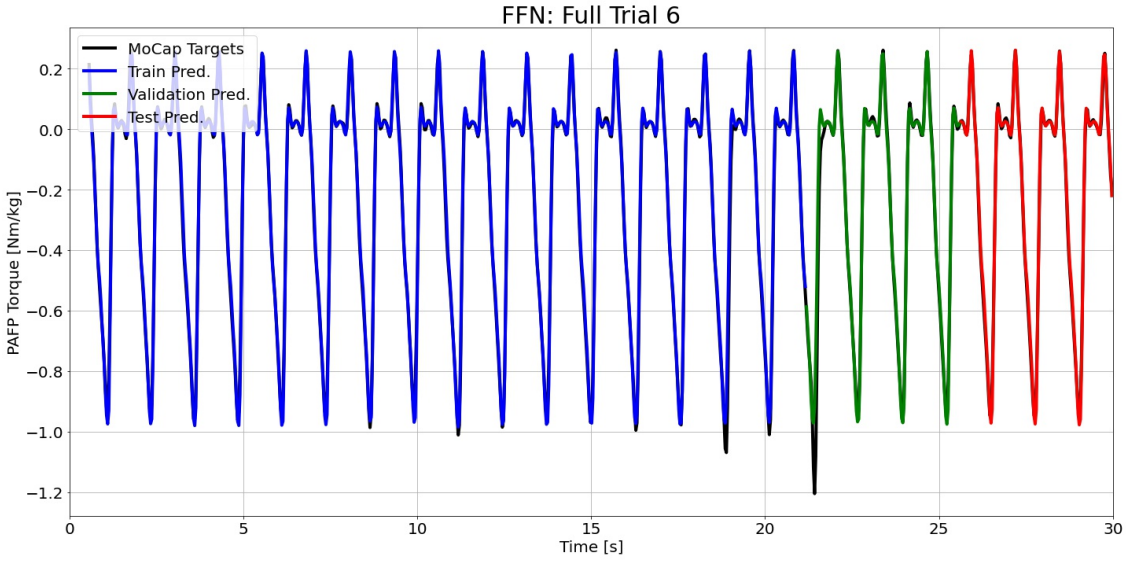


Figure H.13: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 6.

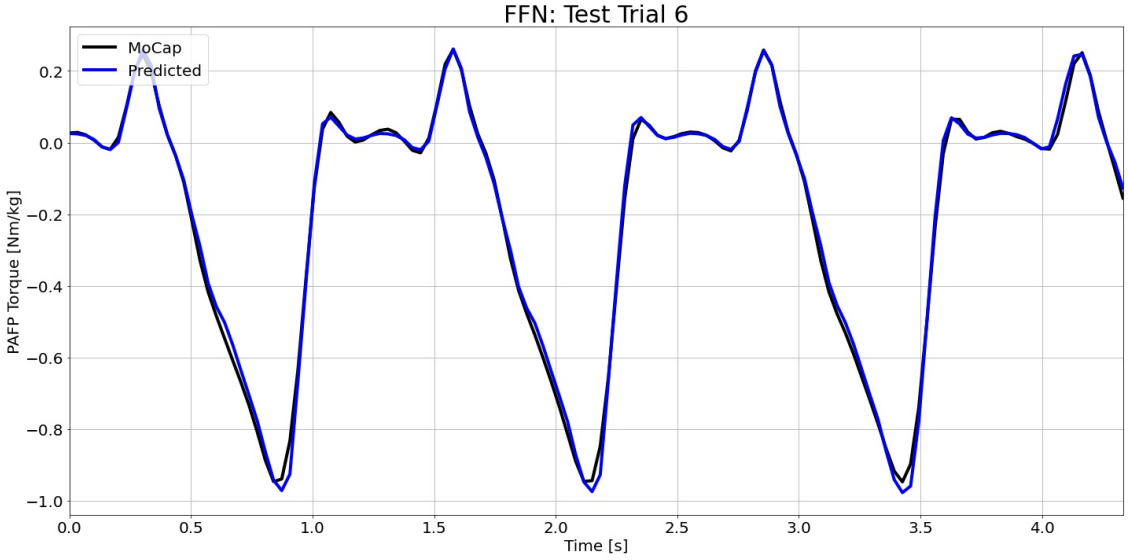


Figure H.14: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 6.

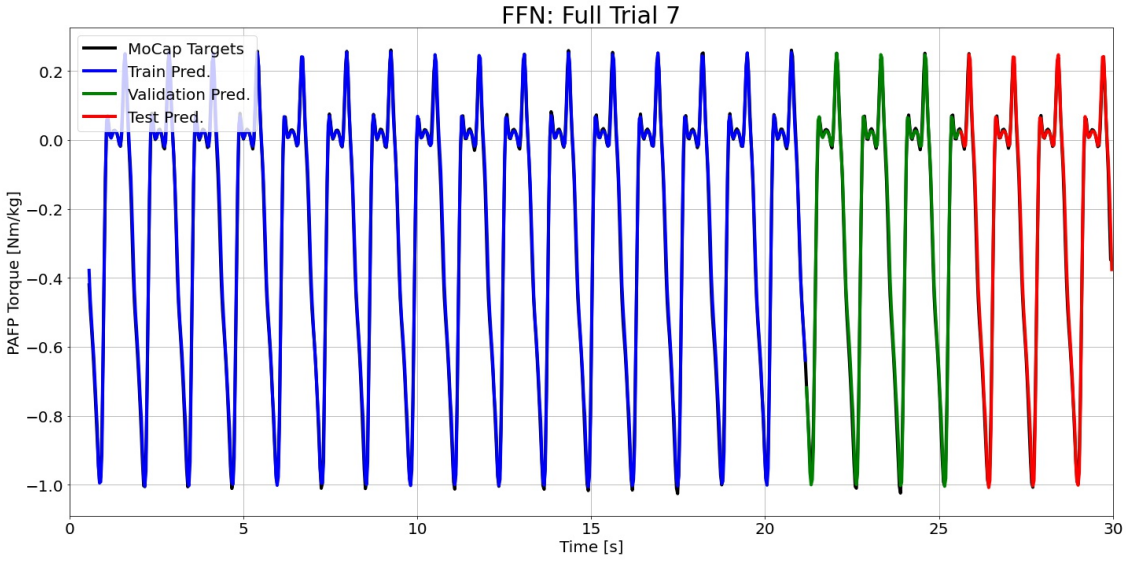


Figure H.15: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 7.

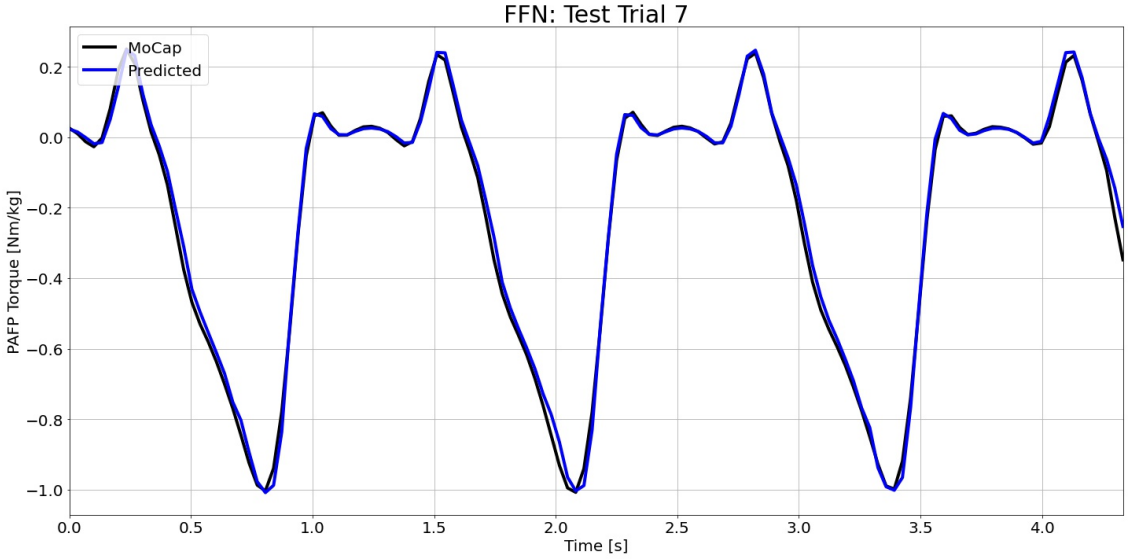


Figure H.16: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 7.

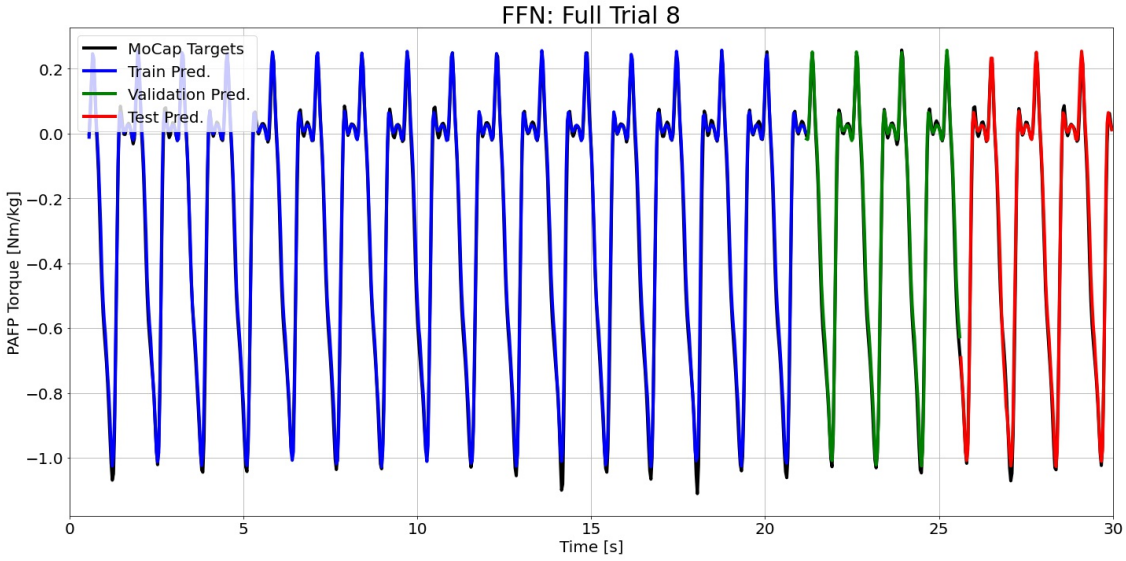


Figure H.17: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 8.

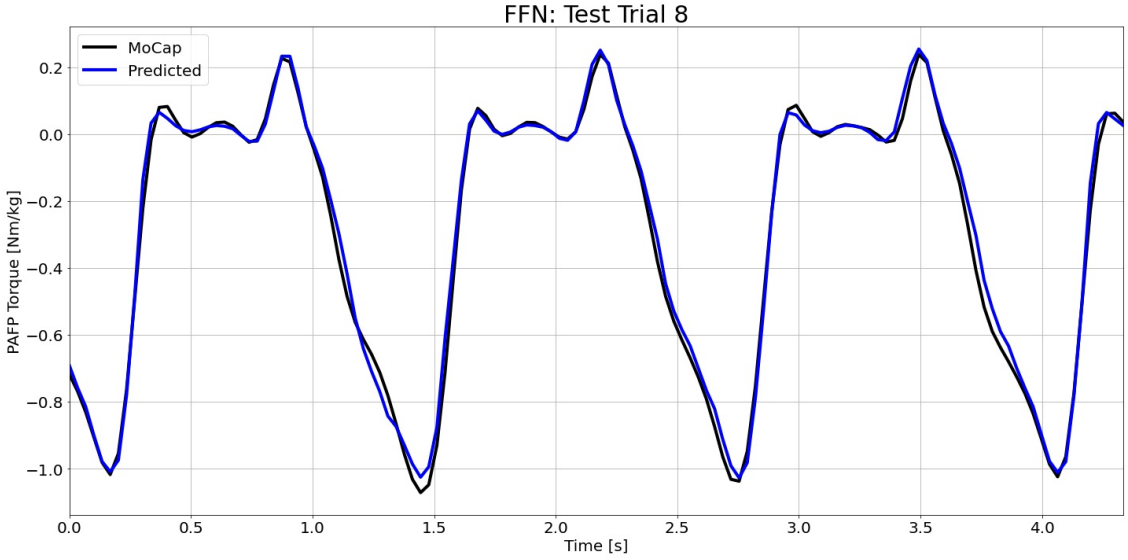


Figure H.18: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 8.

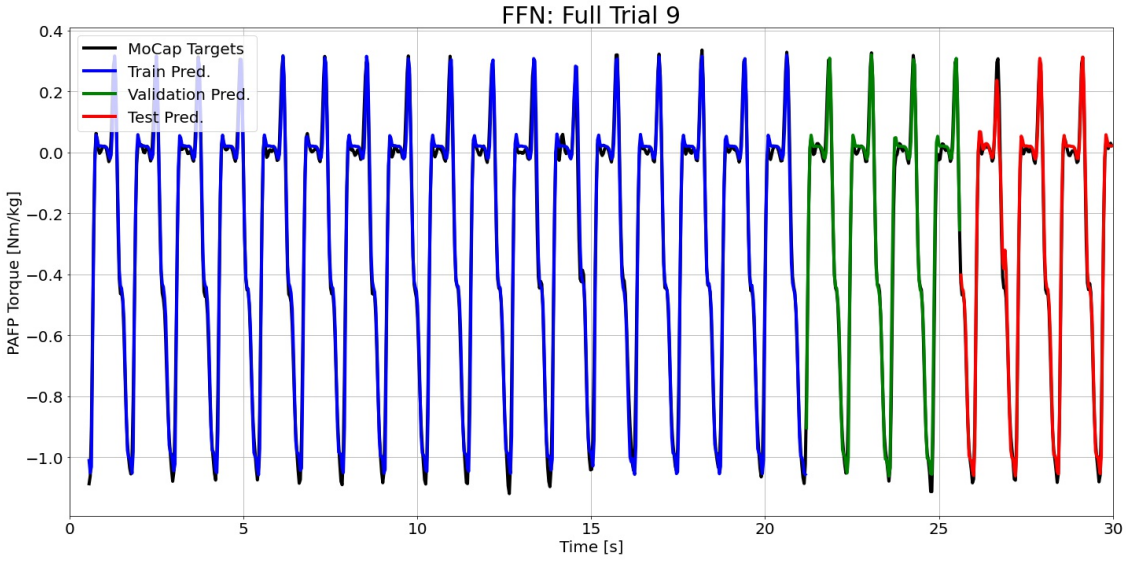


Figure H.19: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 9.

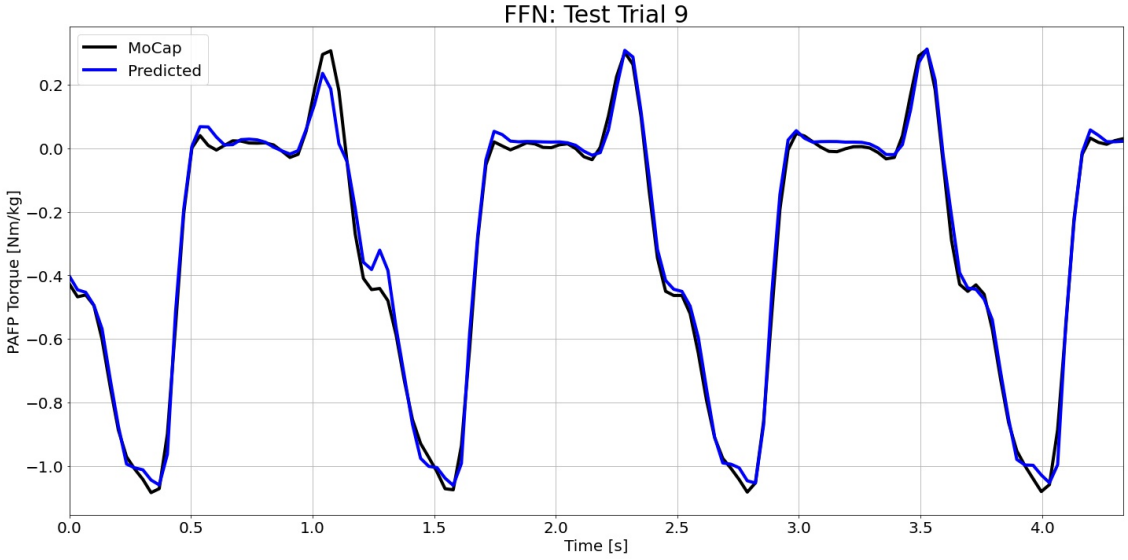


Figure H.20: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 9.

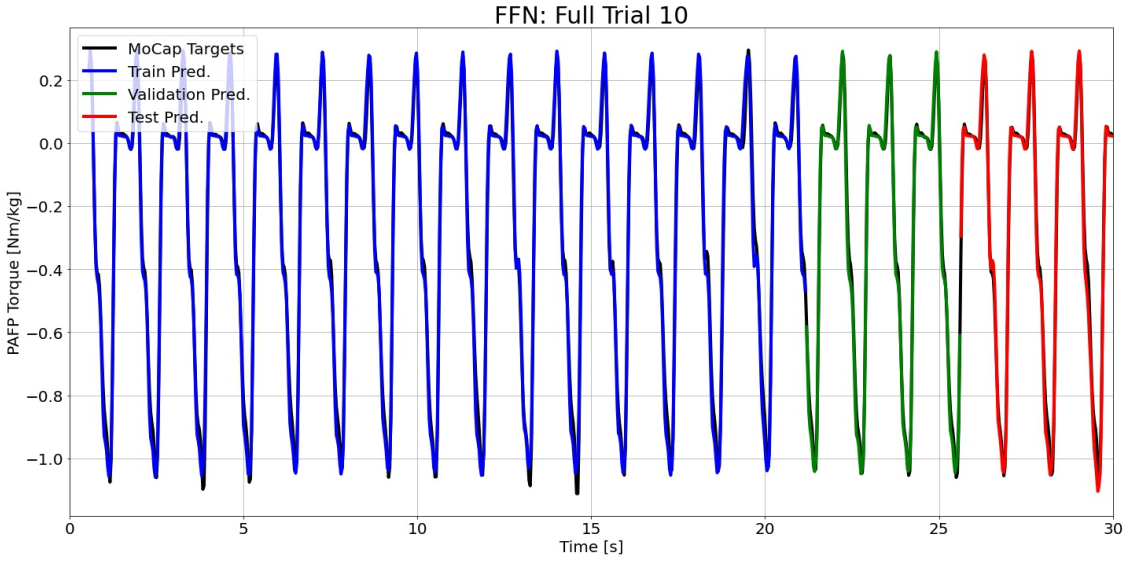


Figure H.21: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 10.

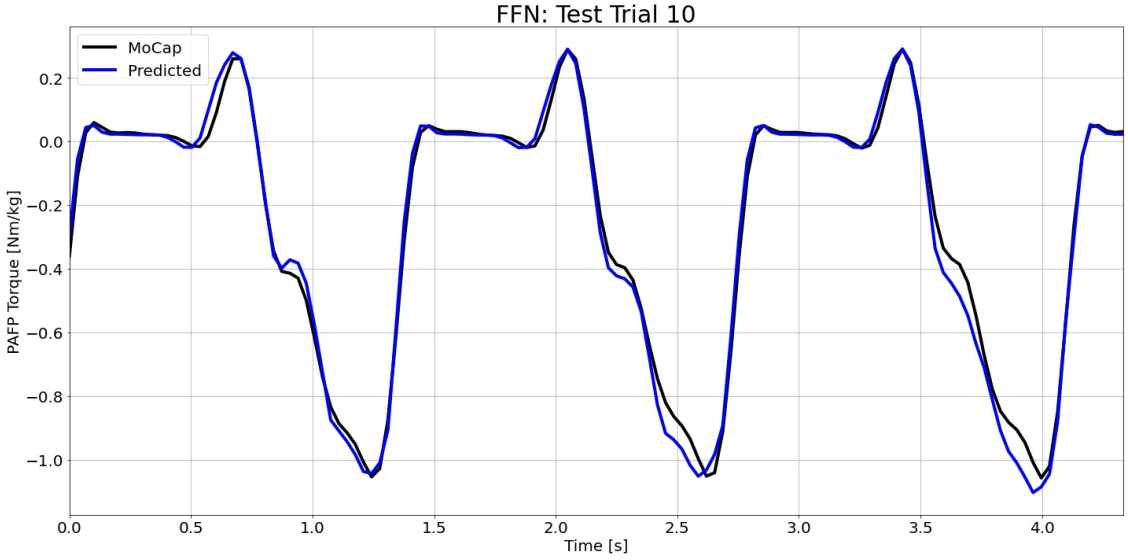


Figure H.22: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 10.

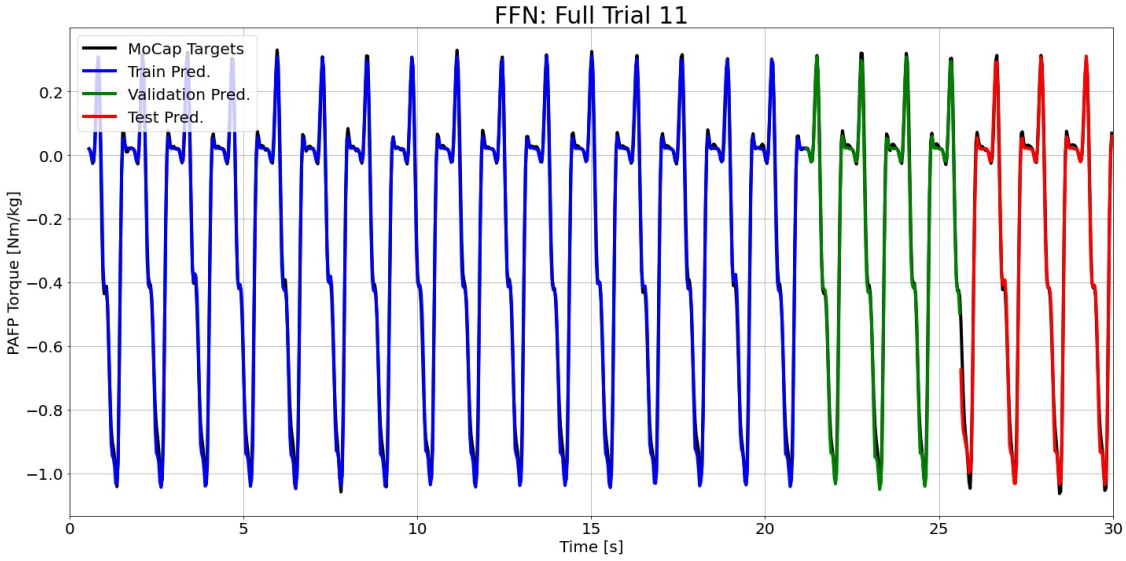


Figure H.23: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 11.

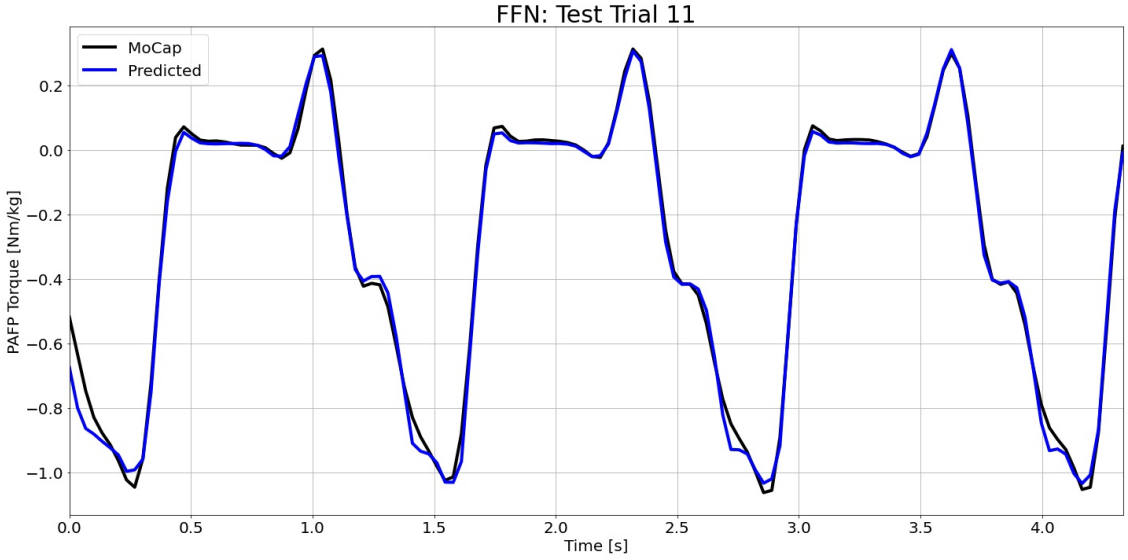


Figure H.24: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 11.

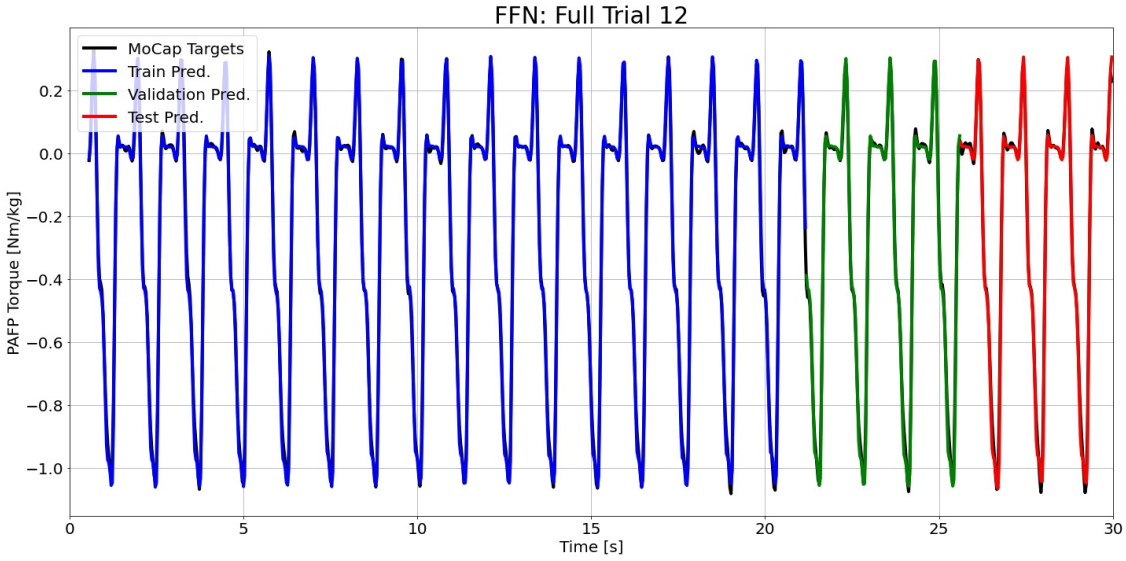


Figure H.25: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 12.

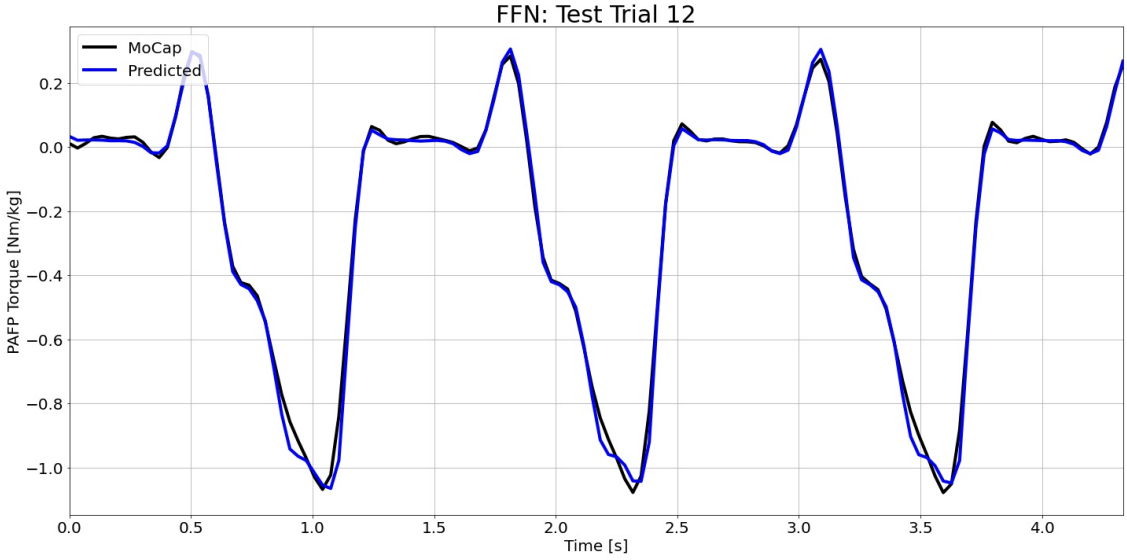


Figure H.26: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 12.

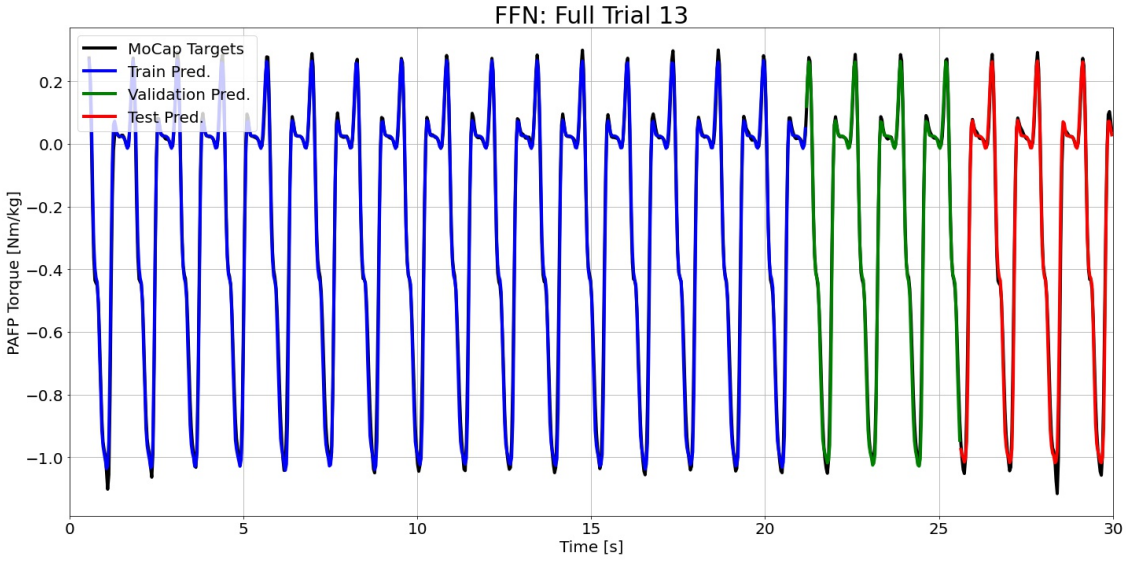


Figure H.27: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 13.

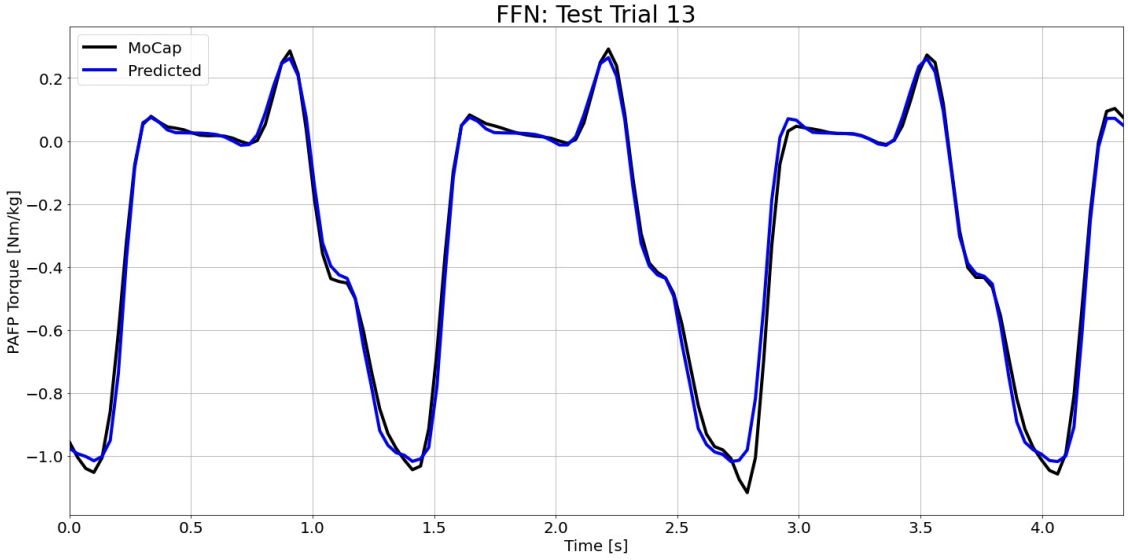


Figure H.28: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 13.

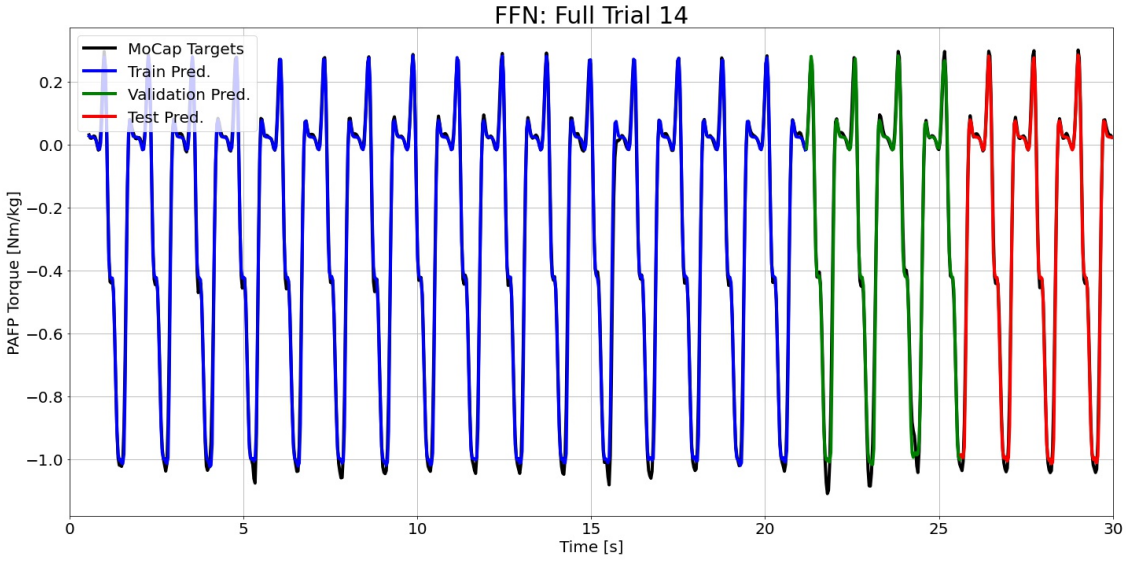


Figure H.29: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 14.

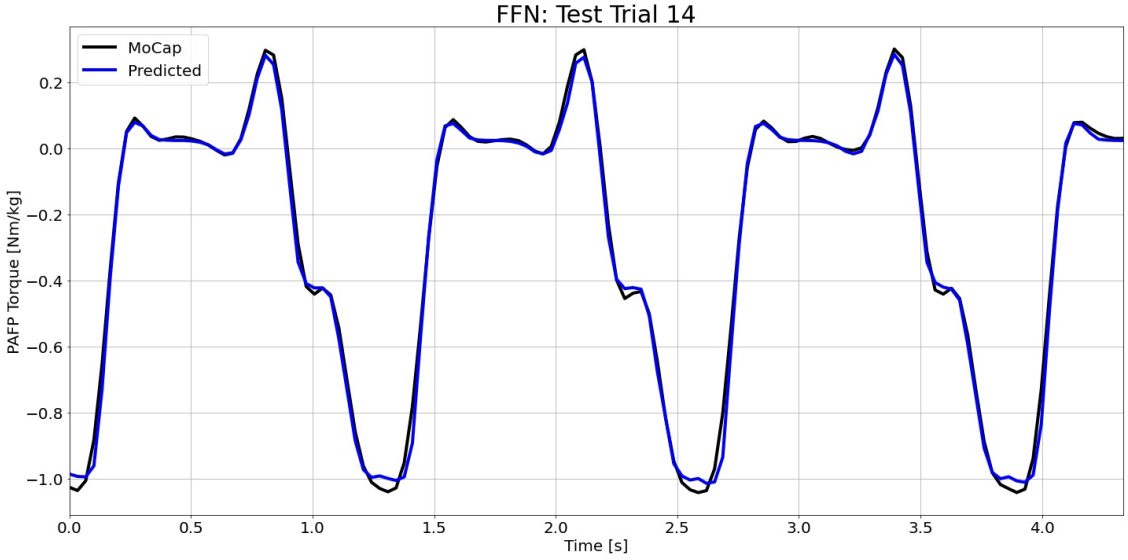


Figure H.30: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 14.

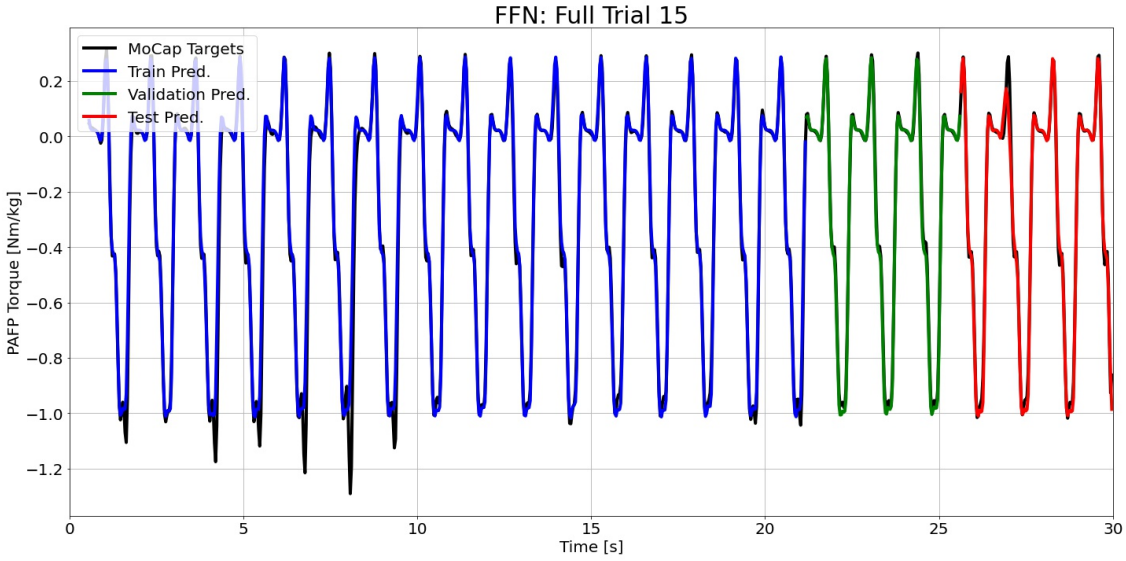


Figure H.31: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 15.

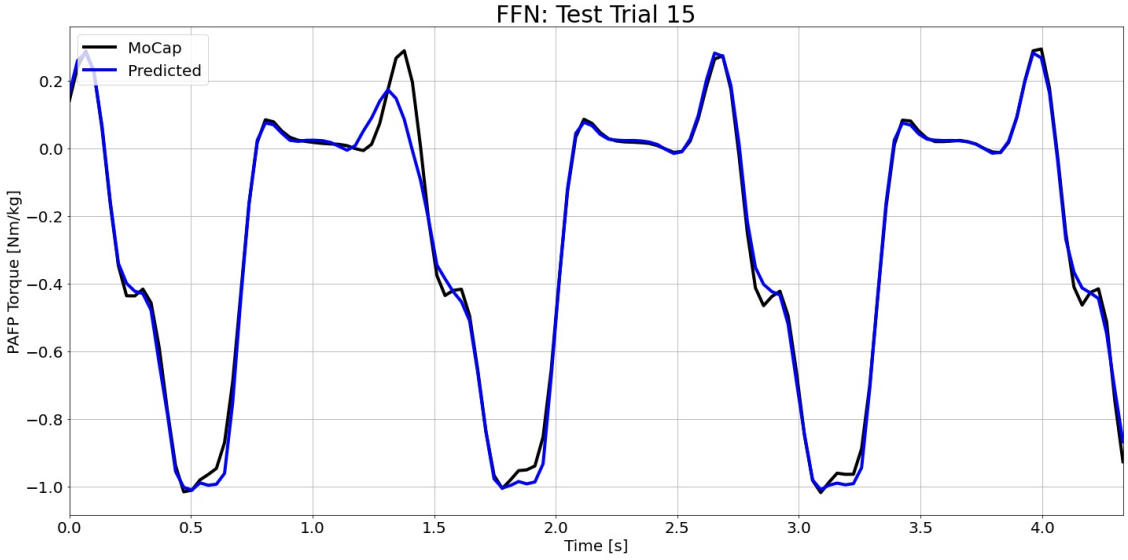


Figure H.32: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 15.

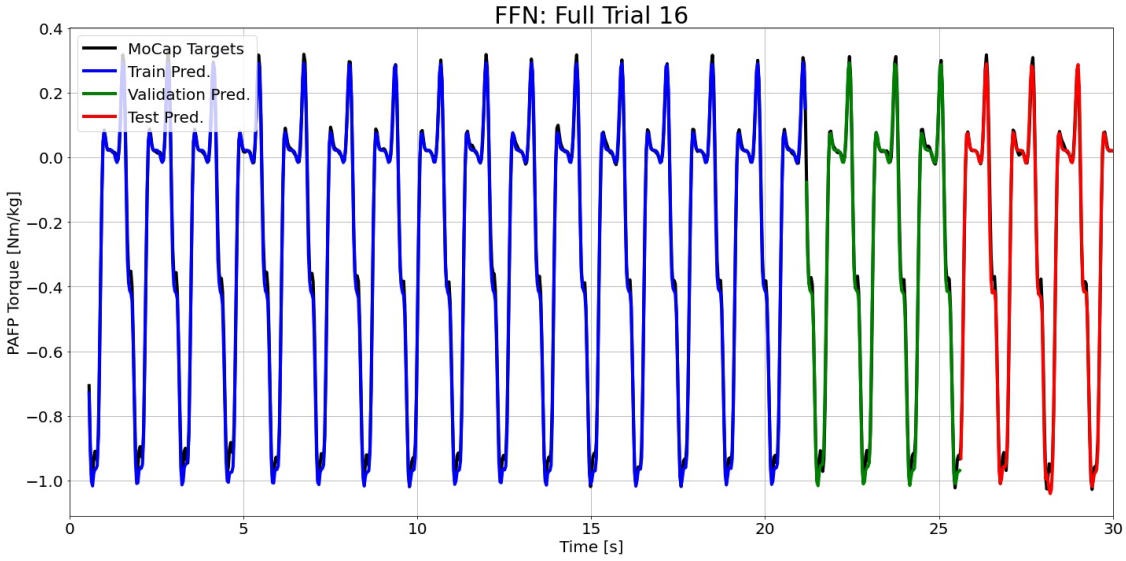


Figure H.33: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 16.

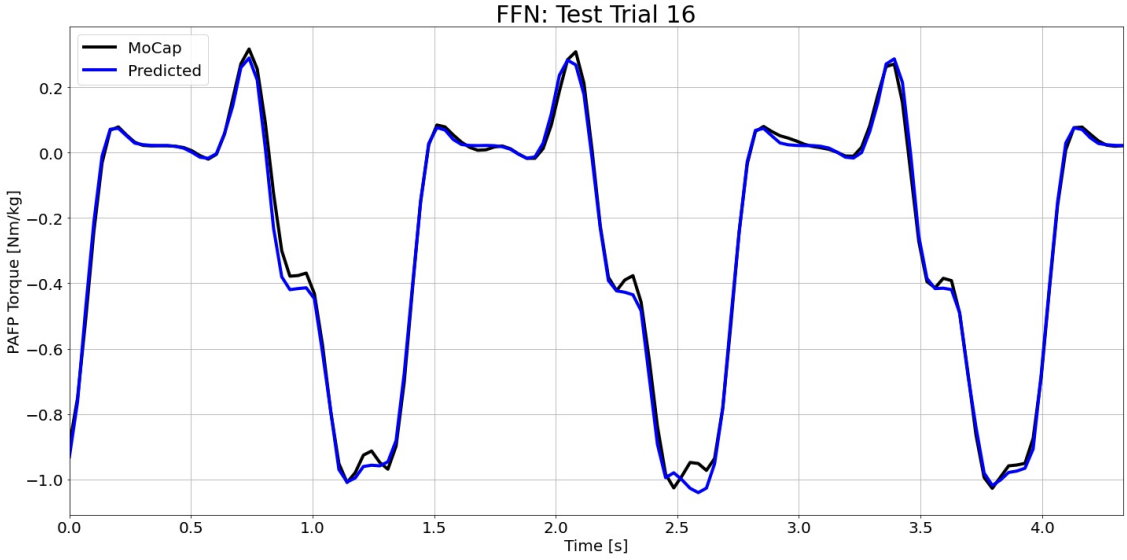


Figure H.34: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 16.

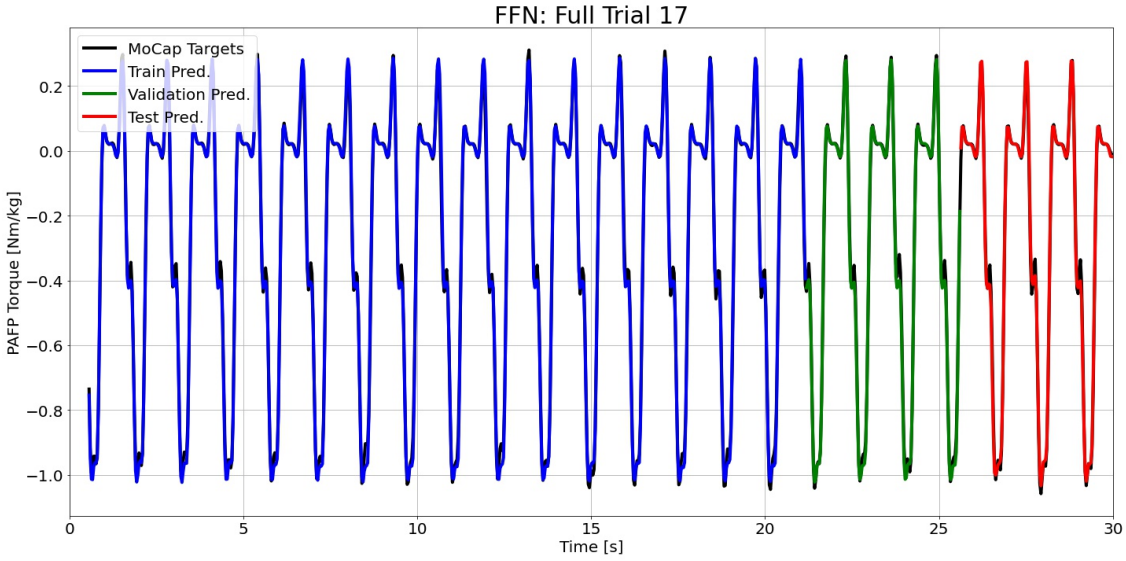


Figure H.35: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 17.

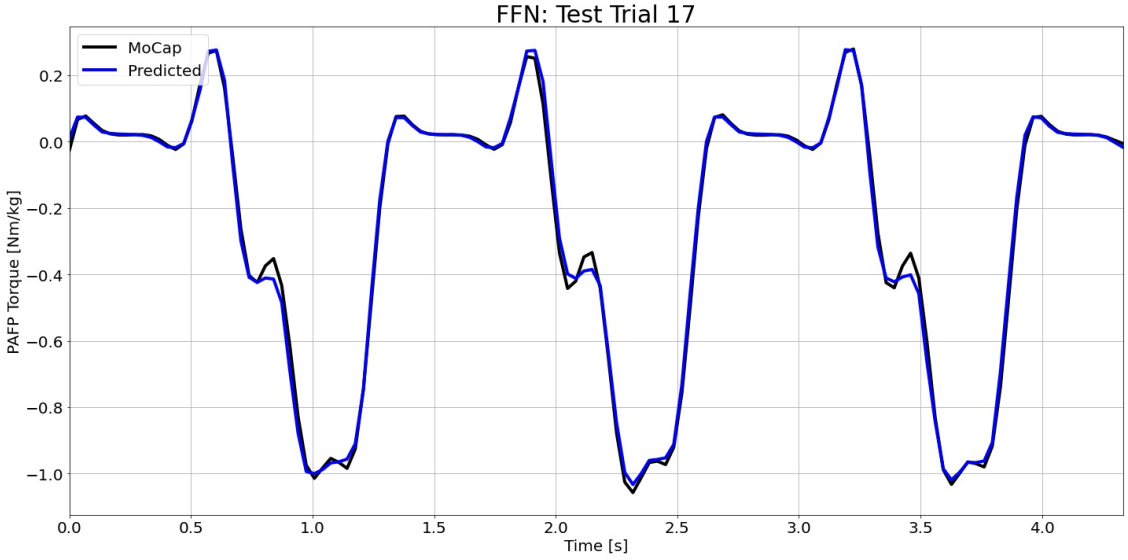


Figure H.36: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 17.

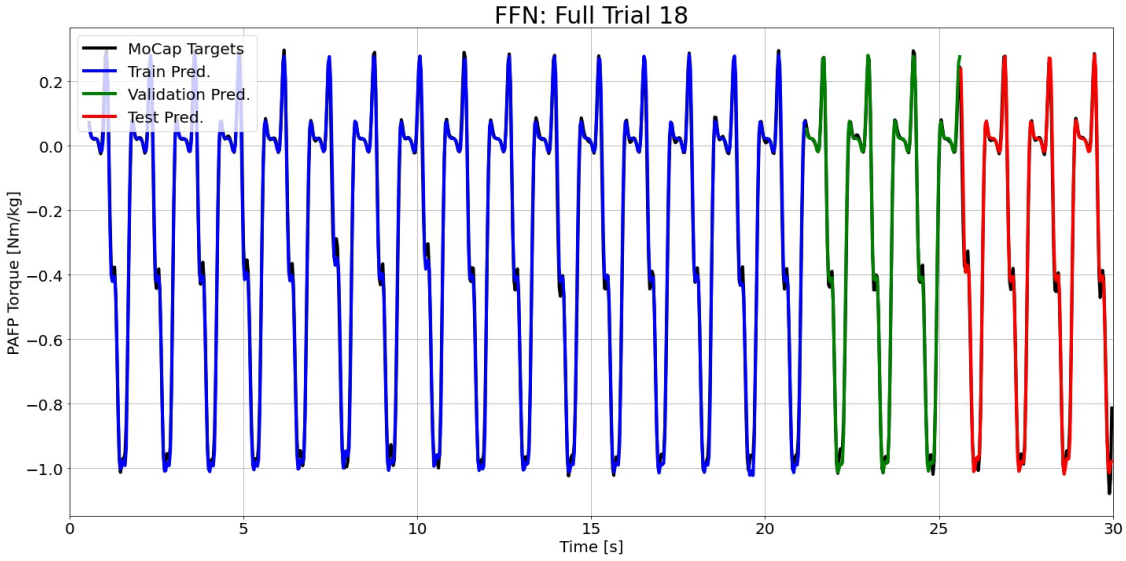


Figure H.37: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 18.

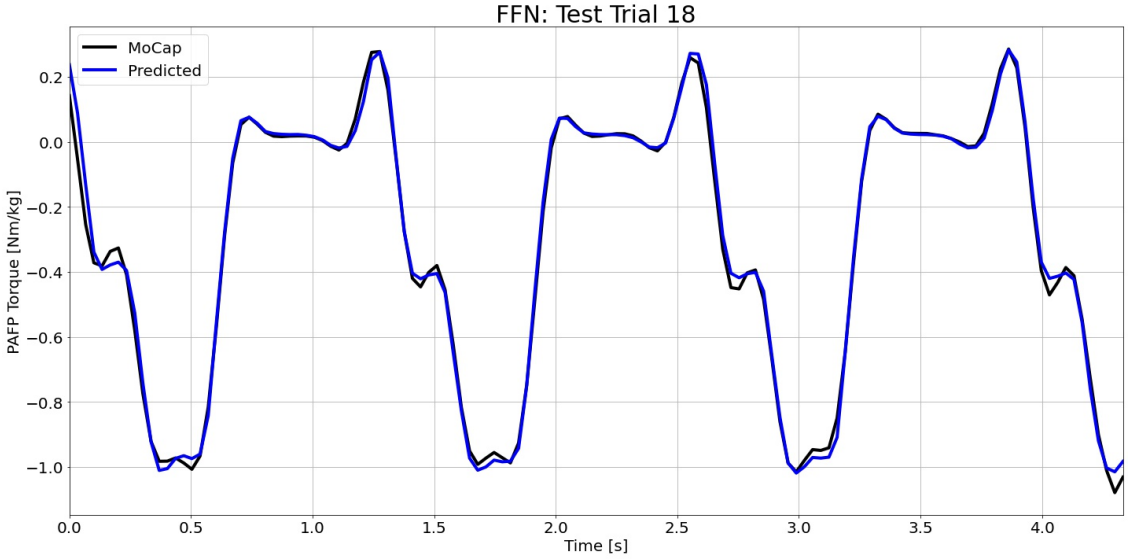


Figure H.38: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 18.

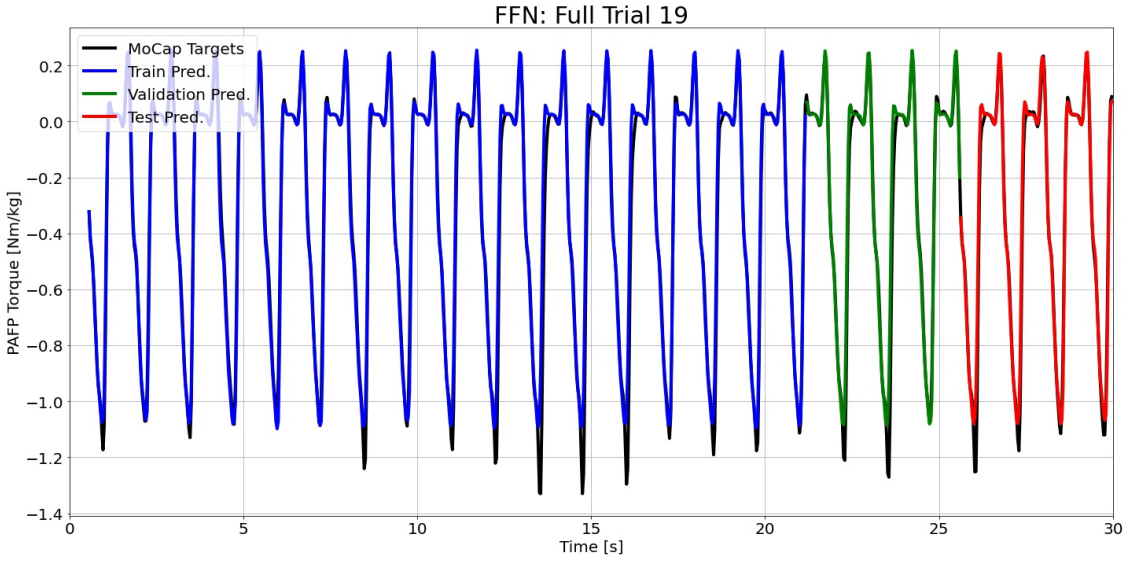


Figure H.39: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 19.

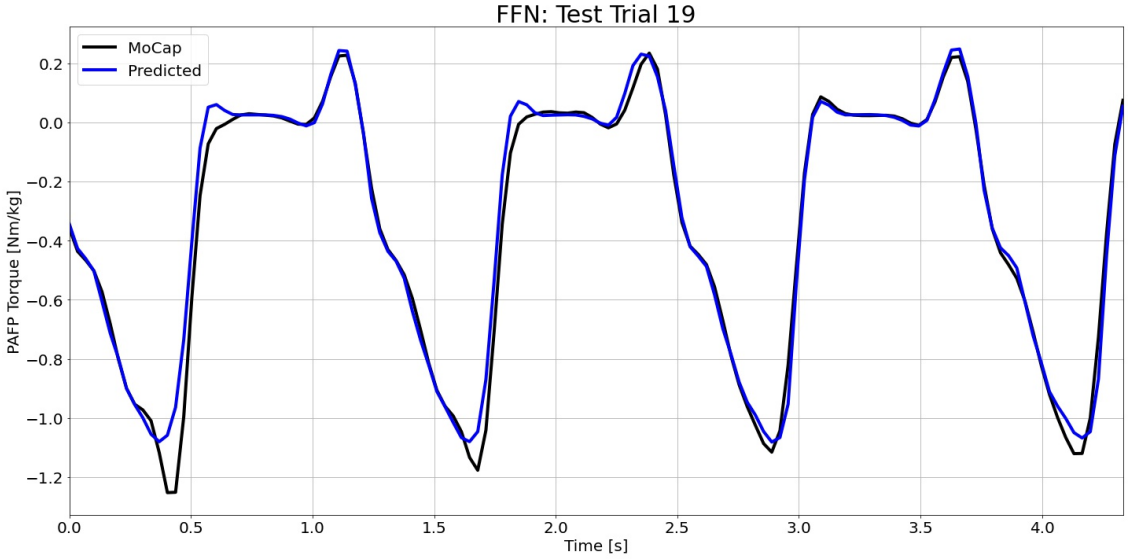


Figure H.40: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 19.

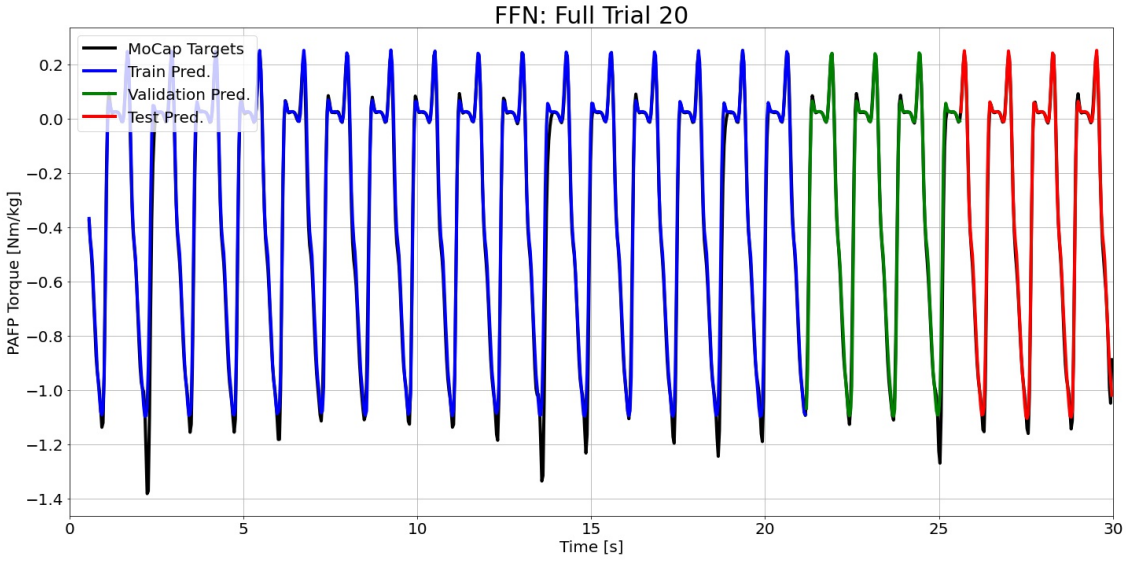


Figure H.41: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 20.

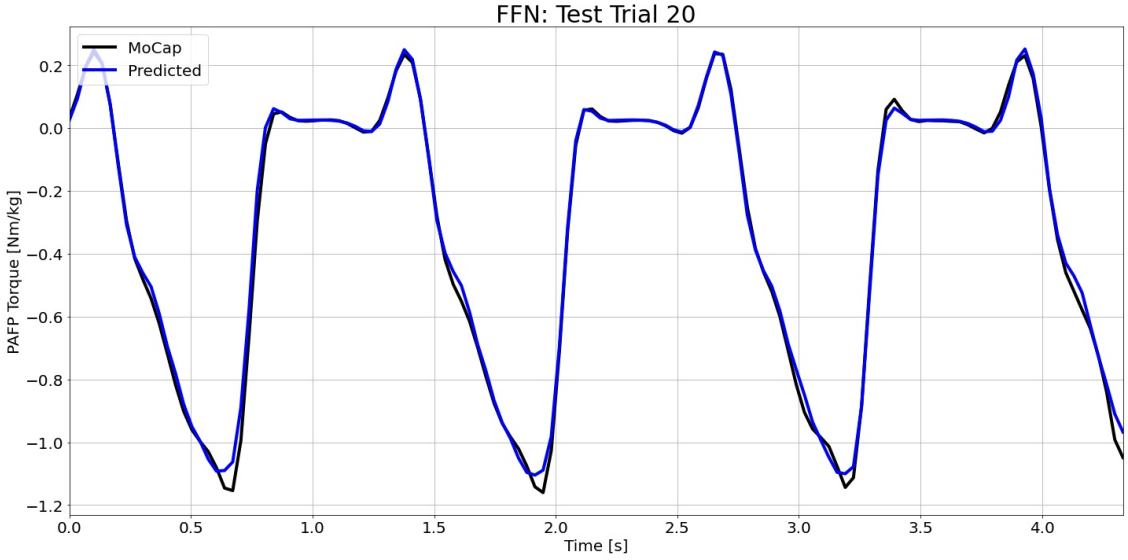


Figure H.42: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 20.

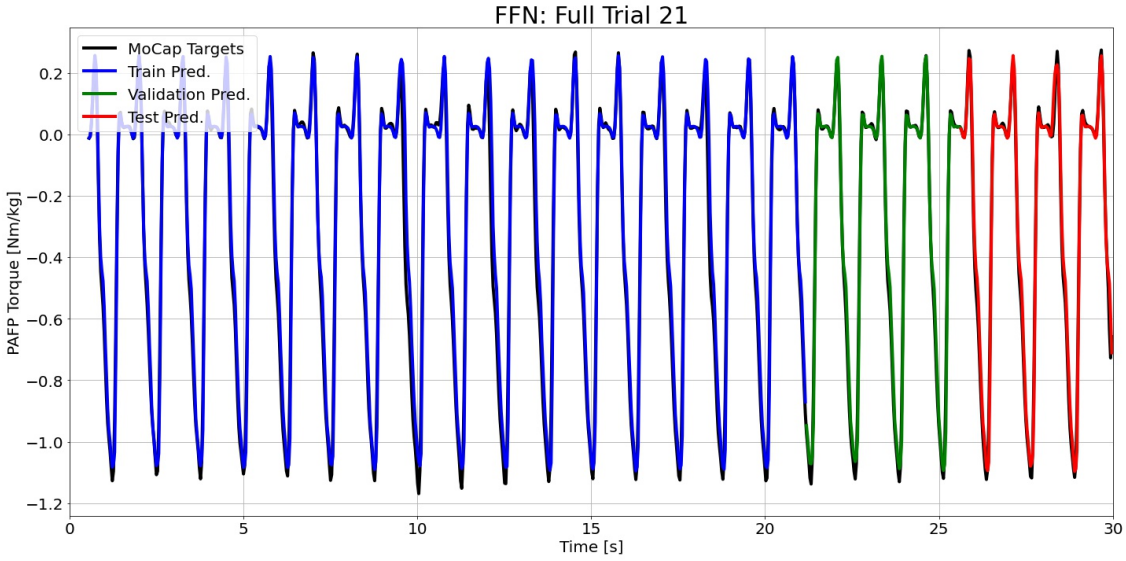


Figure H.43: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 21.

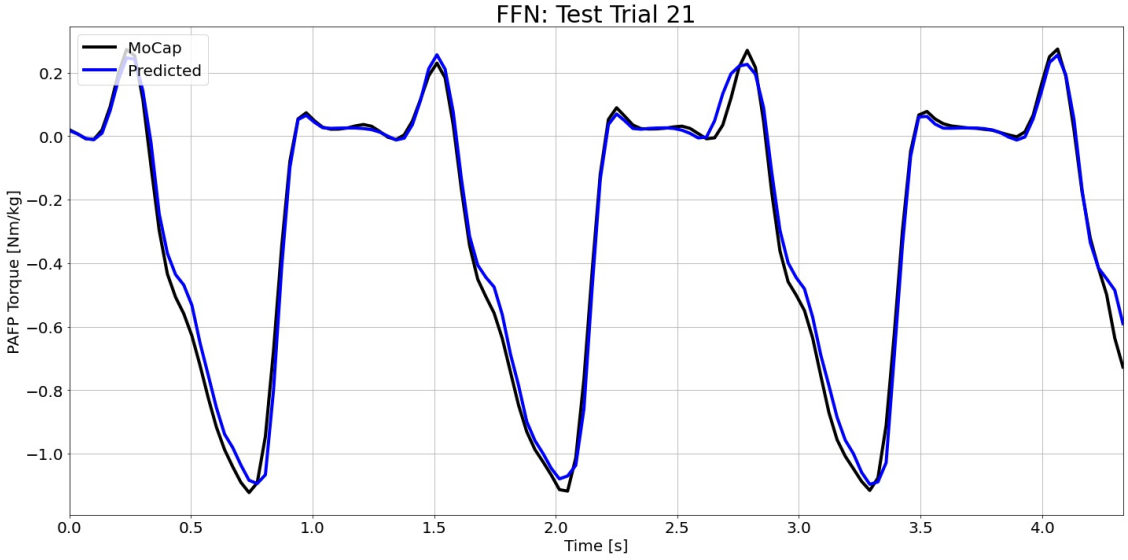


Figure H.44: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 21.

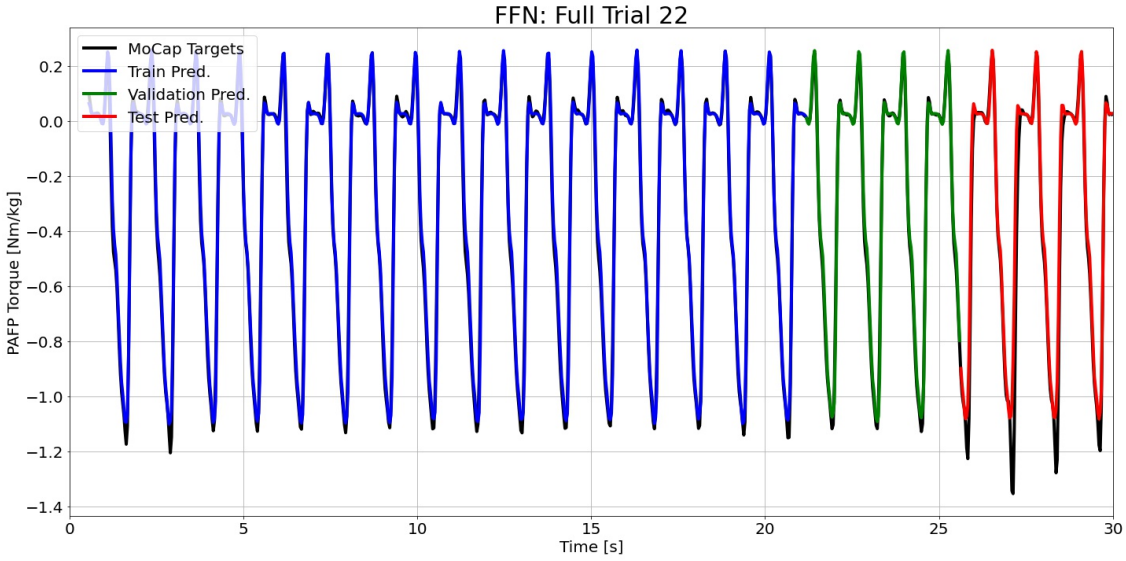


Figure H.45: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 22.

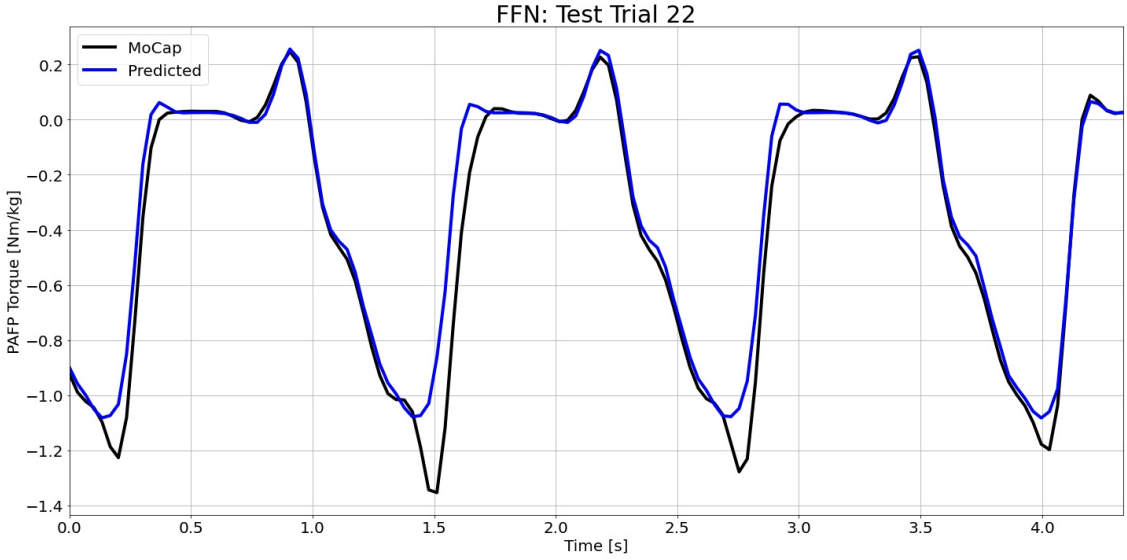


Figure H.46: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 22.

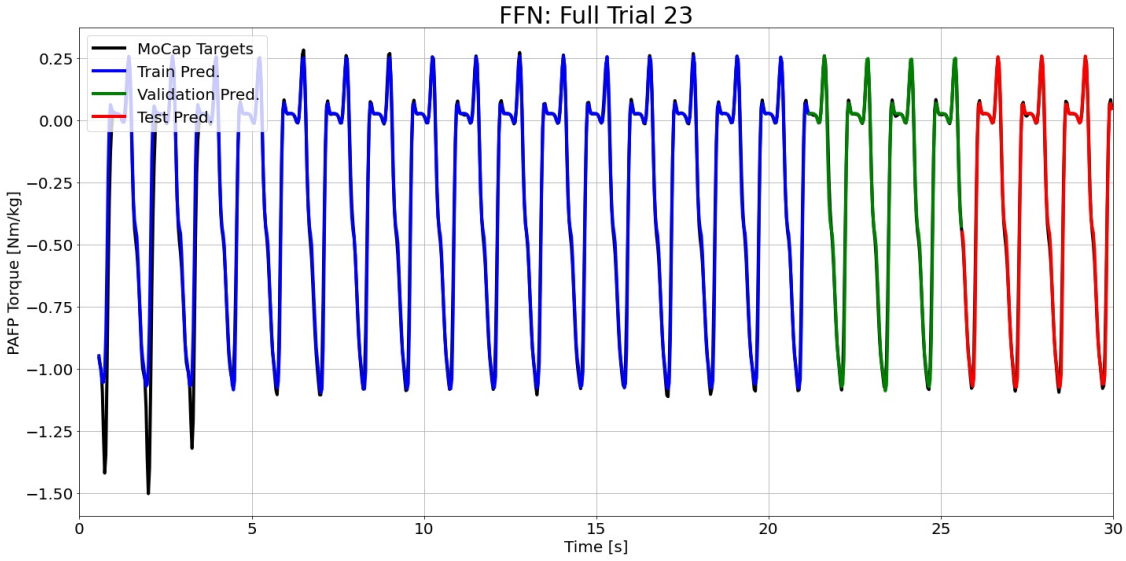


Figure H.47: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 23.

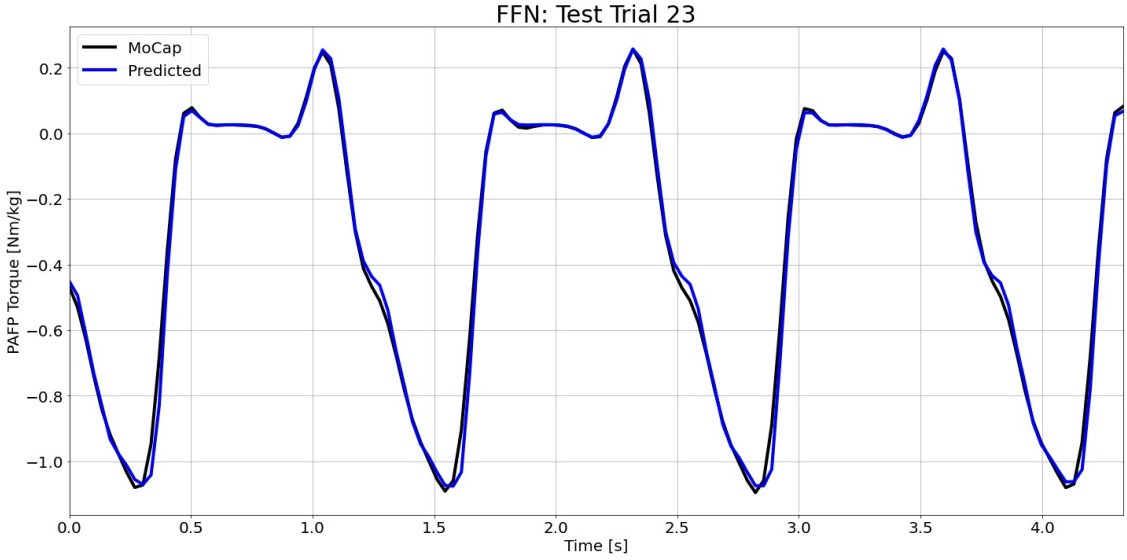


Figure H.48: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 23.

H.2 Gated Recurrent Unit Neural Network

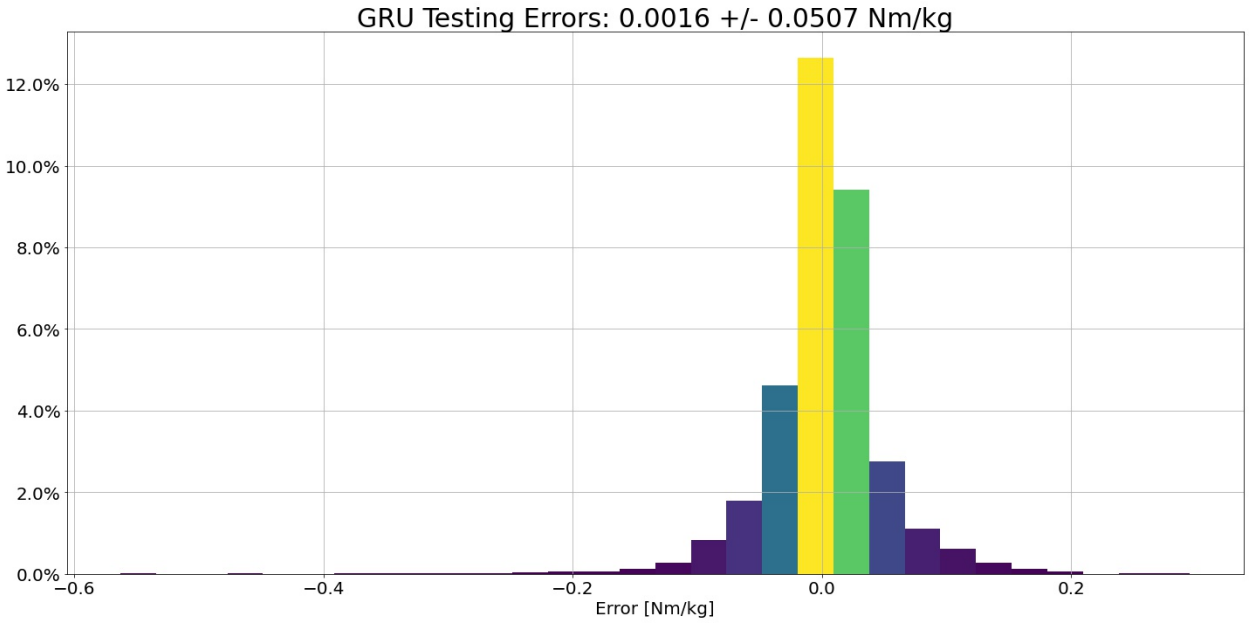


Figure H.49: GRU prediction error histogram for all test time series data.

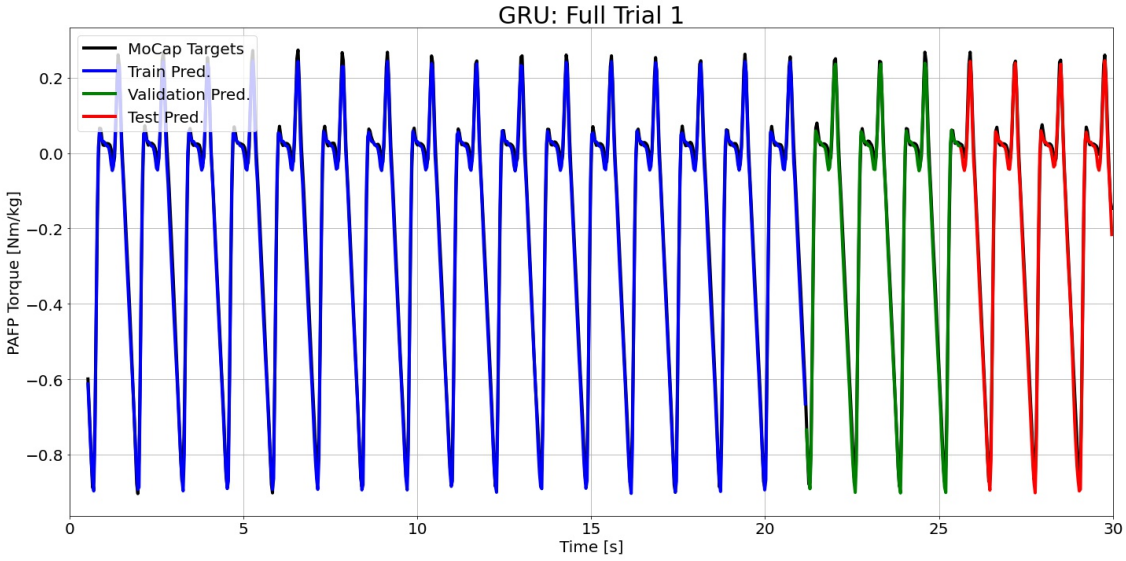


Figure H.50: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 1.

Figure H.51: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 1.

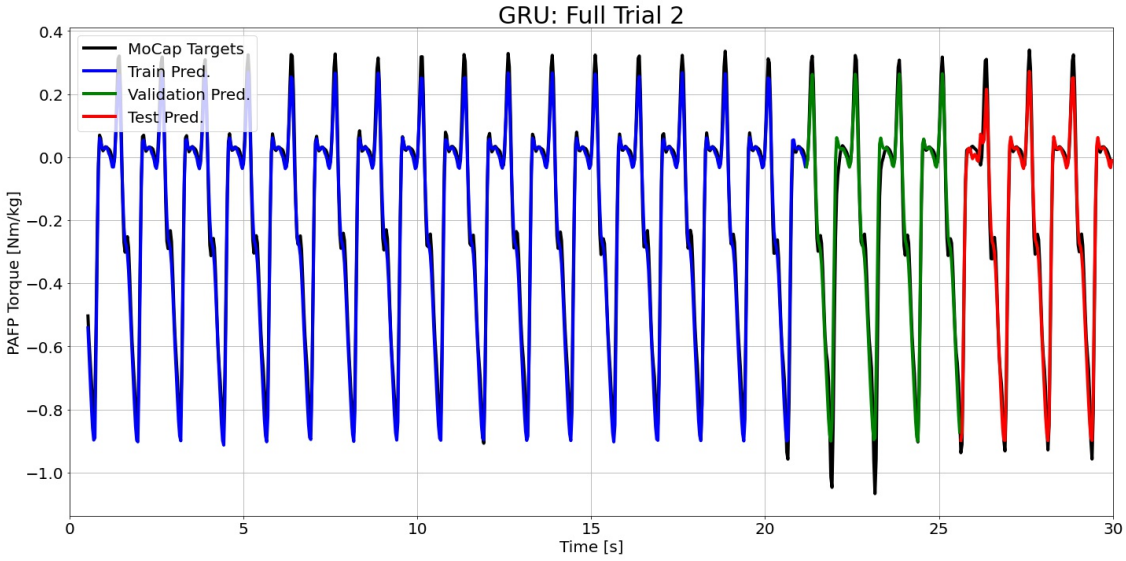


Figure H.52: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 2.

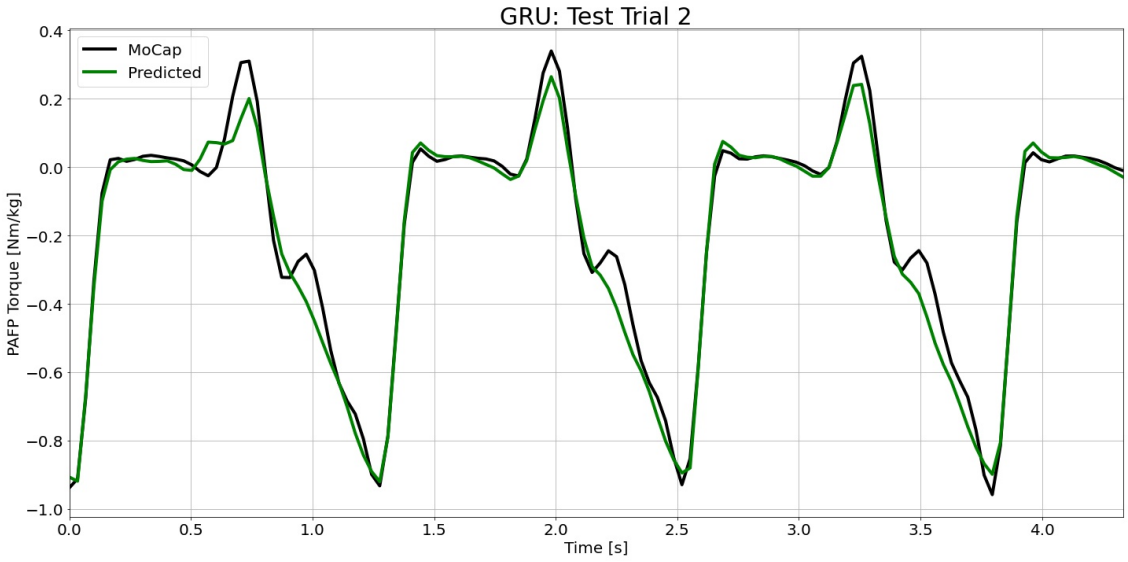


Figure H.53: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 2.

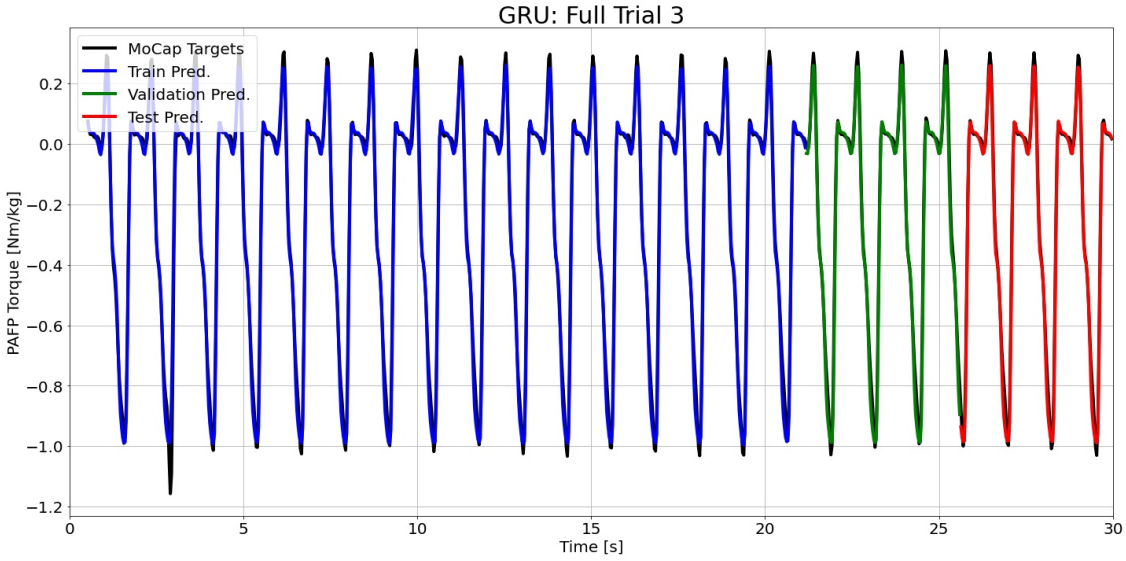


Figure H.54: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 3.

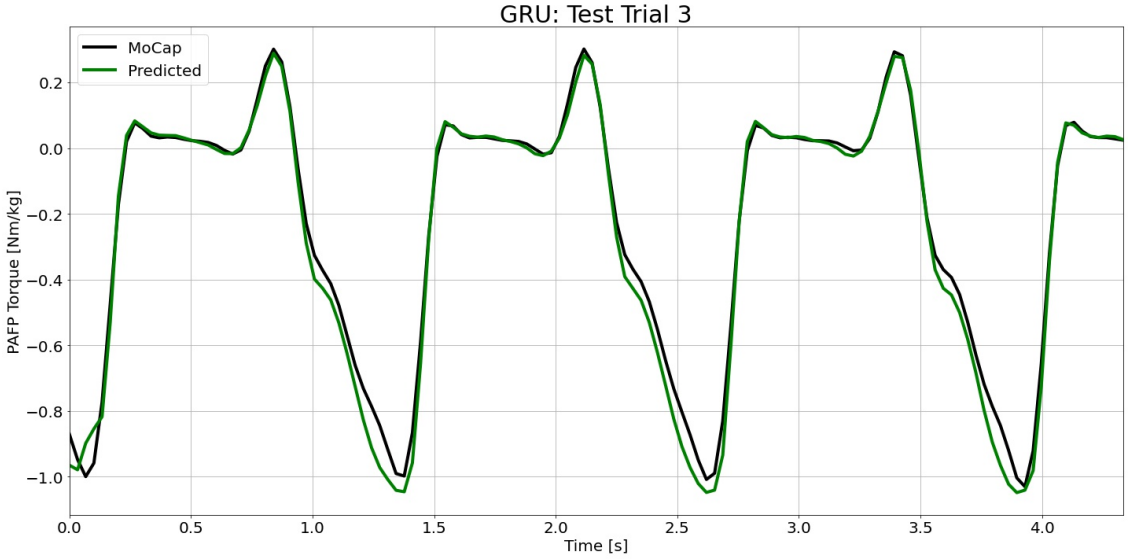


Figure H.55: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 3.

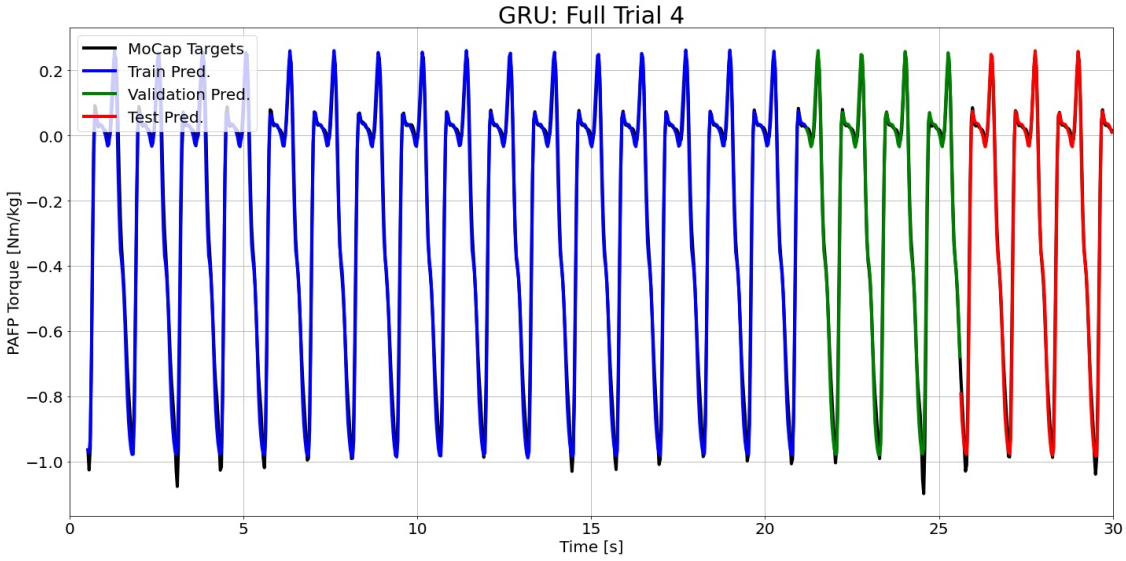


Figure H.56: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 4.

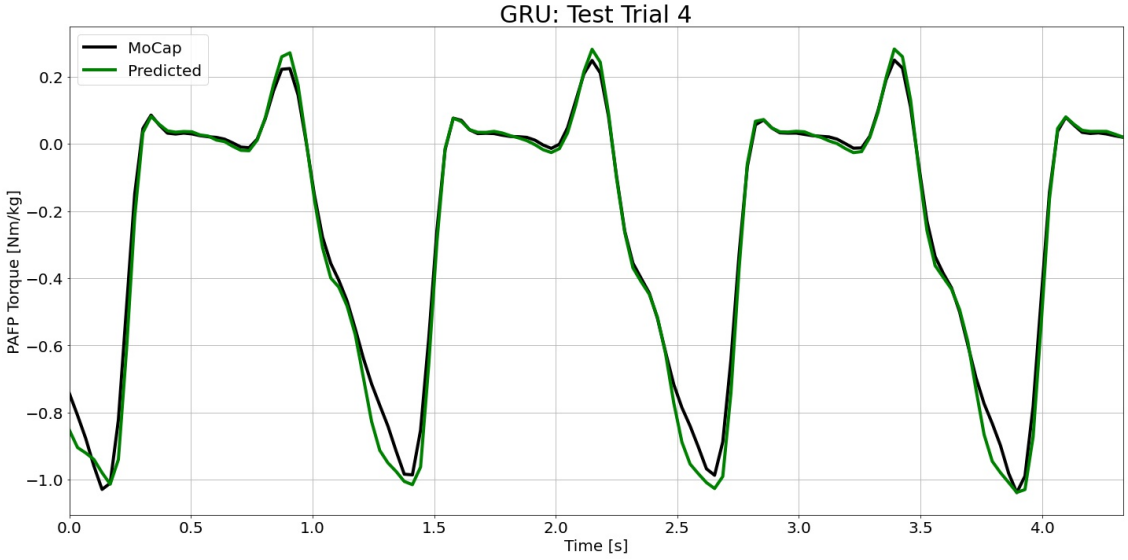


Figure H.57: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 4.

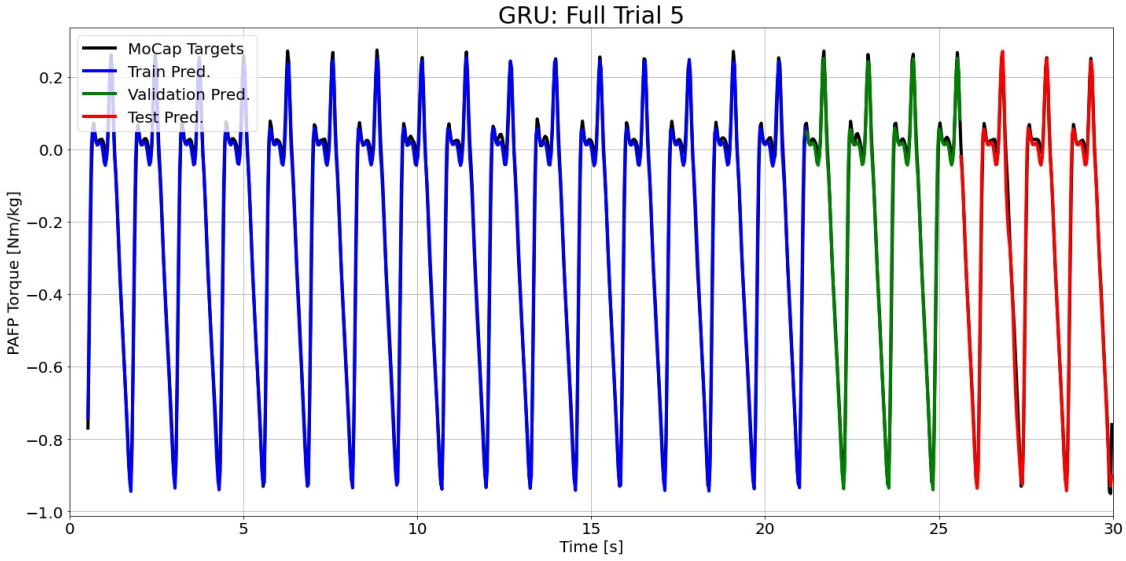


Figure H.58: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 5.

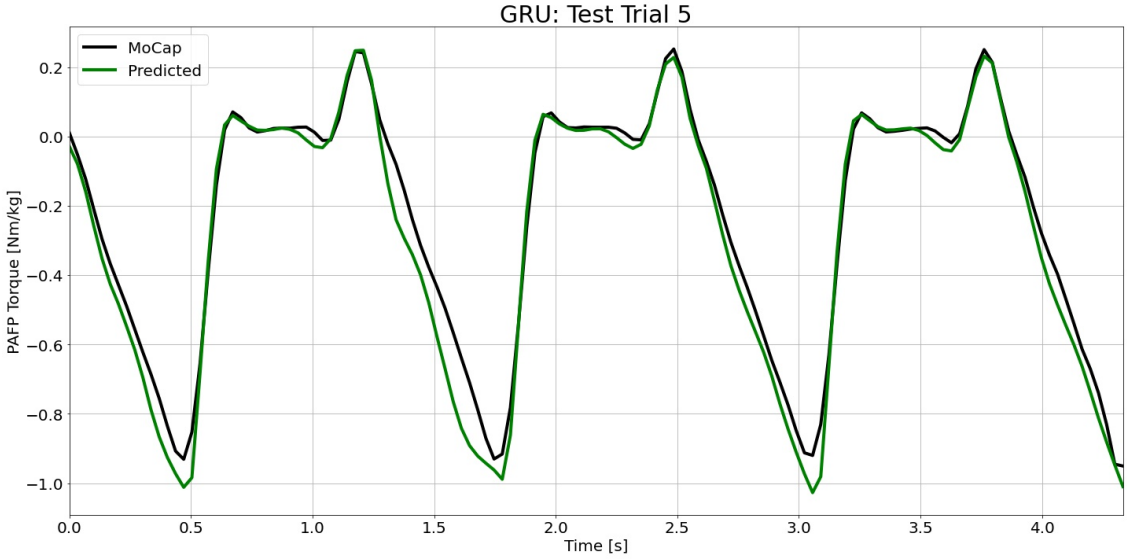


Figure H.59: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 5.

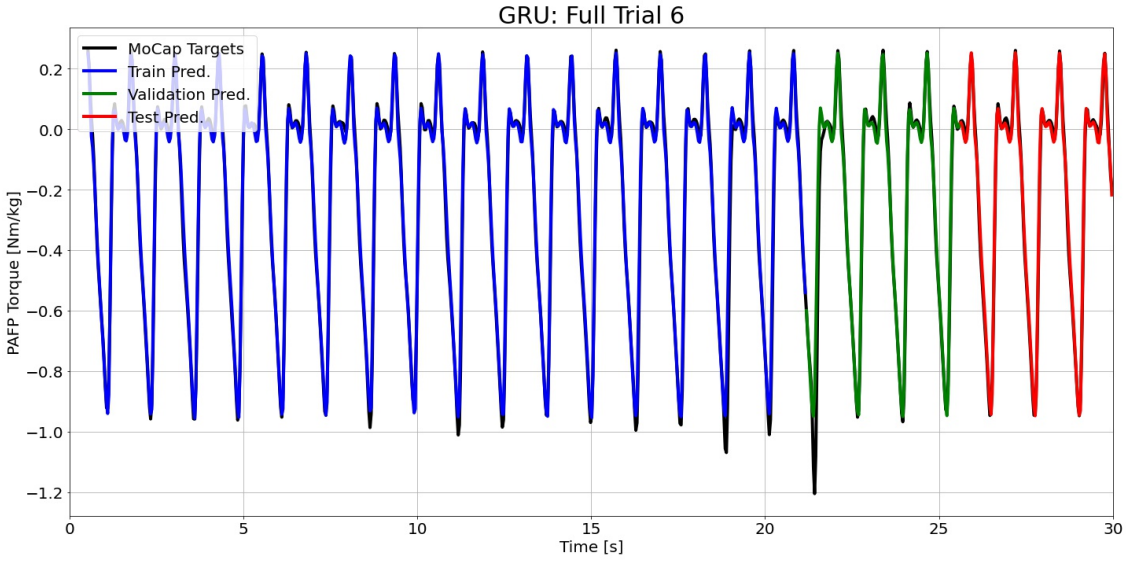


Figure H.60: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 6.

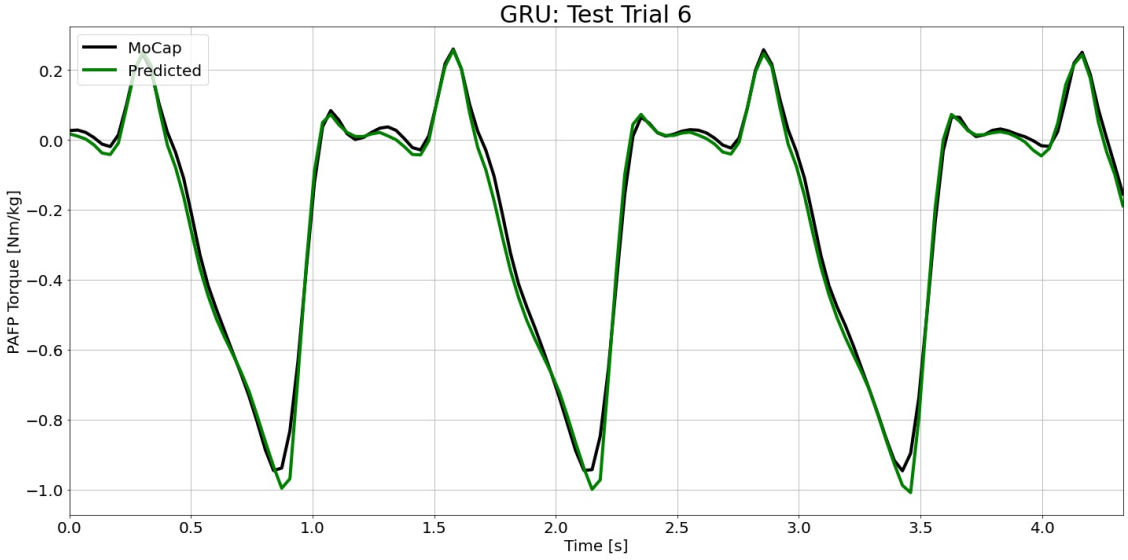


Figure H.61: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 6.

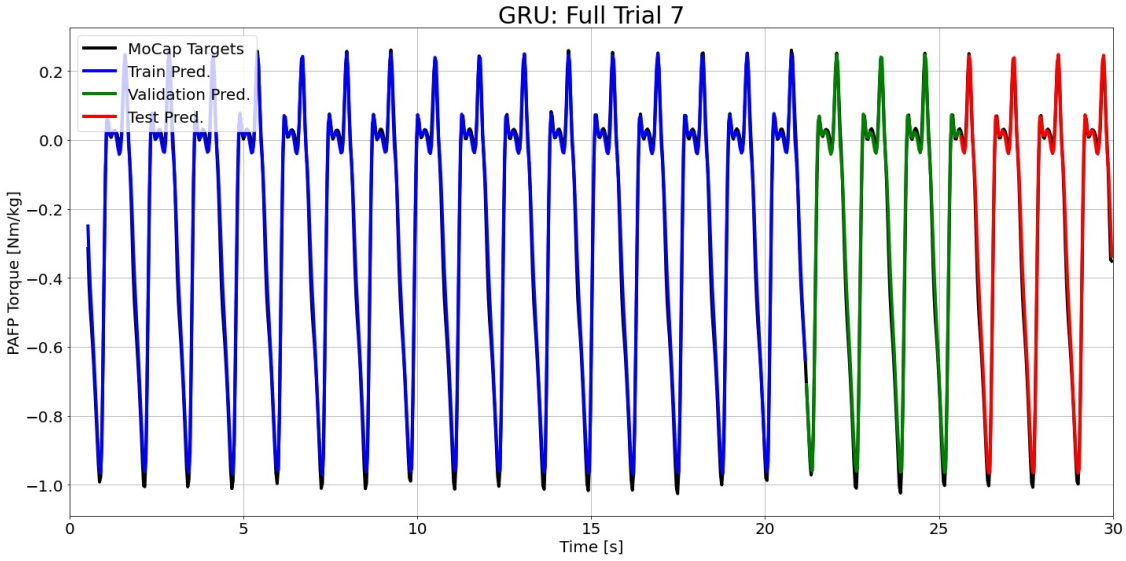


Figure H.62: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 7.

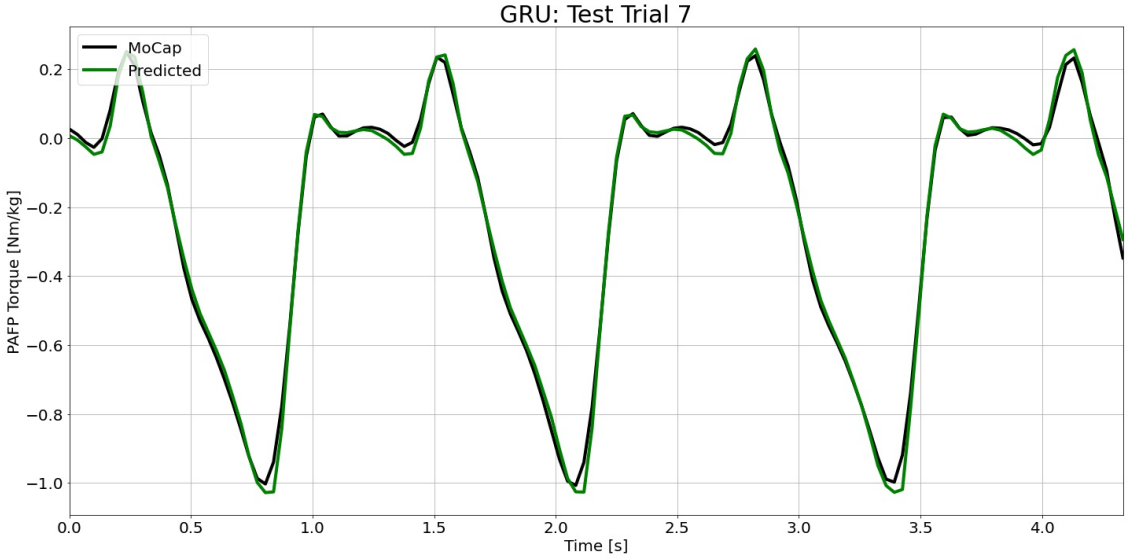


Figure H.63: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 7.

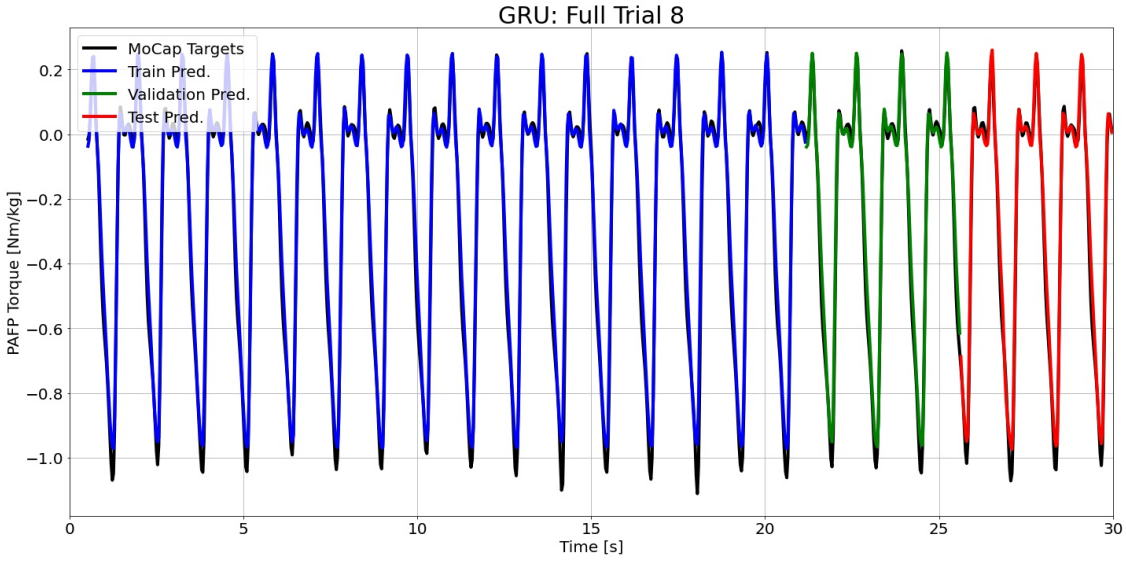


Figure H.64: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 8.

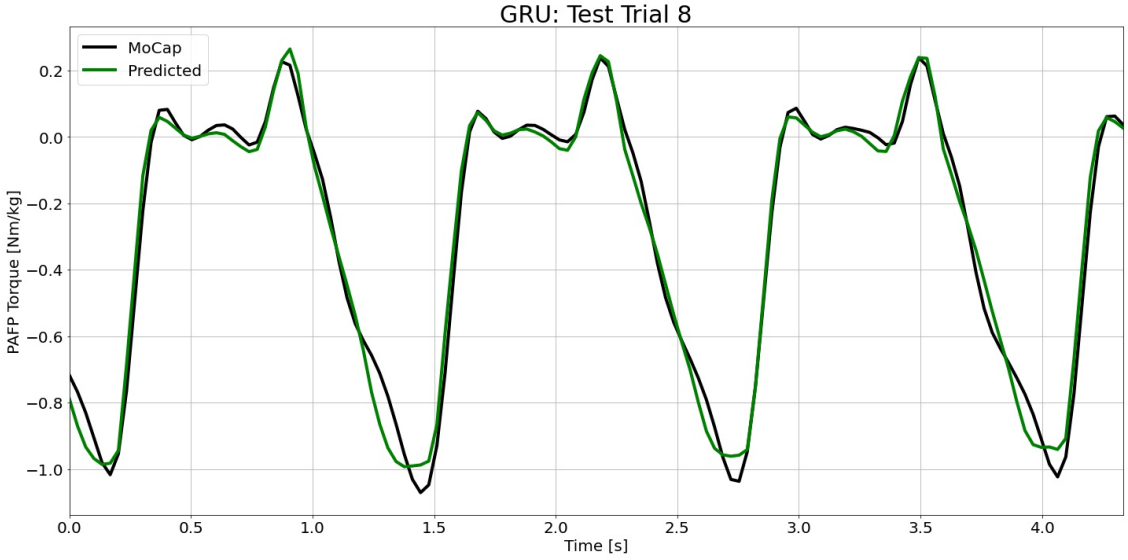


Figure H.65: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 8.

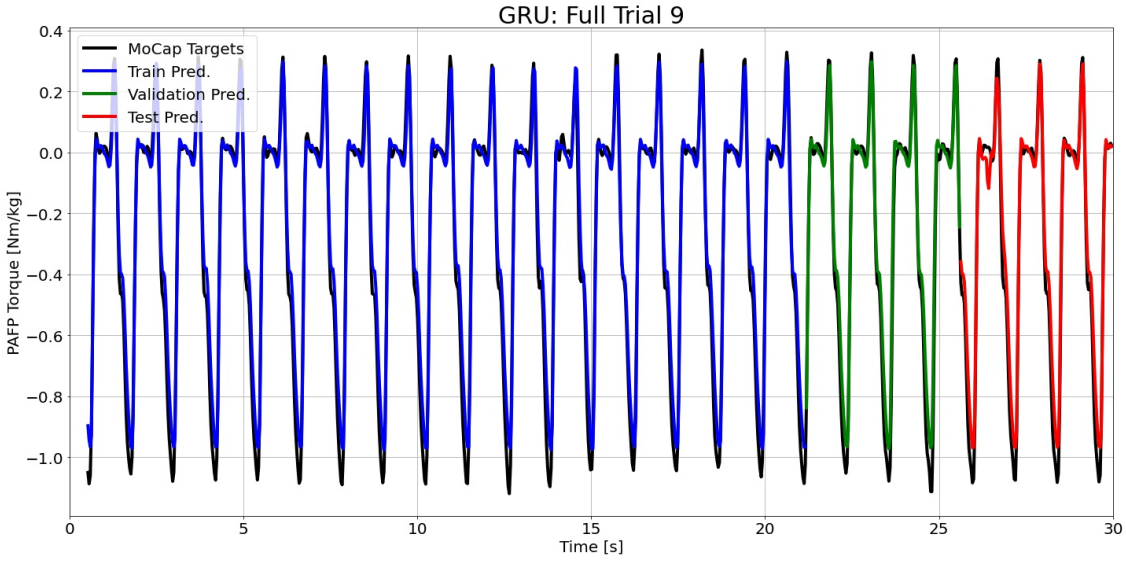


Figure H.66: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 9.

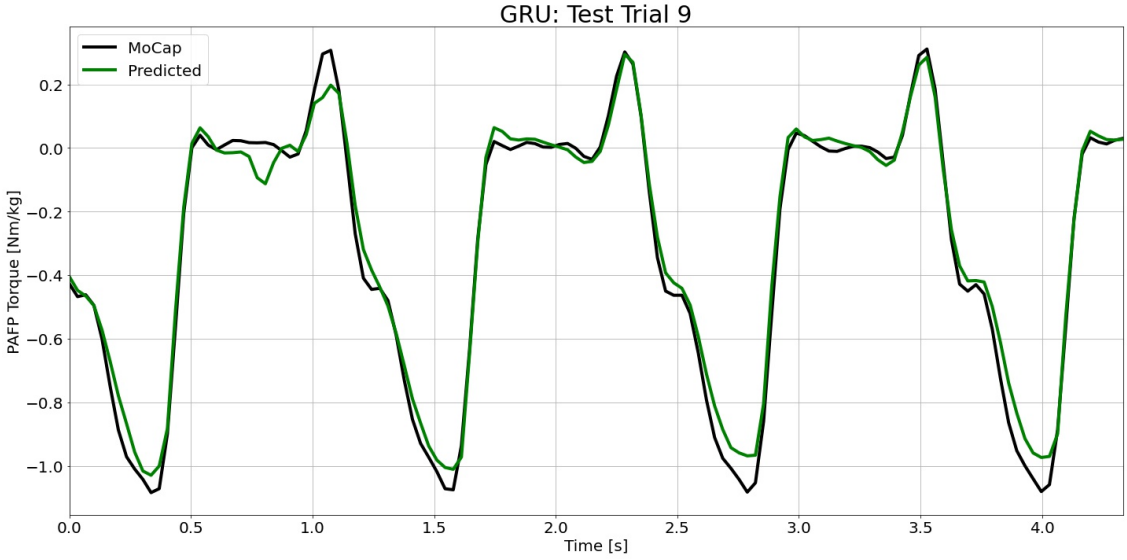


Figure H.67: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 9.

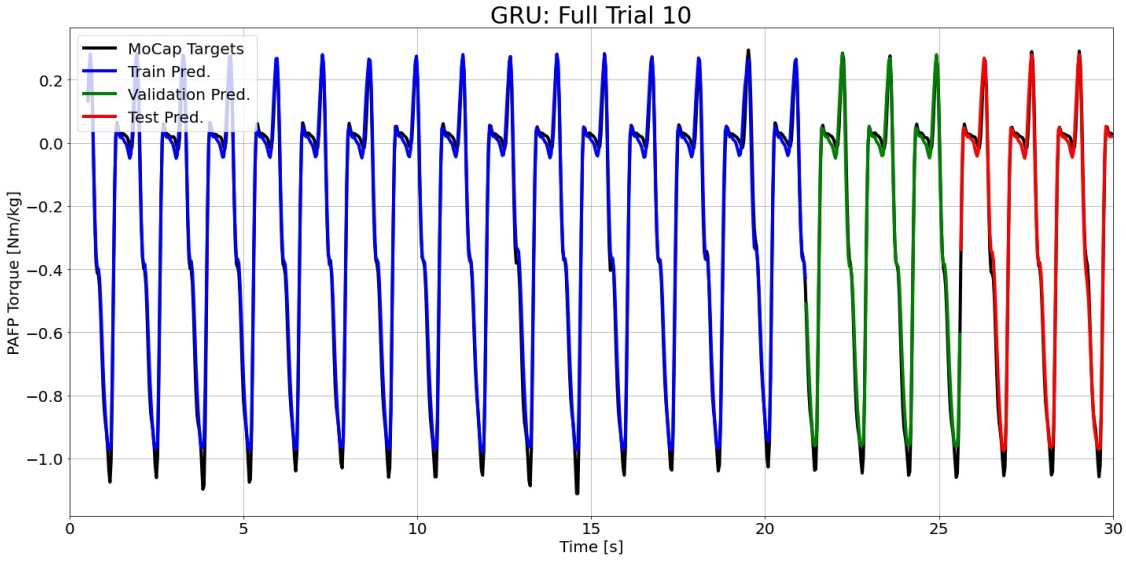


Figure H.68: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 10.

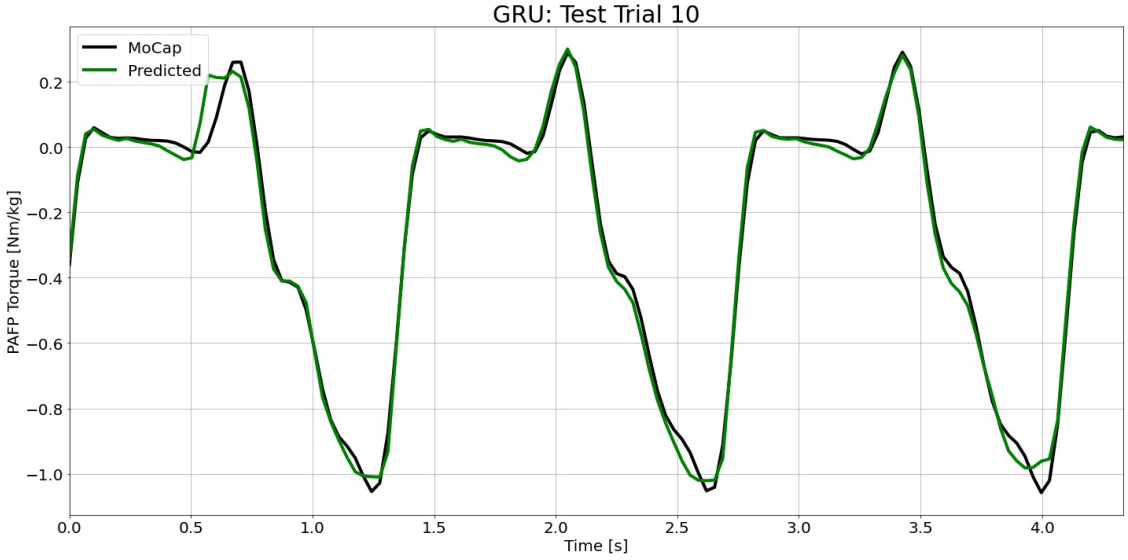


Figure H.69: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 10.

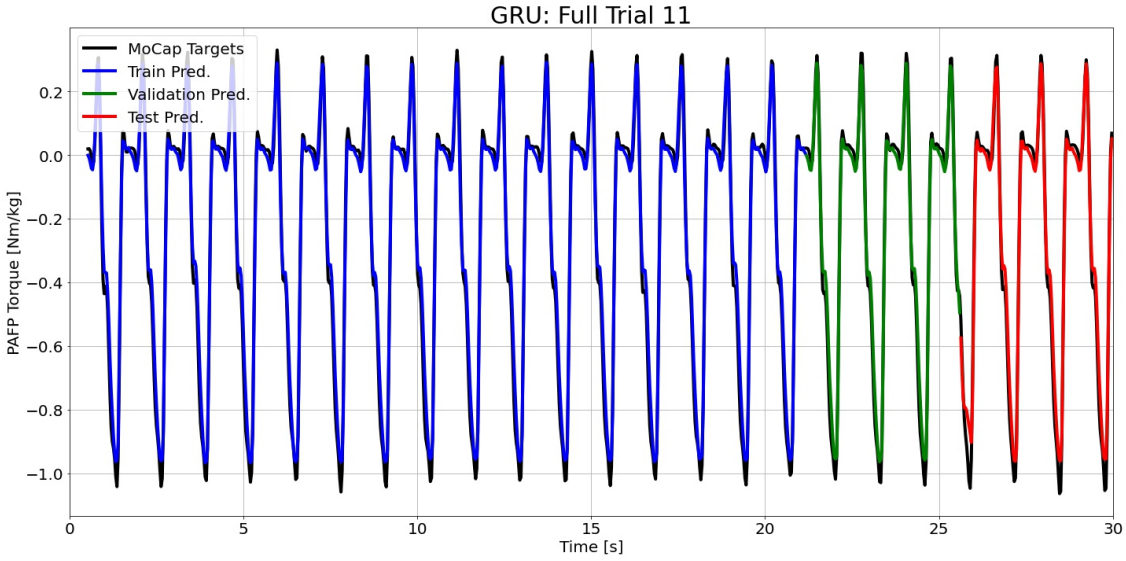


Figure H.70: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 11.

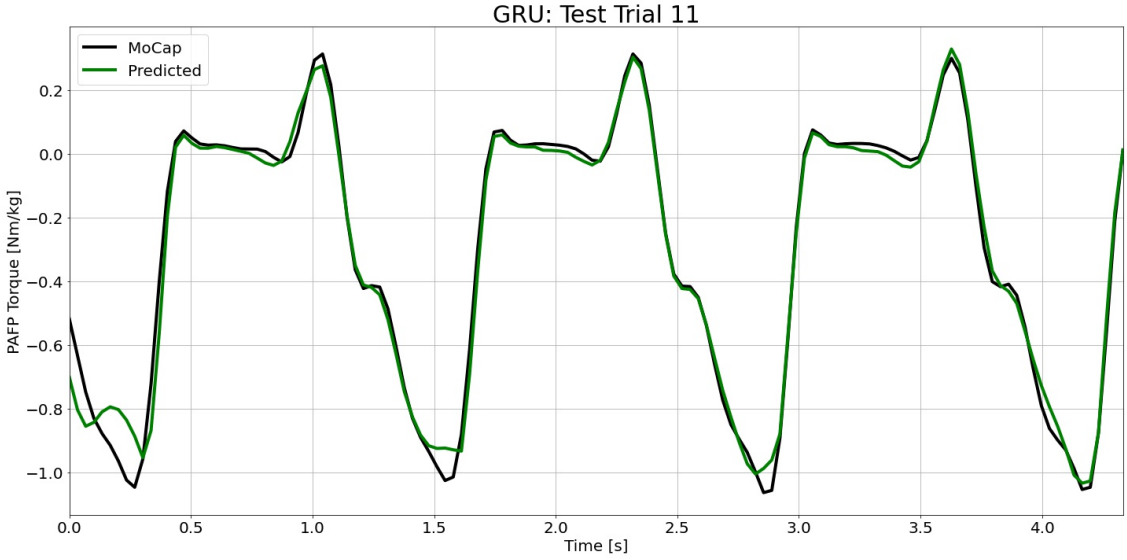


Figure H.71: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 11.

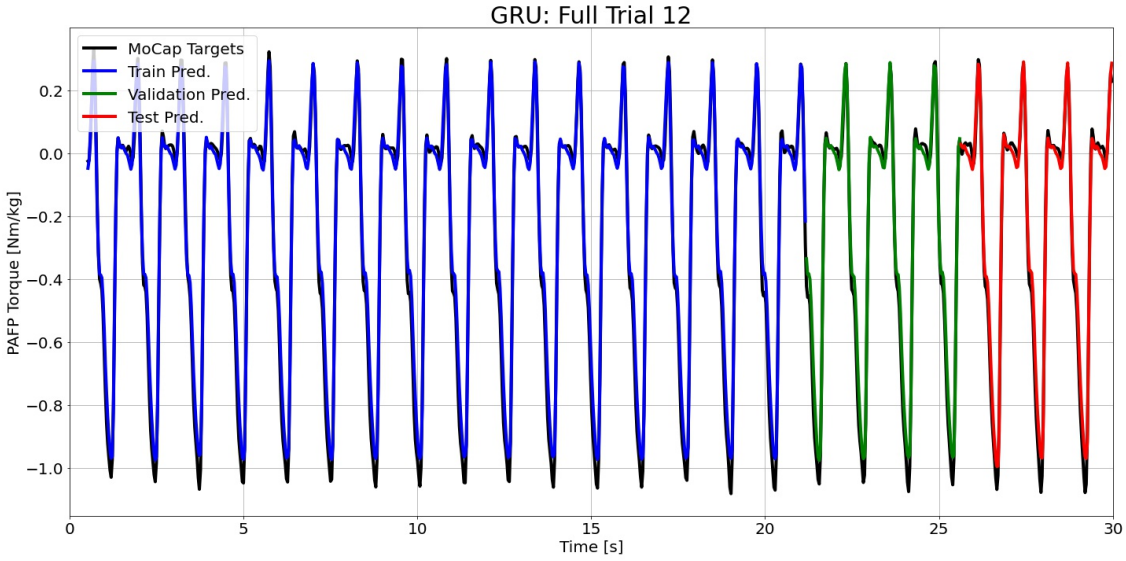


Figure H.72: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 12.

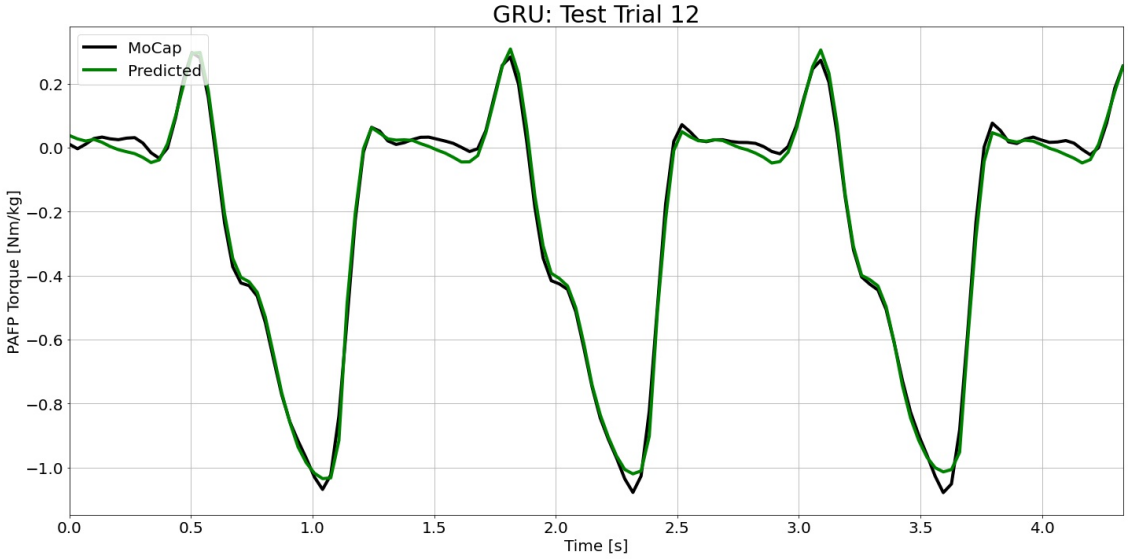


Figure H.73: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 12.

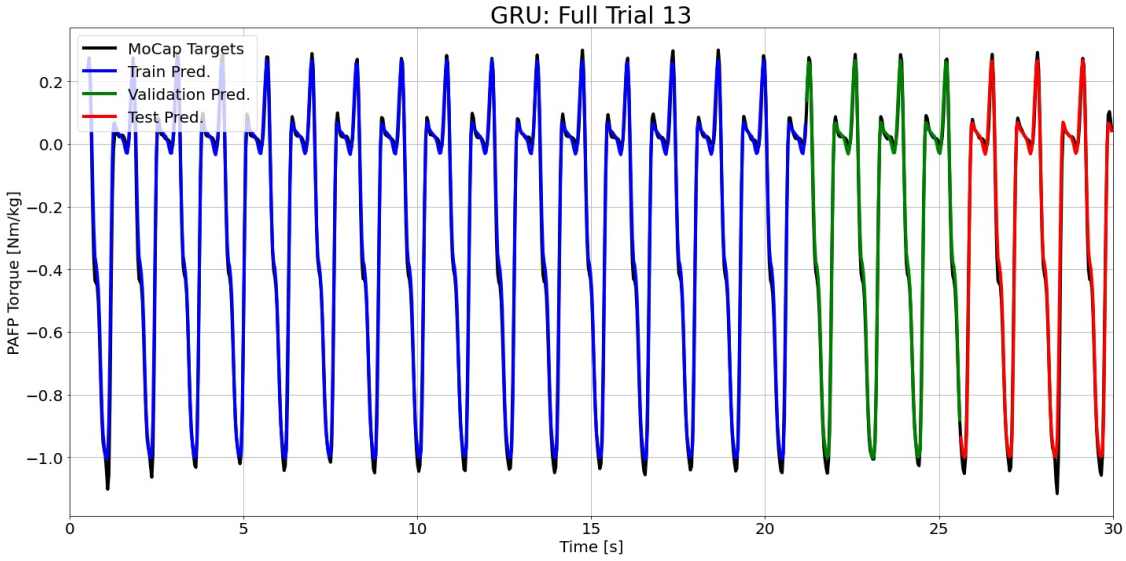


Figure H.74: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 13.

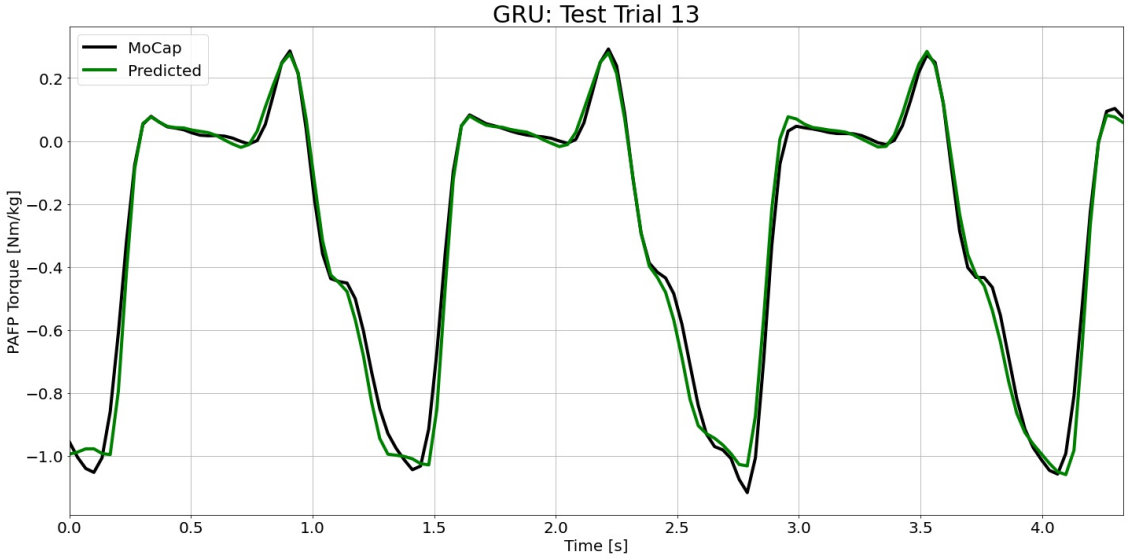


Figure H.75: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 13.

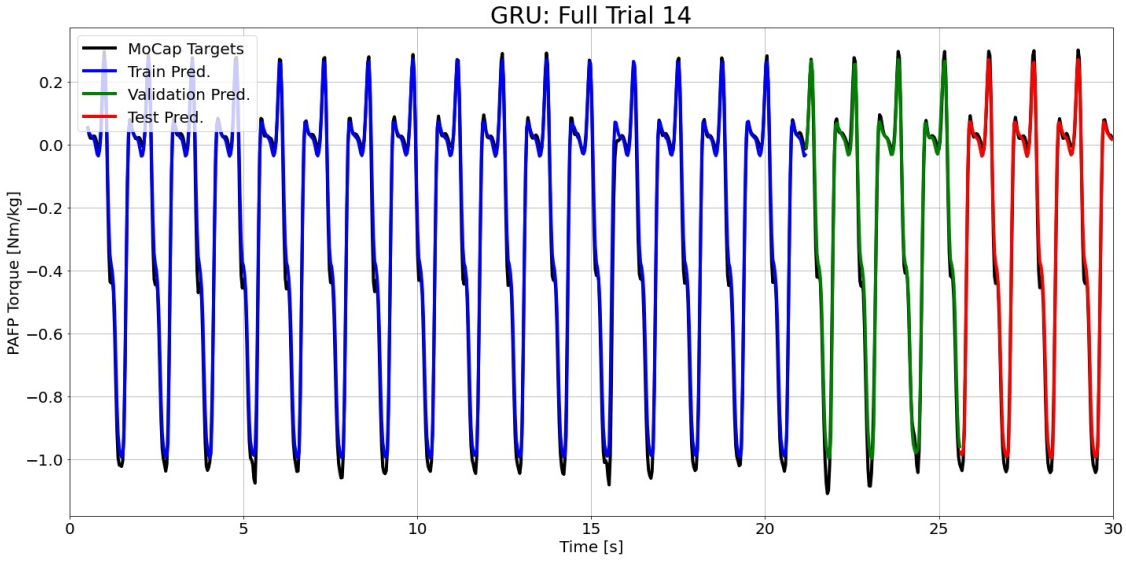


Figure H.76: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 14.

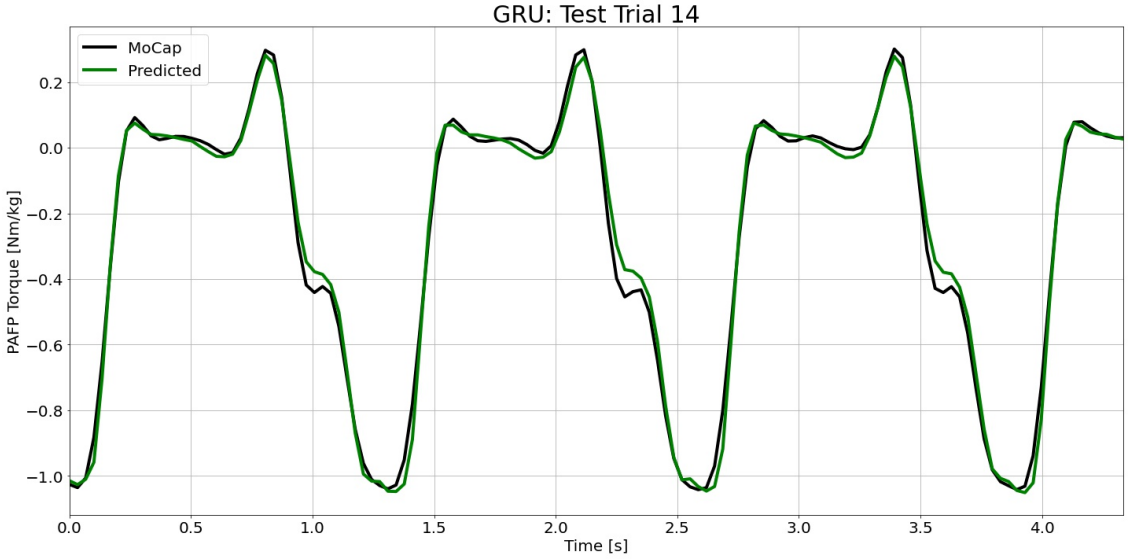


Figure H.77: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 14.

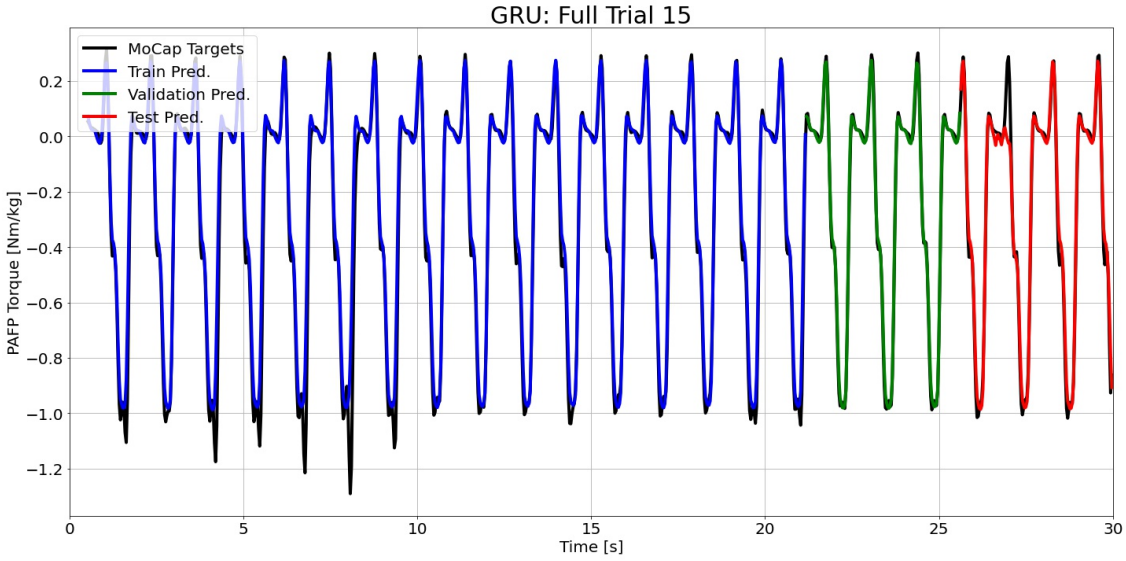


Figure H.78: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 15.

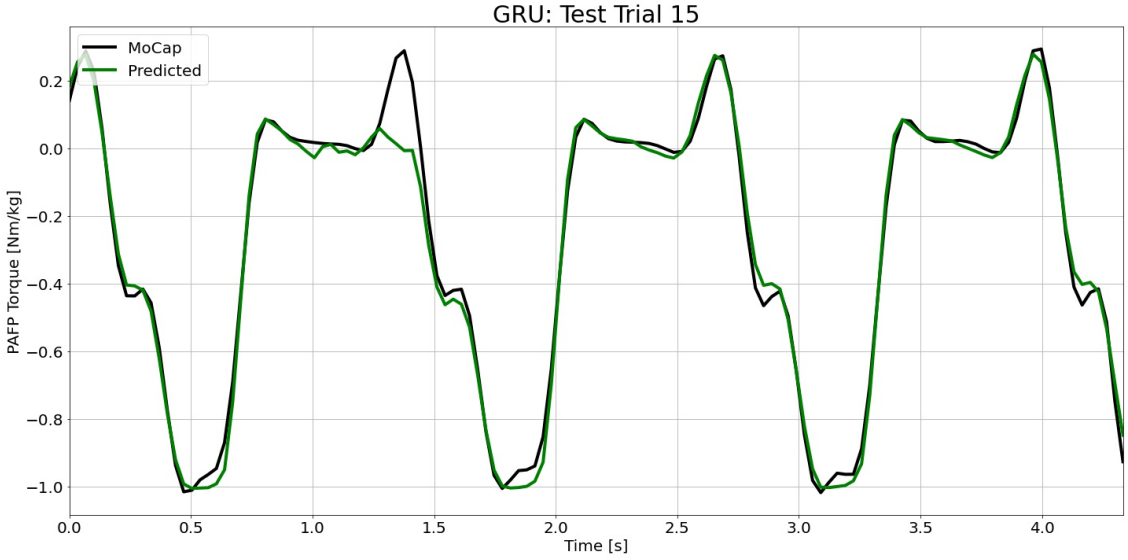


Figure H.79: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 15.

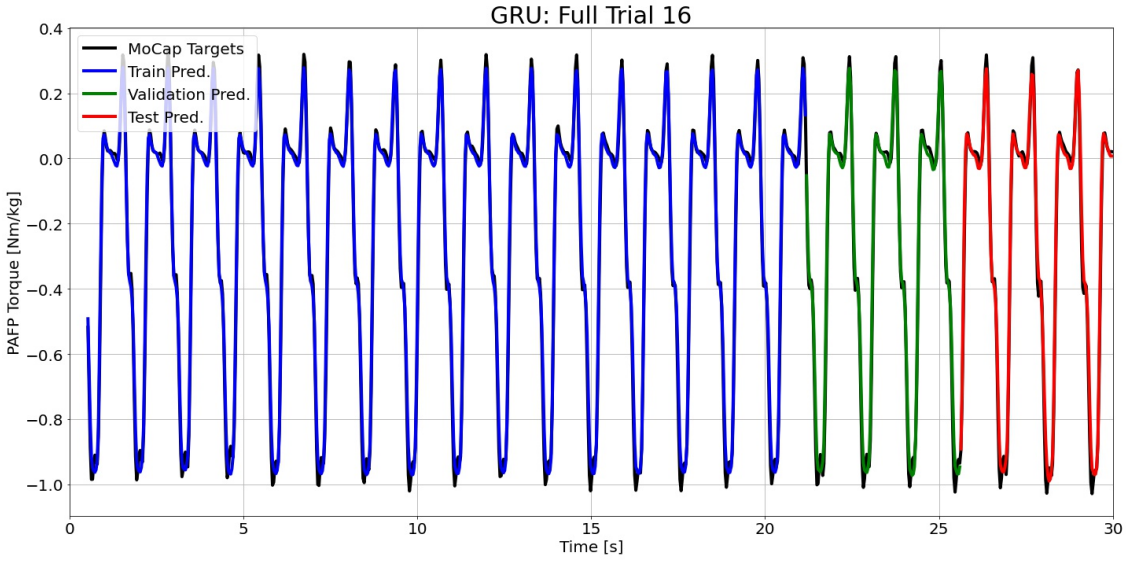


Figure H.80: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 16.

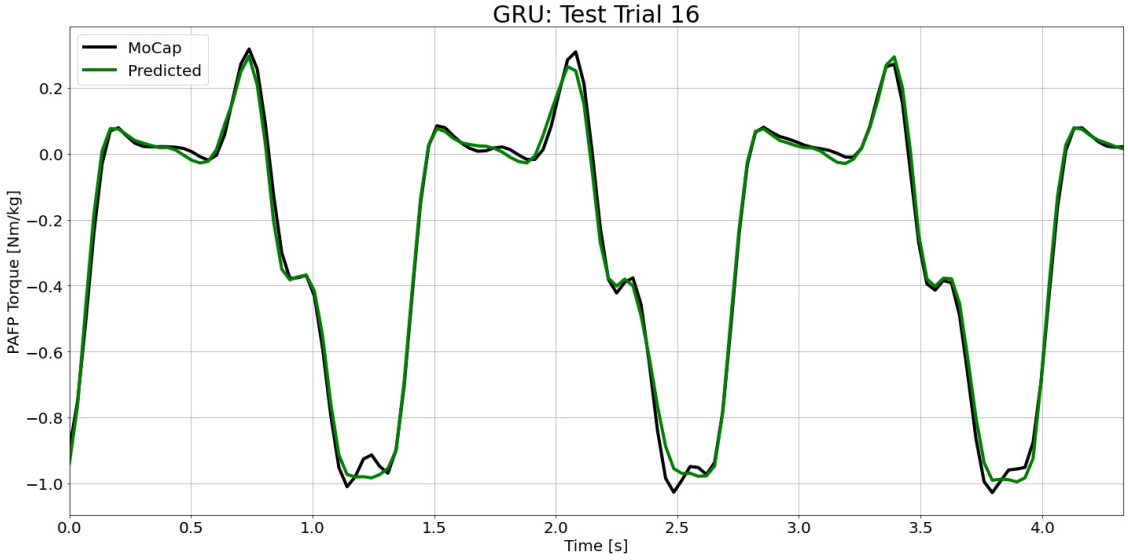


Figure H.81: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 16.

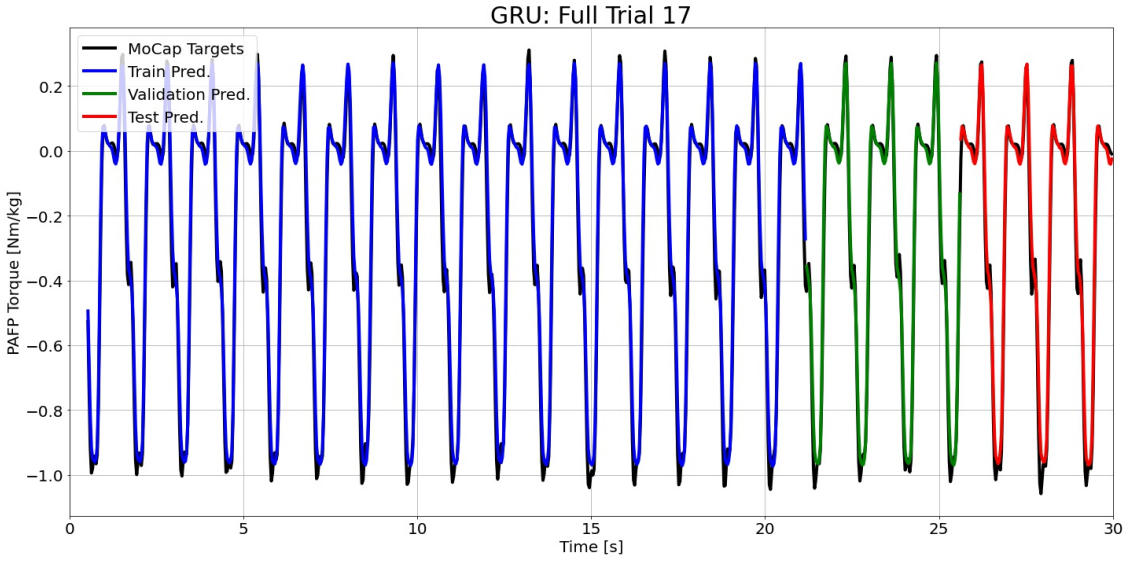


Figure H.82: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 17.

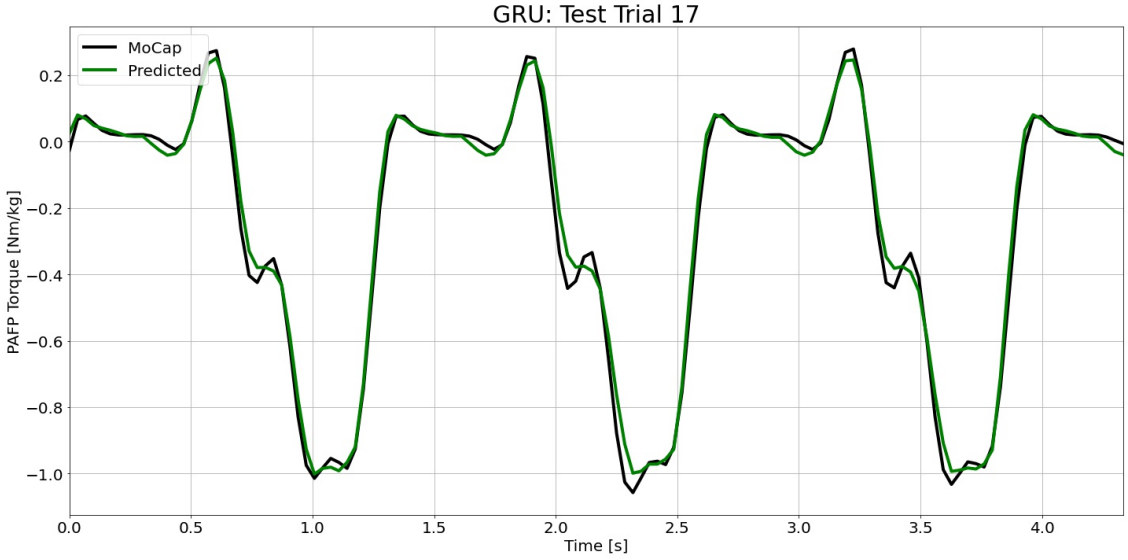


Figure H.83: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 17.

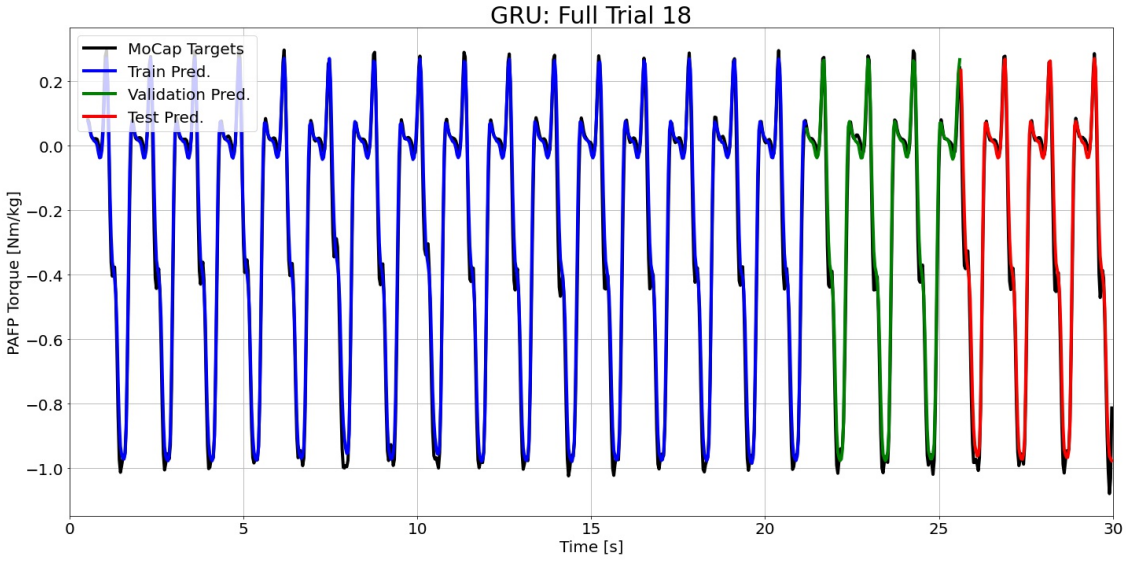


Figure H.84: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 18.

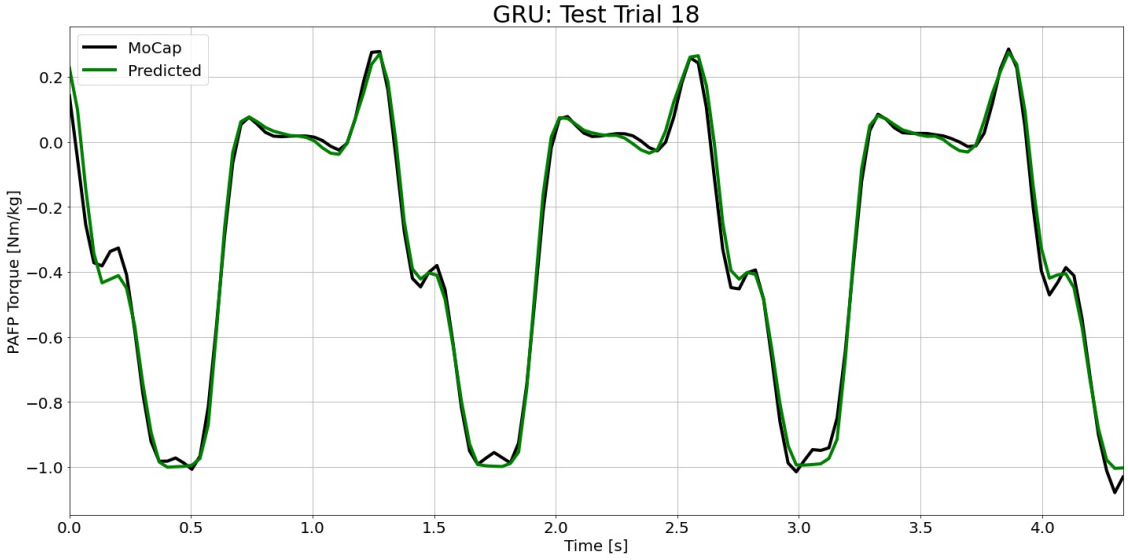


Figure H.85: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 18.

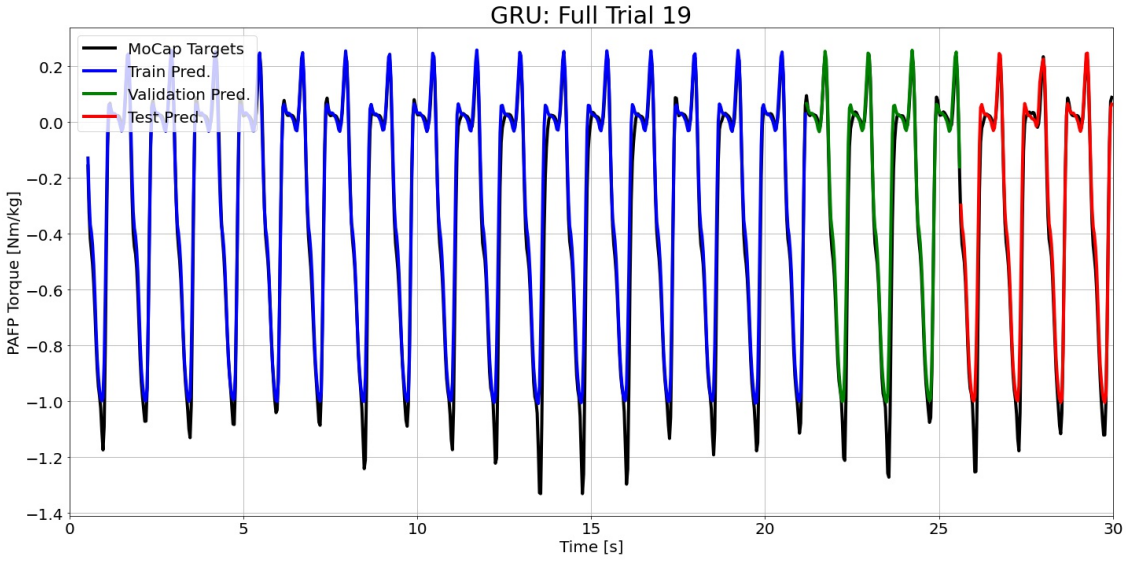


Figure H.86: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 19.

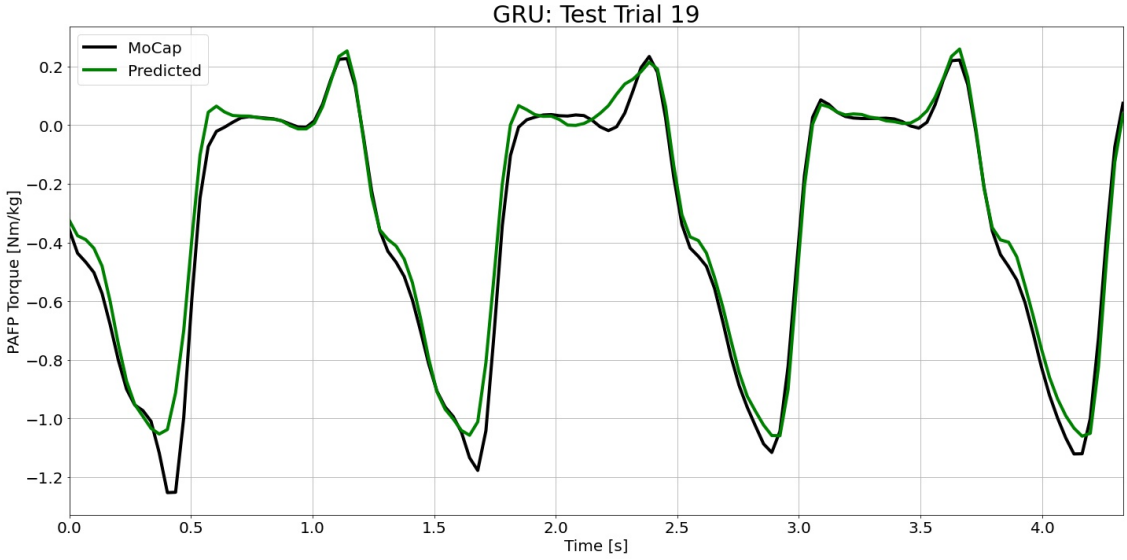


Figure H.87: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 19.

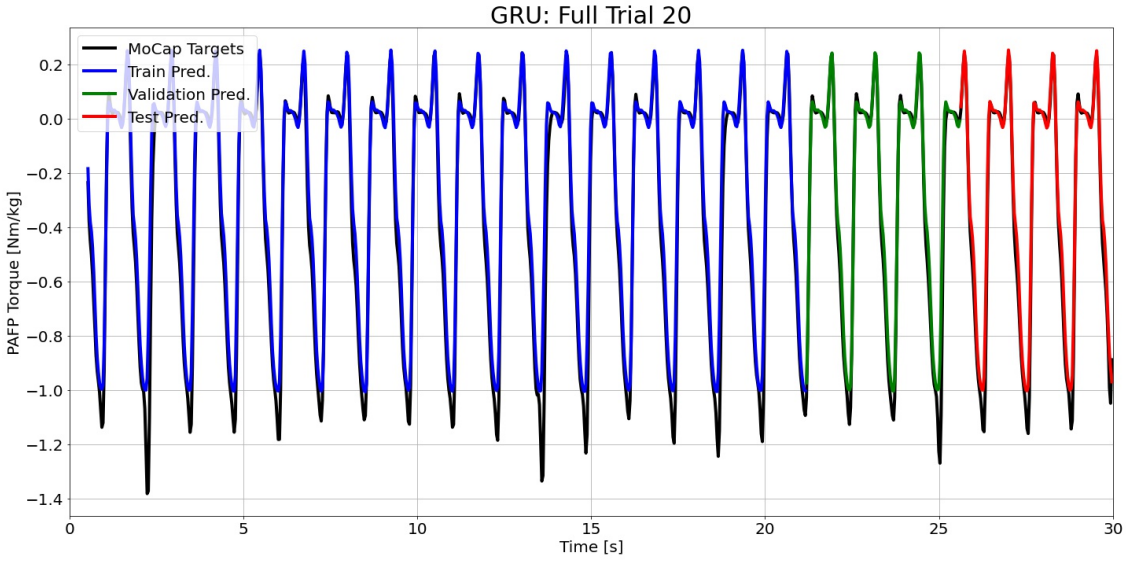


Figure H.88: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 20.

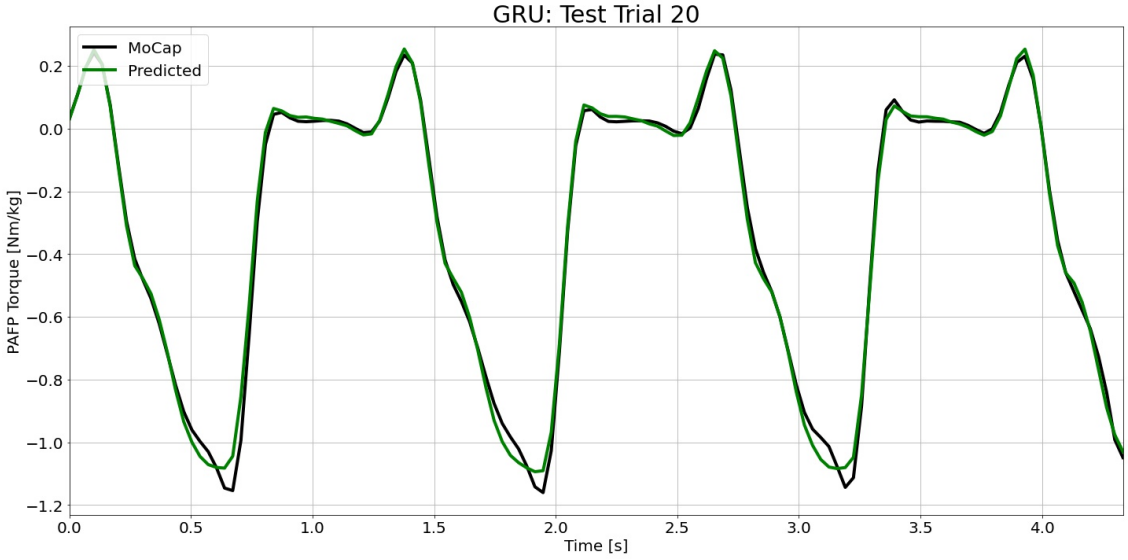


Figure H.89: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 20.

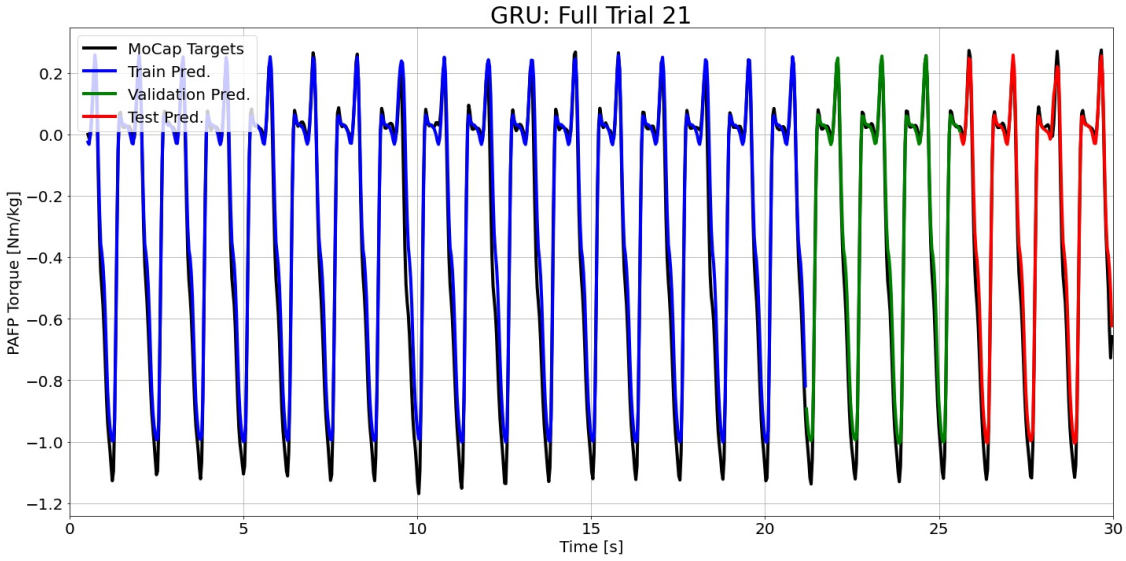


Figure H.90: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 21.

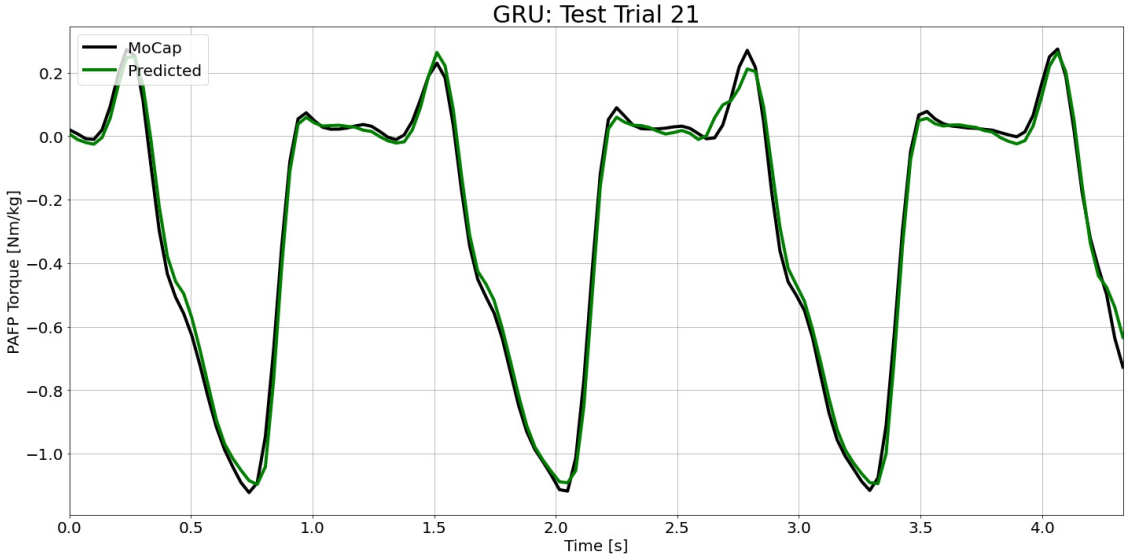


Figure H.91: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 21.

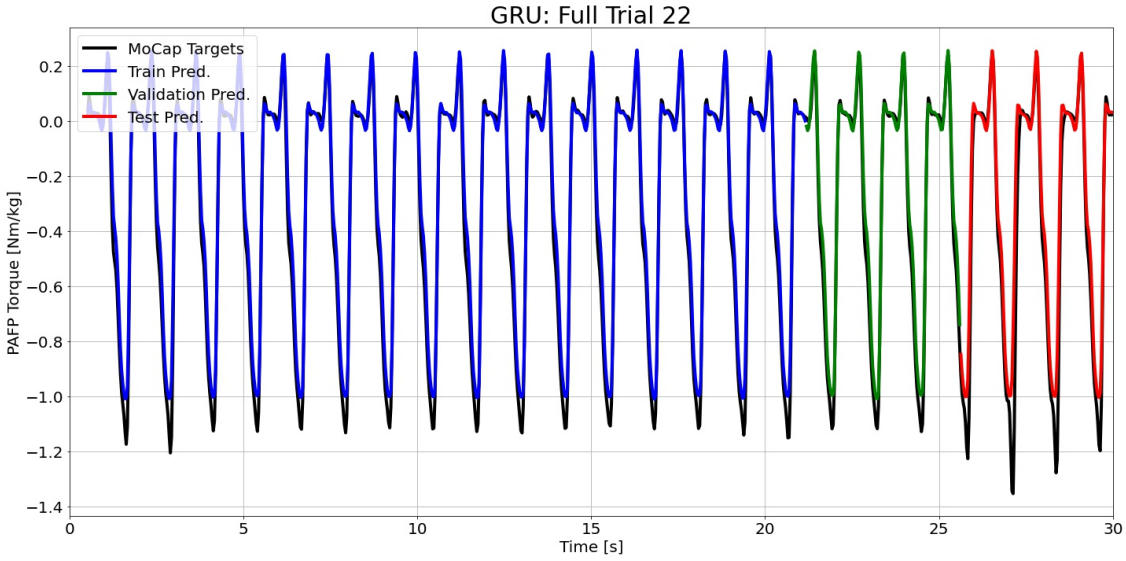


Figure H.92: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 22.

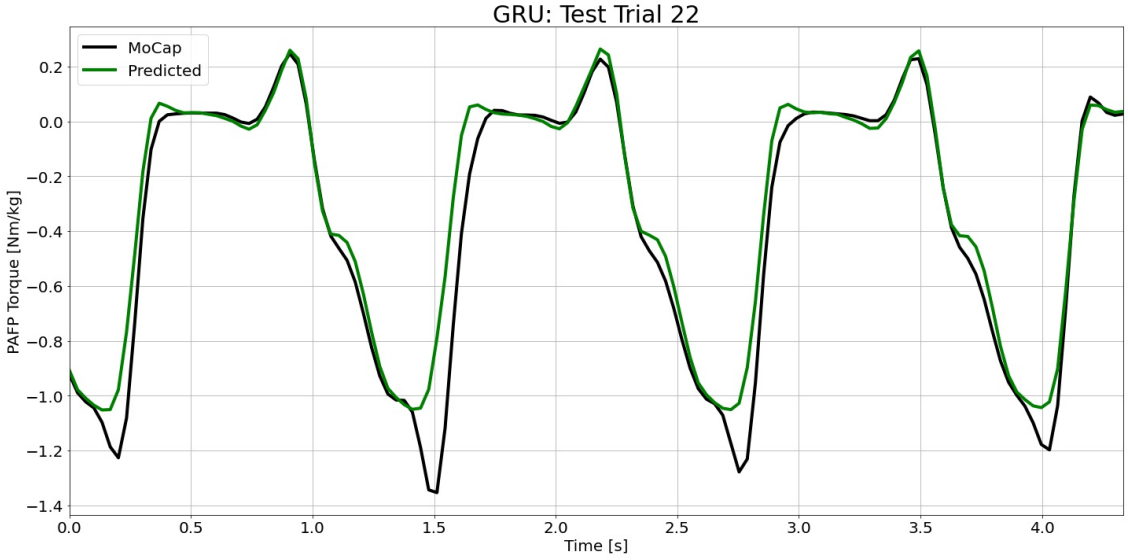


Figure H.93: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 22.

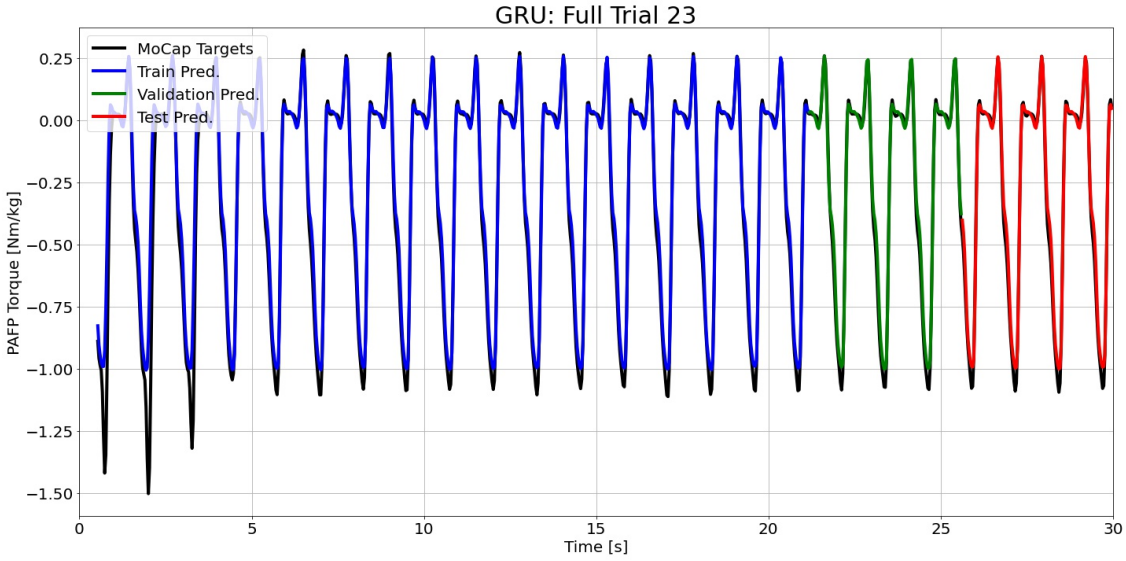


Figure H.94: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 23.

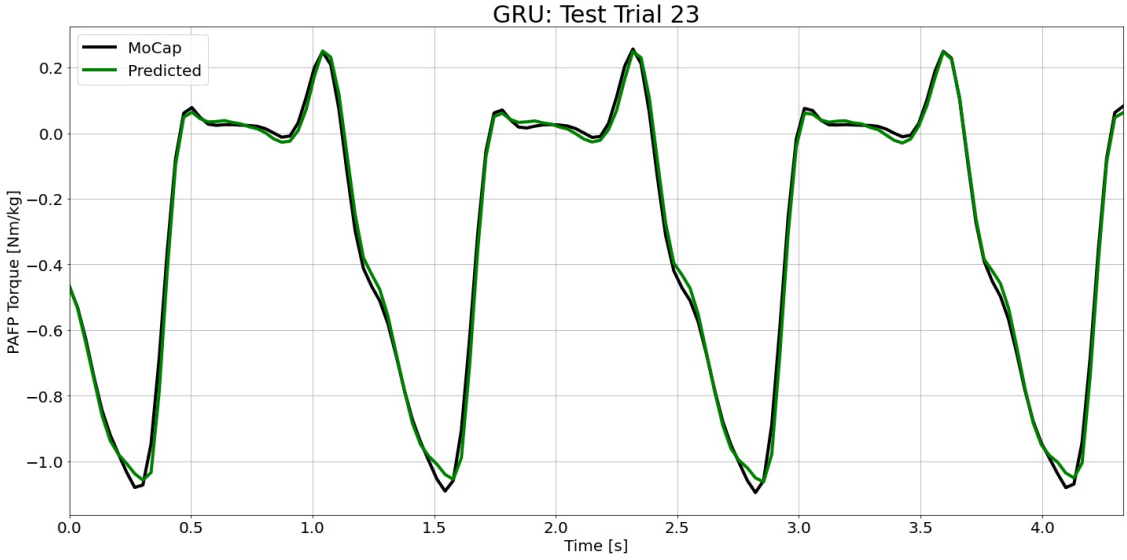


Figure H.95: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 23.

H.3 Dual-Stage Attention-Based Gated Recurrent Unit Neural Network

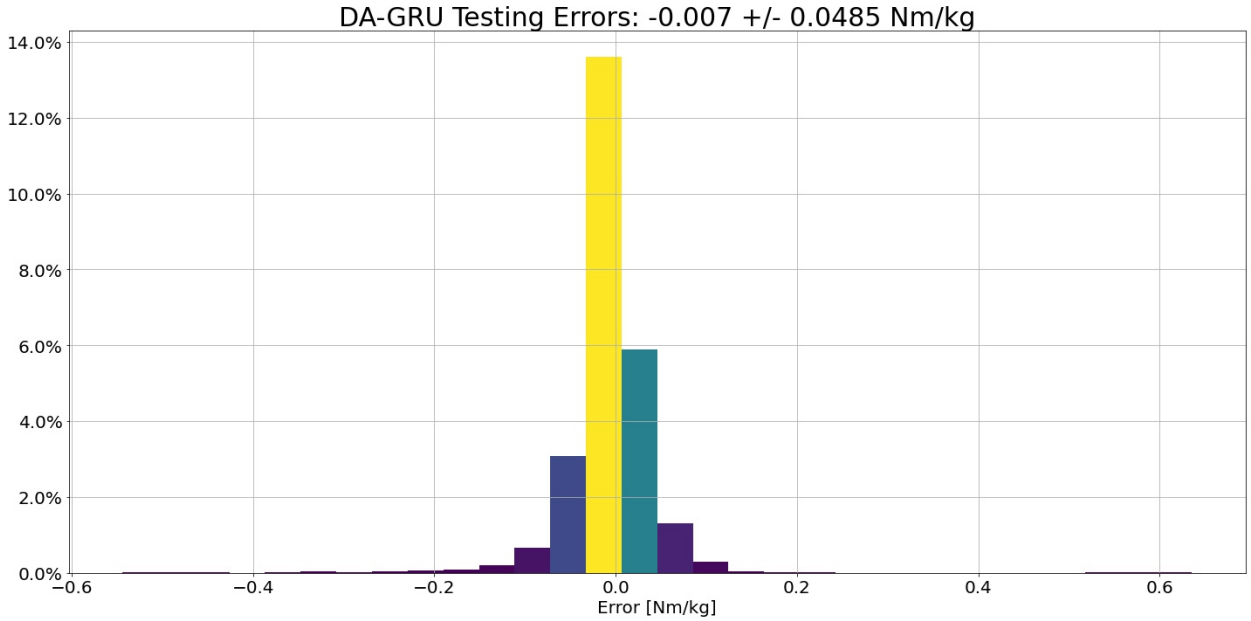


Figure H.96: DA-GRU prediction error histogram for all test time series data.

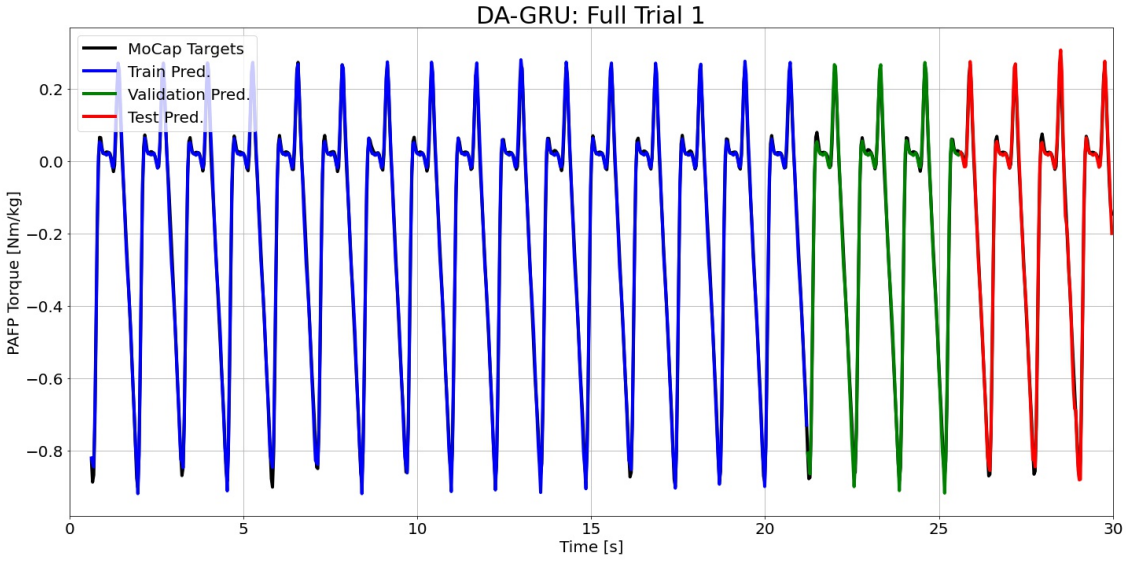


Figure H.97: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 1.

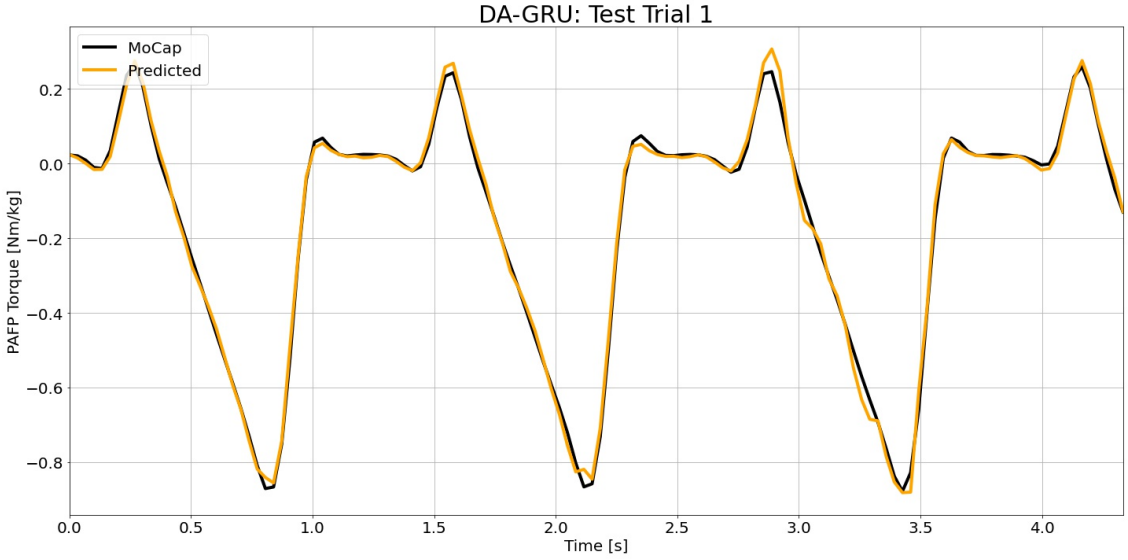


Figure H.98: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 1.

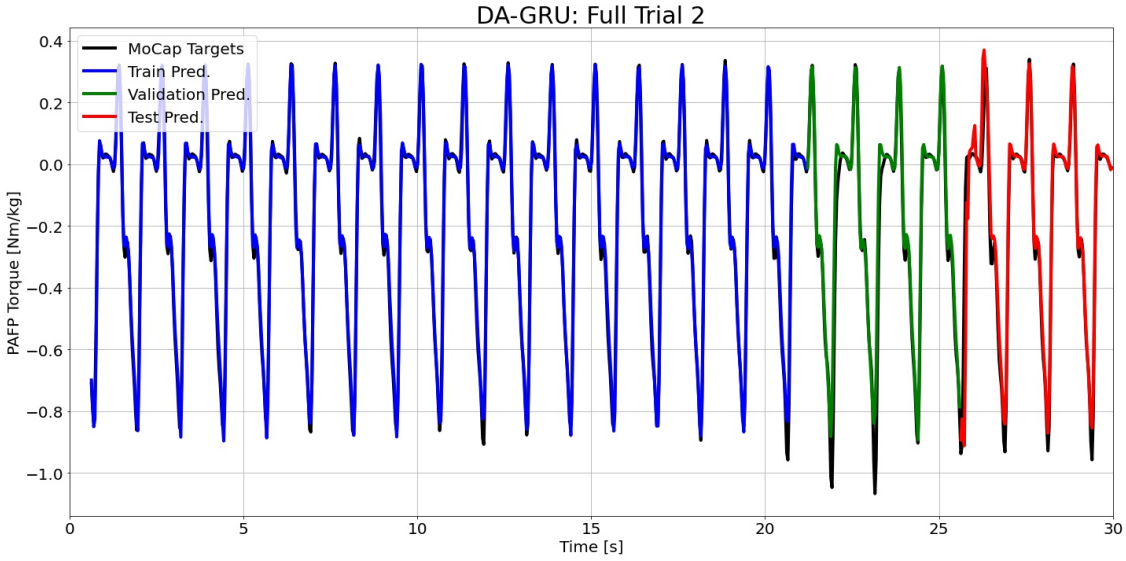


Figure H.99: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 2.

Figure H.100: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 2.

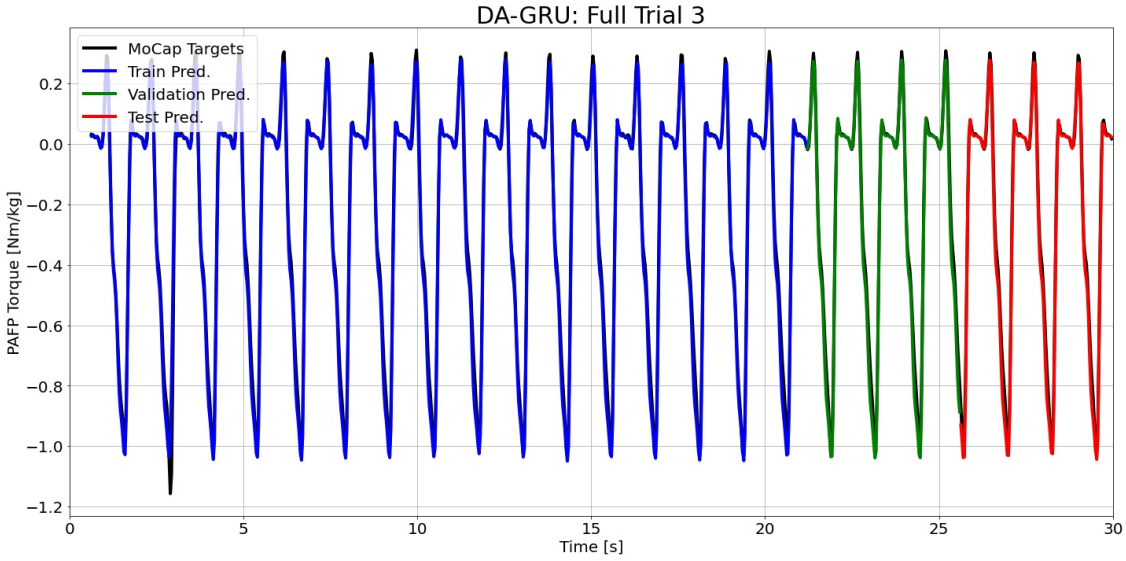


Figure H.101: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 3.

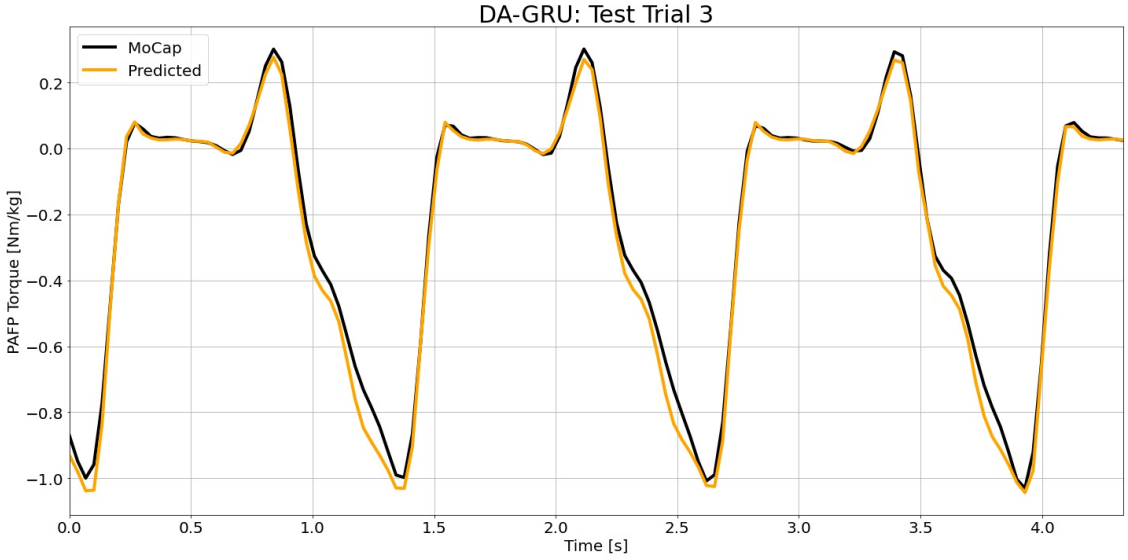


Figure H.102: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 3.

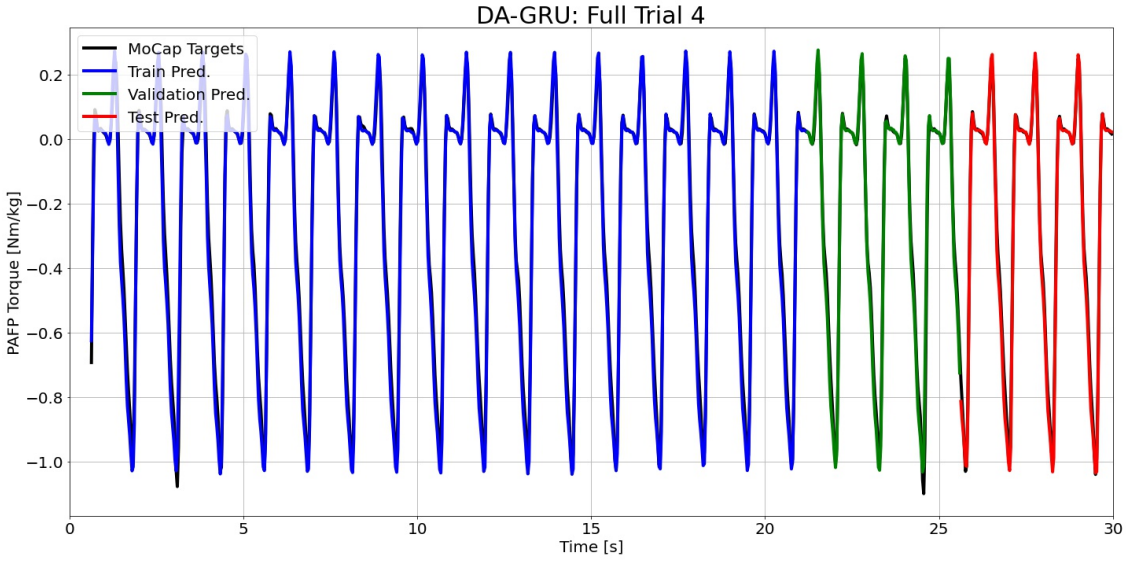


Figure H.103: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 4.

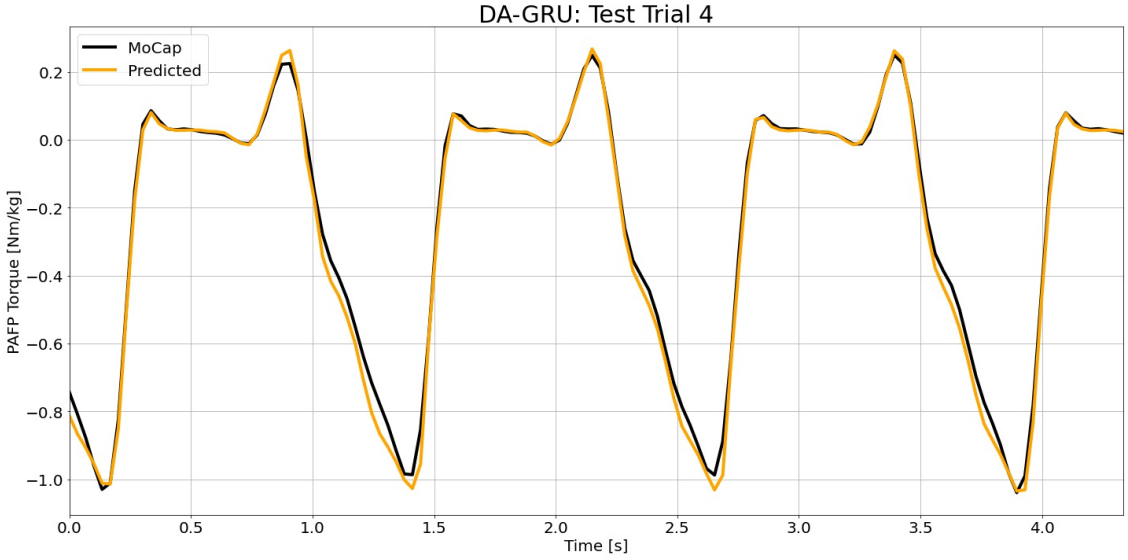


Figure H.104: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 4.

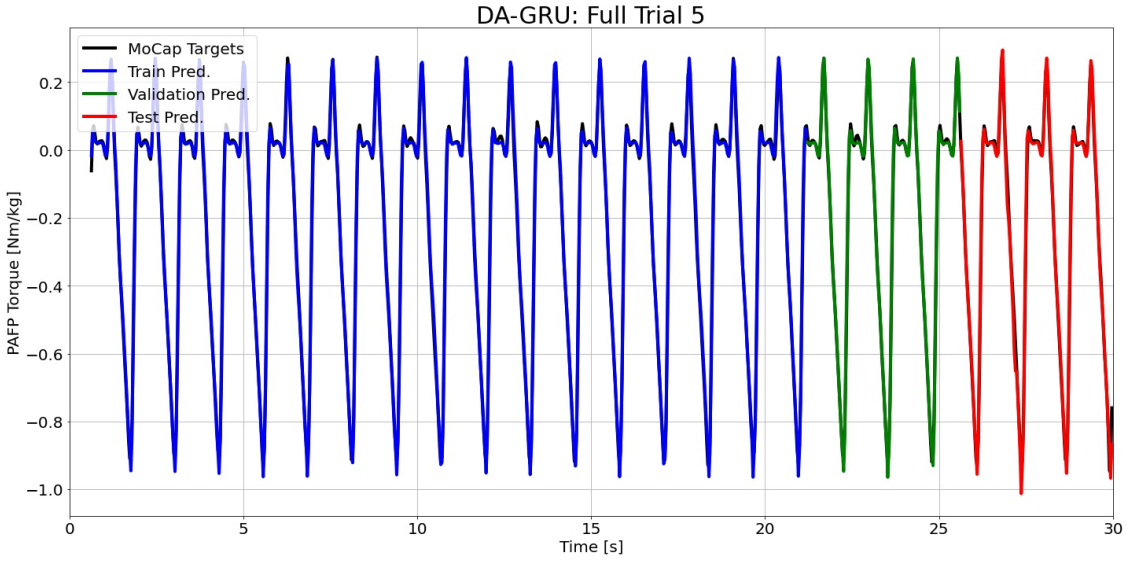


Figure H.105: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 5.

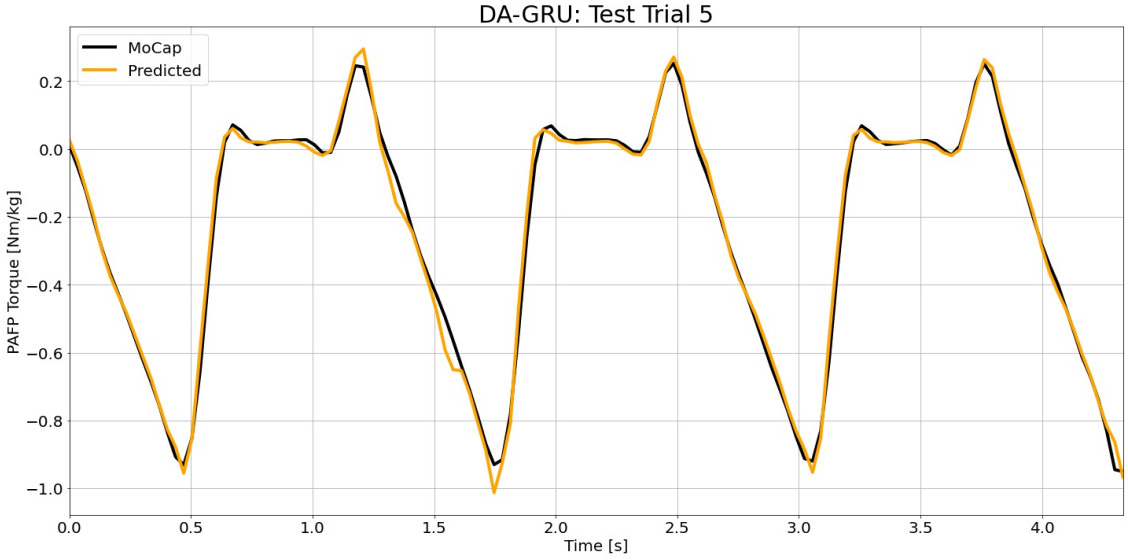


Figure H.106: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 5.

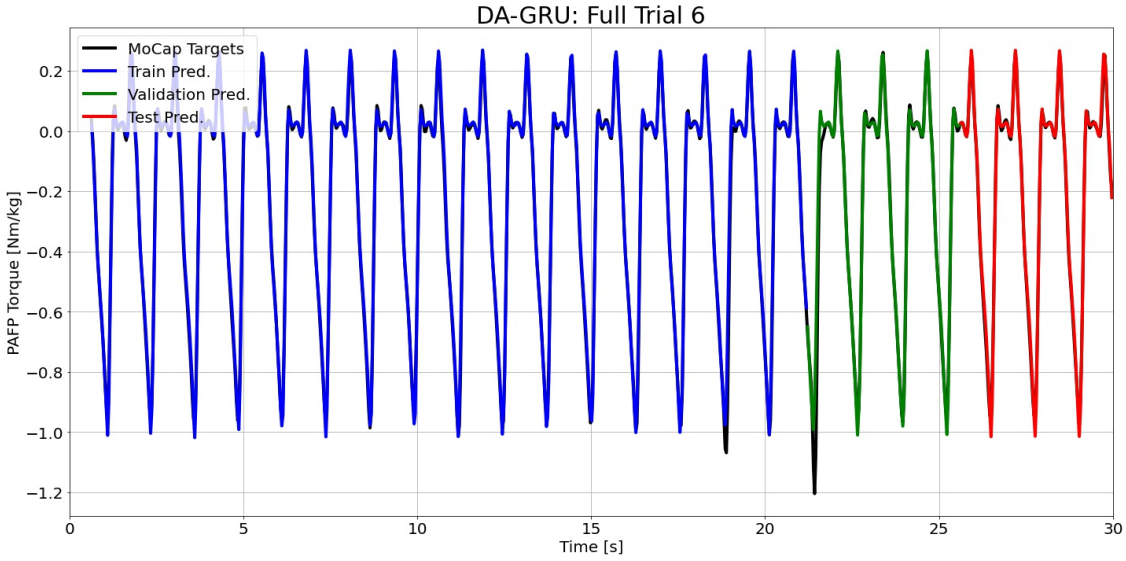


Figure H.107: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 6.

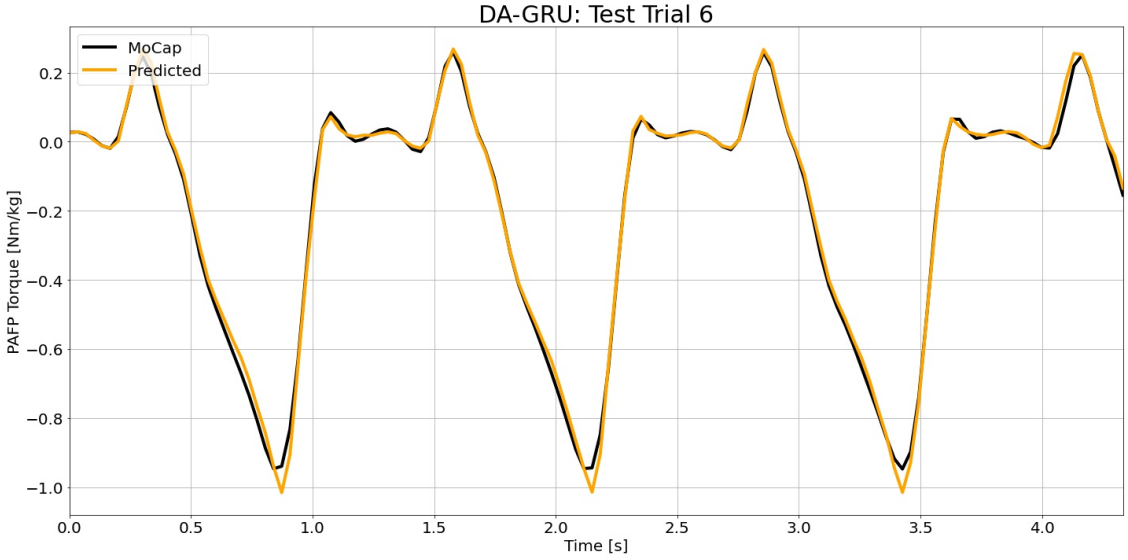


Figure H.108: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 6.

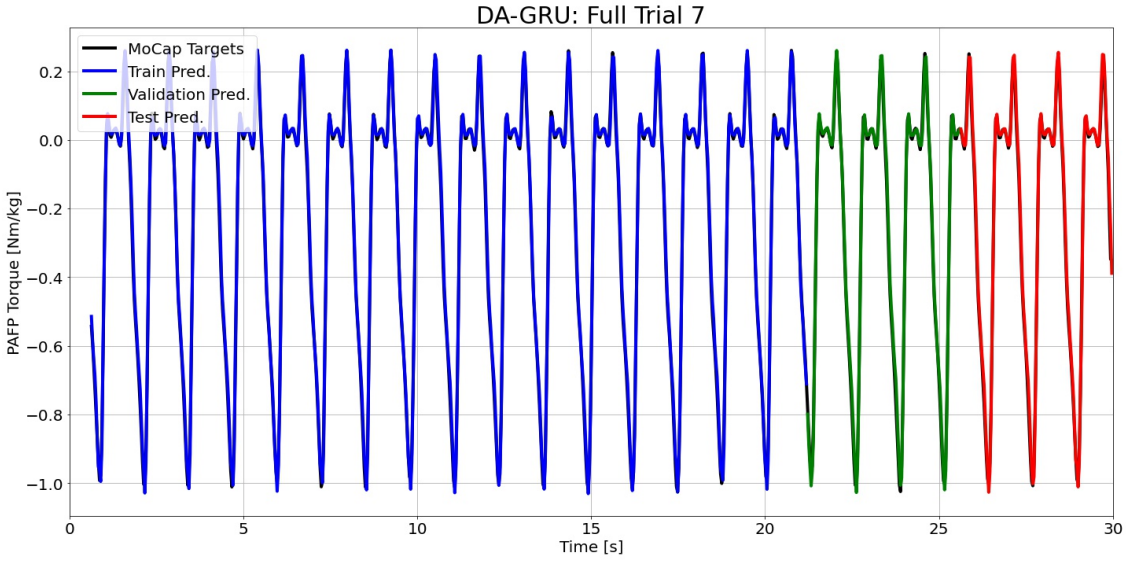


Figure H.109: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 7.

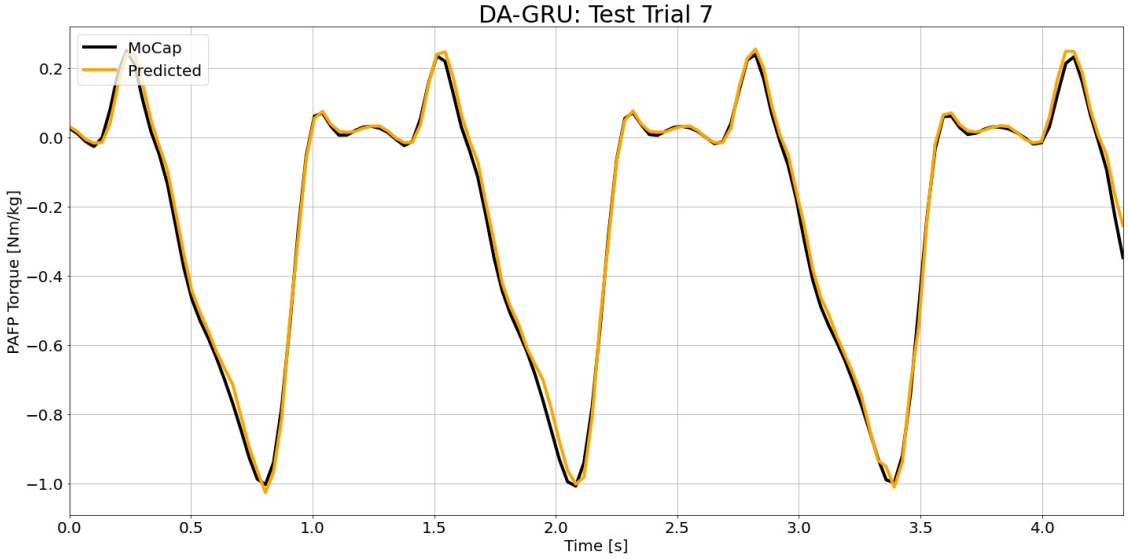


Figure H.110: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 7.

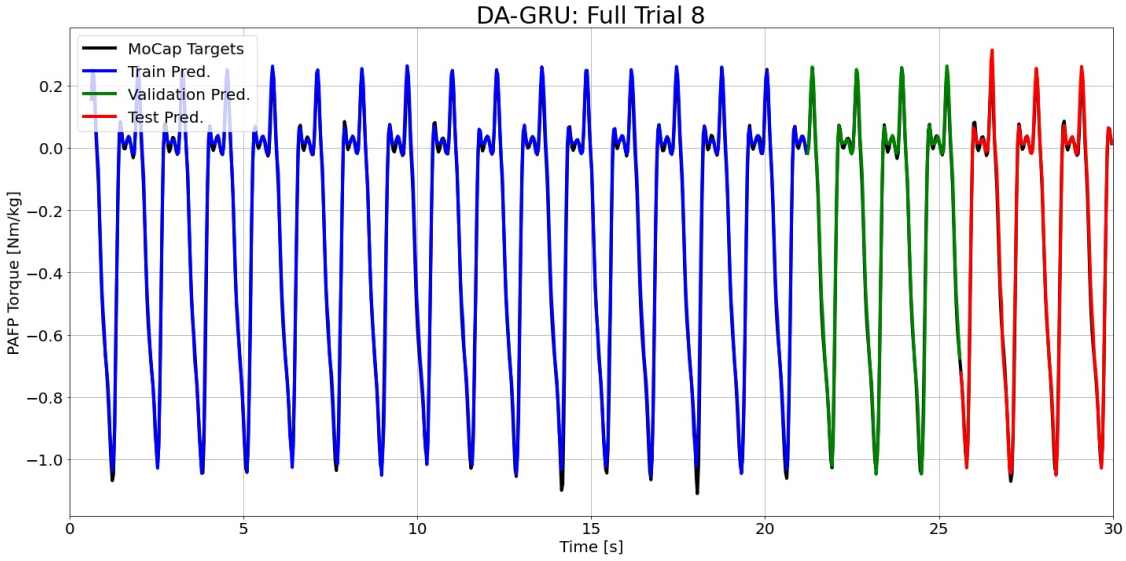


Figure H.111: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 8.

Figure H.112: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 8.

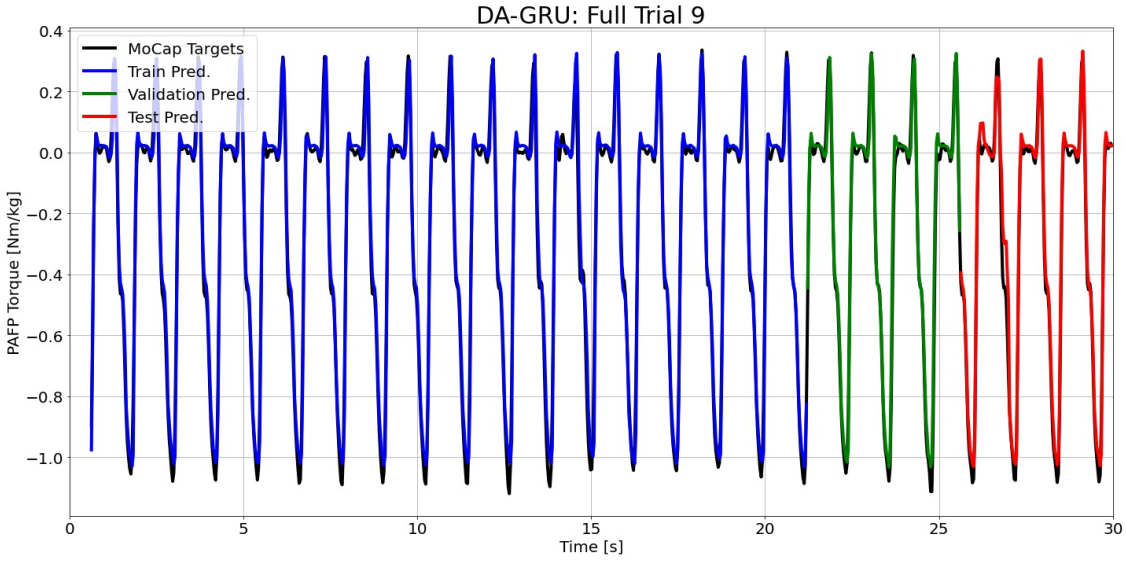


Figure H.113: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 9.

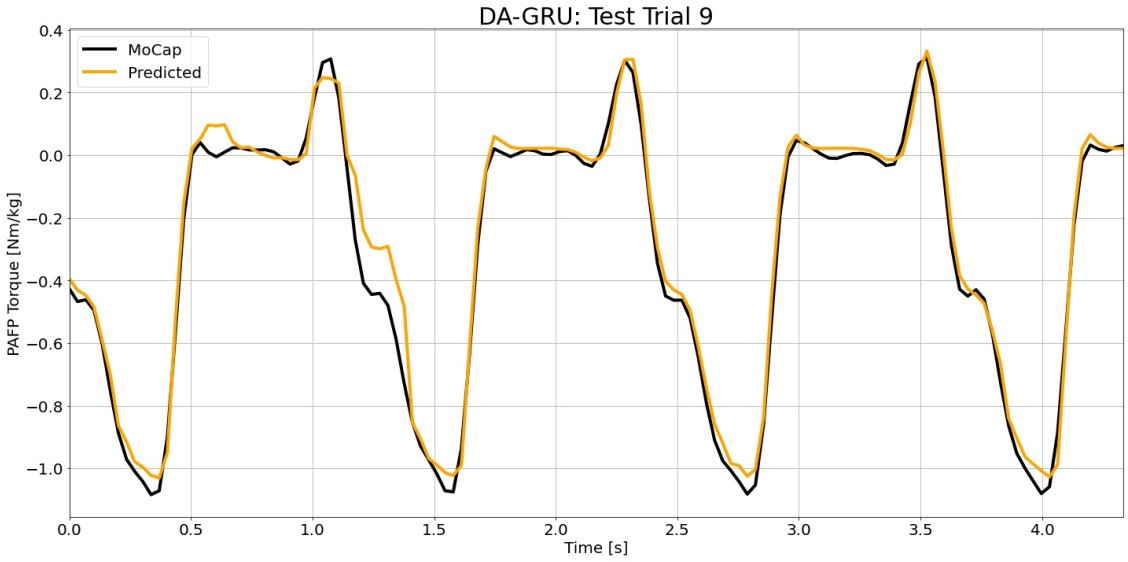


Figure H.114: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 9.

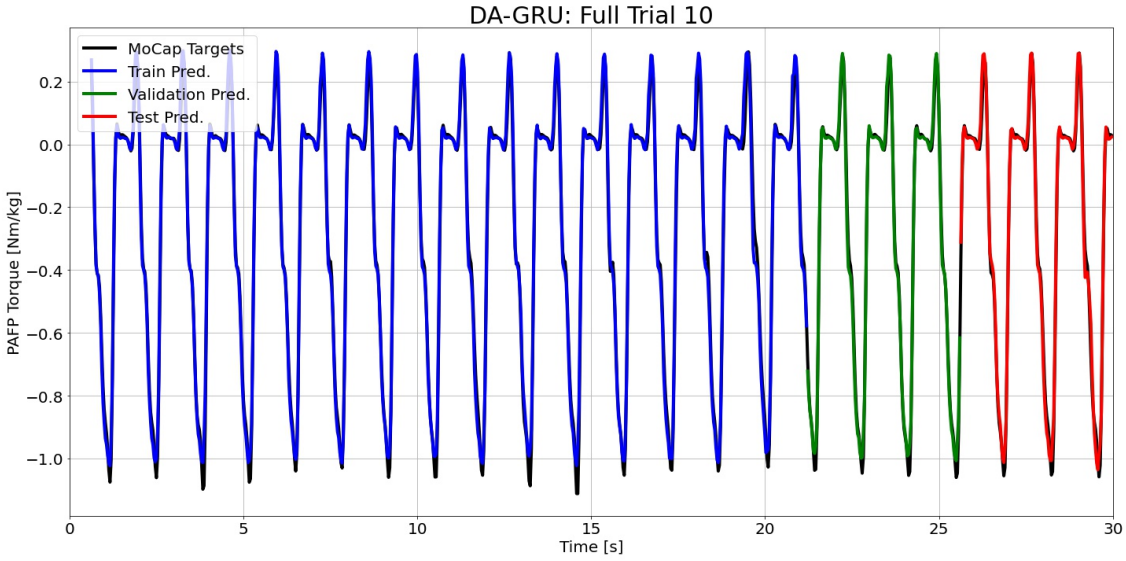


Figure H.115: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 10.

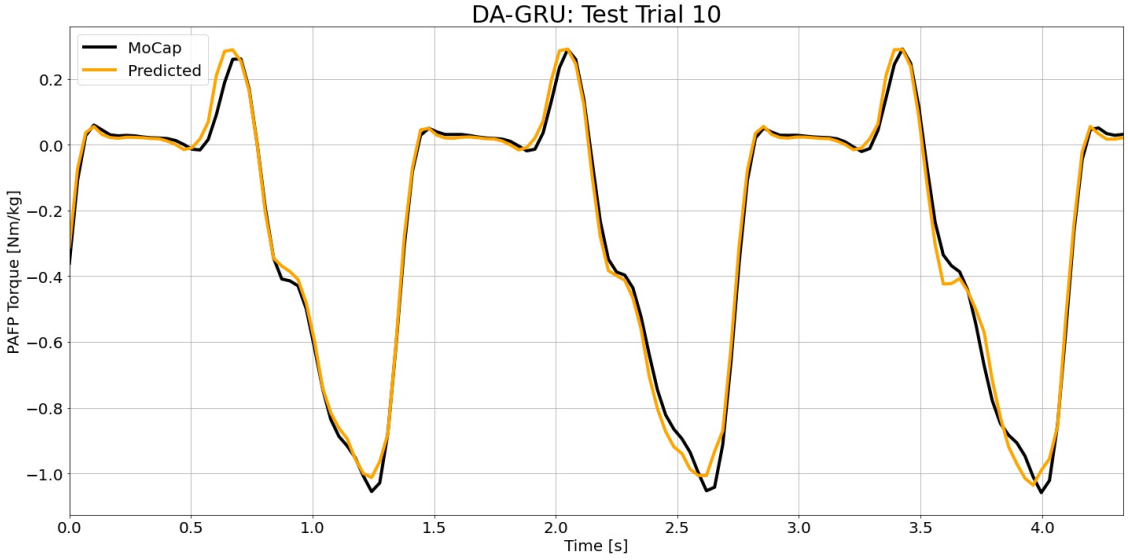


Figure H.116: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 10.

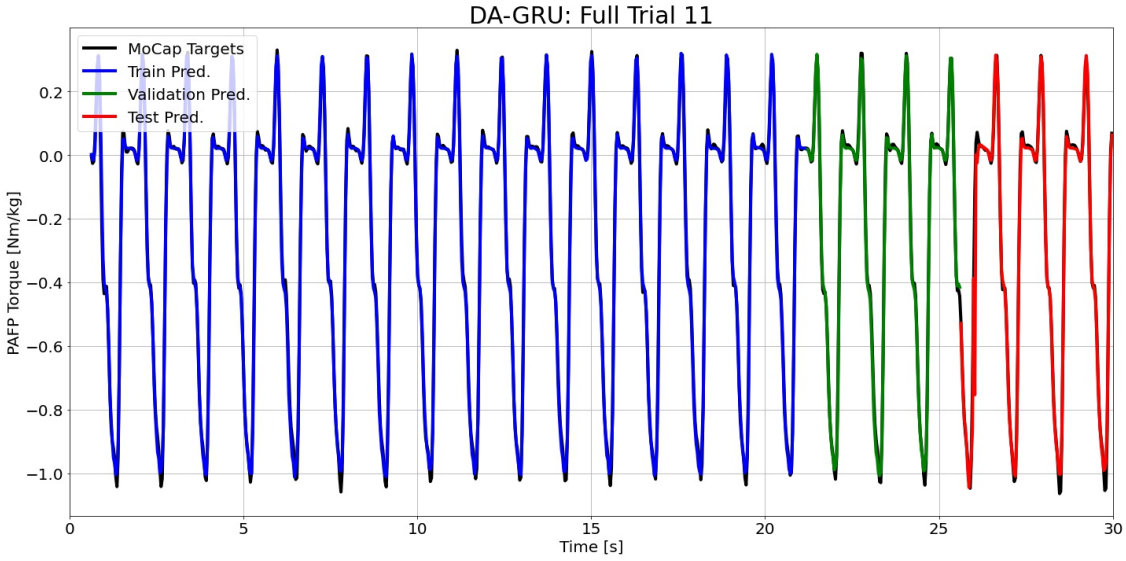


Figure H.117: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 11.

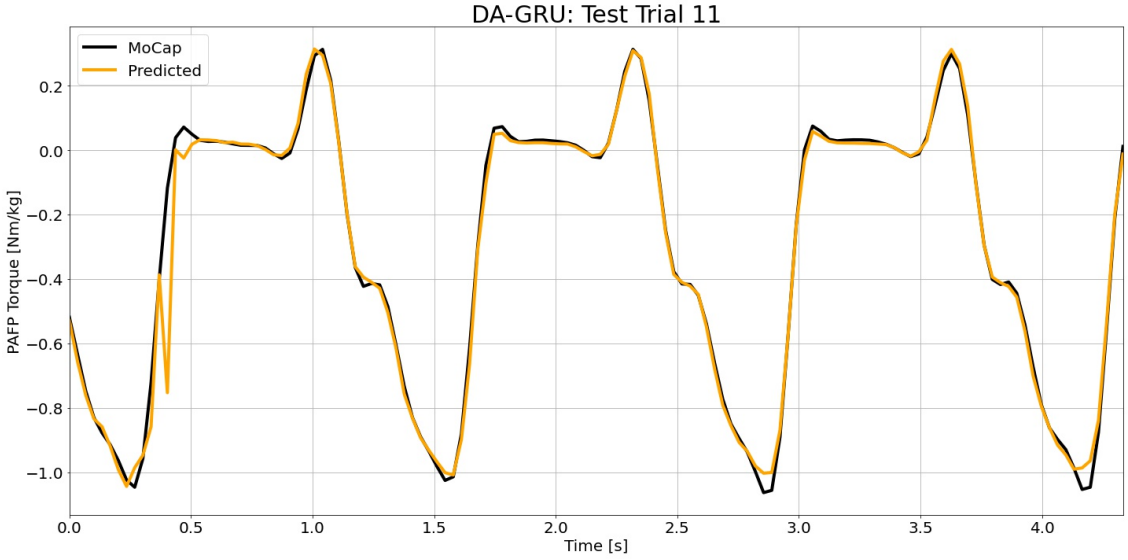


Figure H.118: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 11.

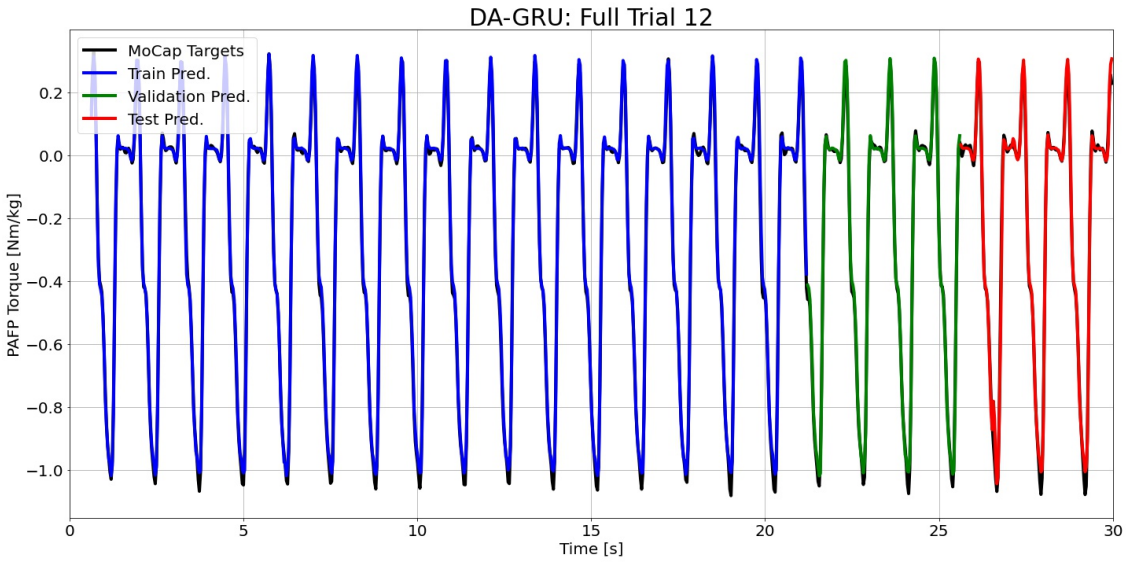


Figure H.119: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 12.

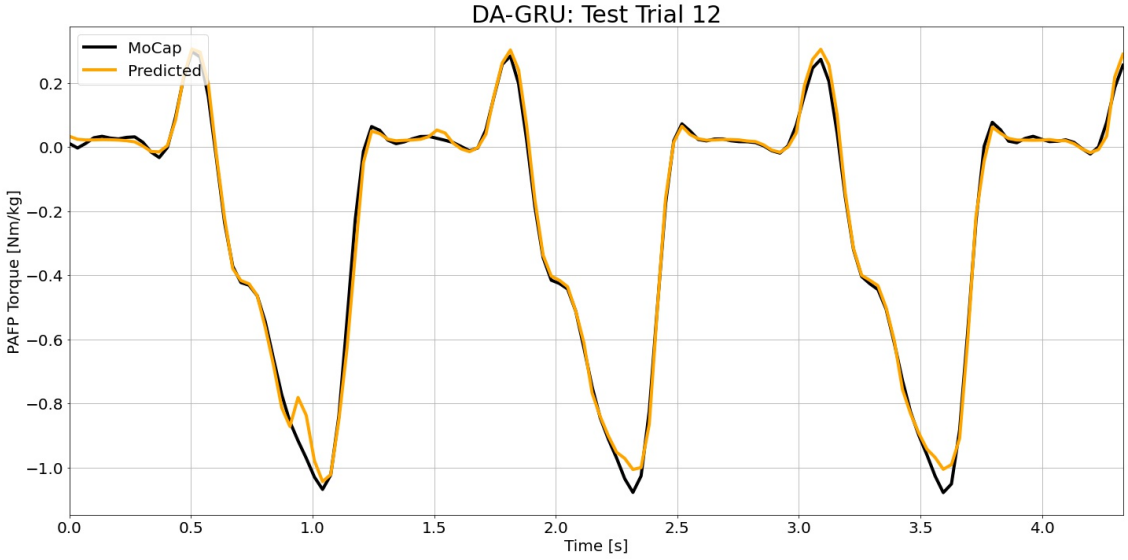


Figure H.120: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 12.

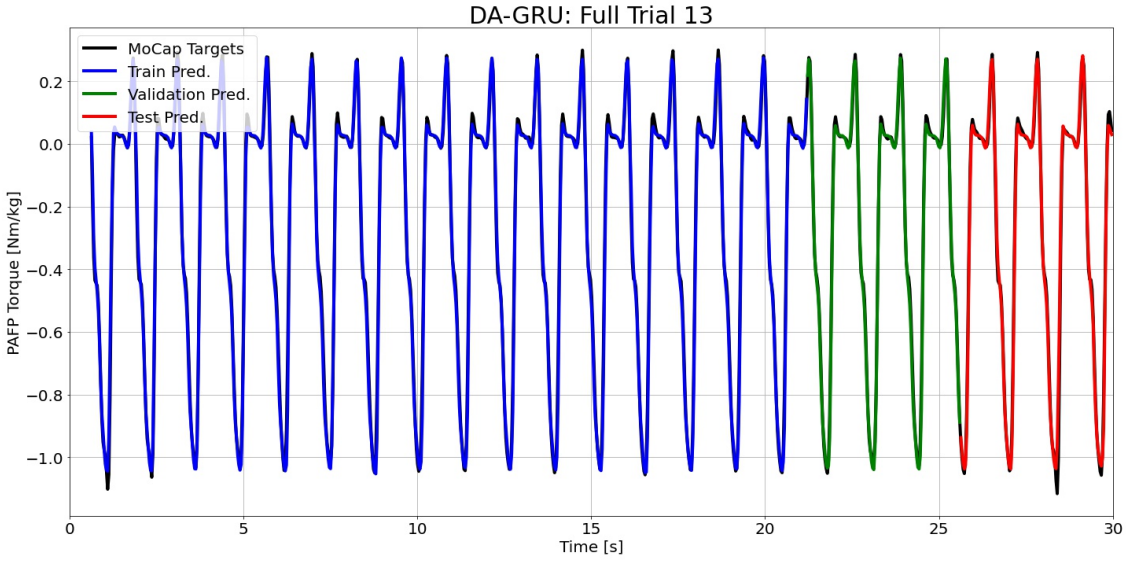


Figure H.121: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 13.

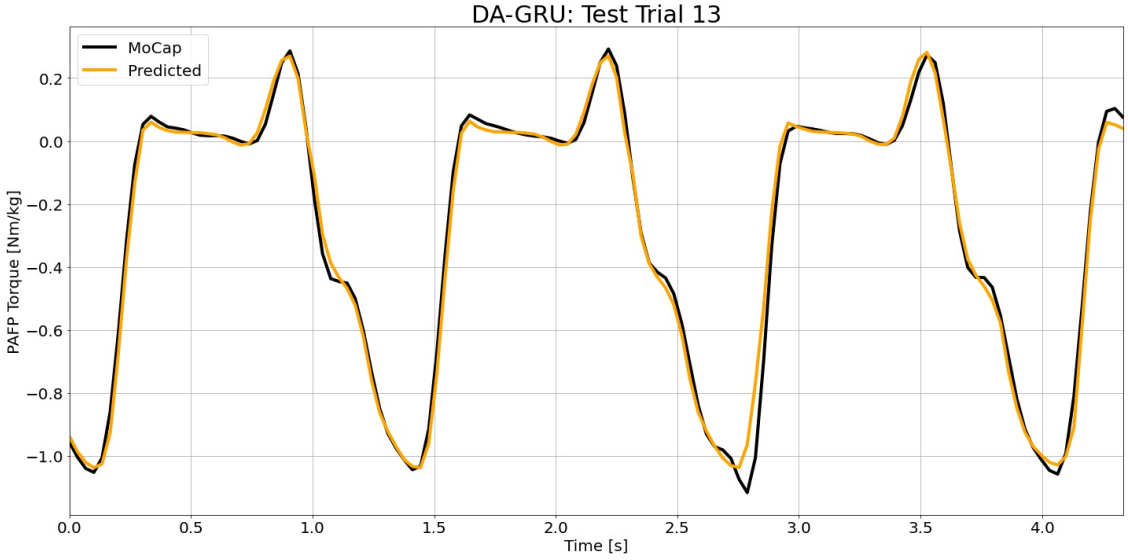


Figure H.122: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 13.

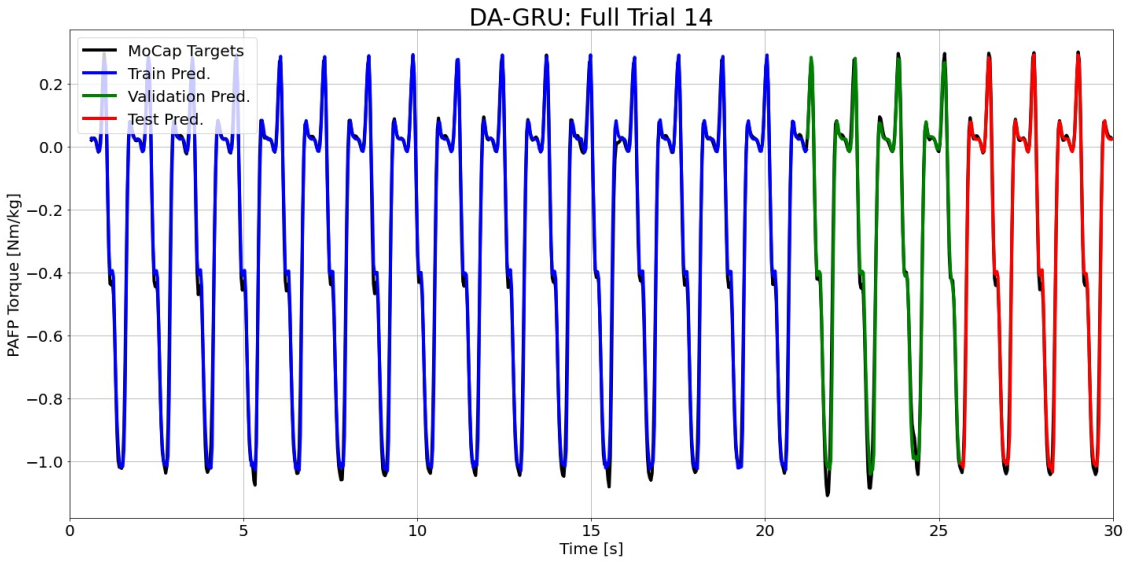


Figure H.123: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 14.

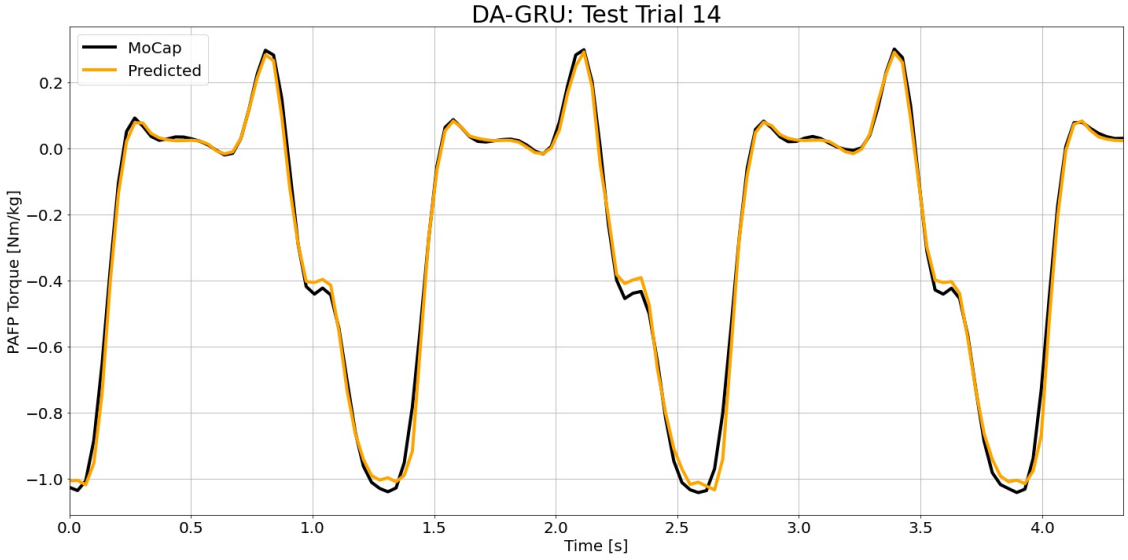


Figure H.124: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 14.

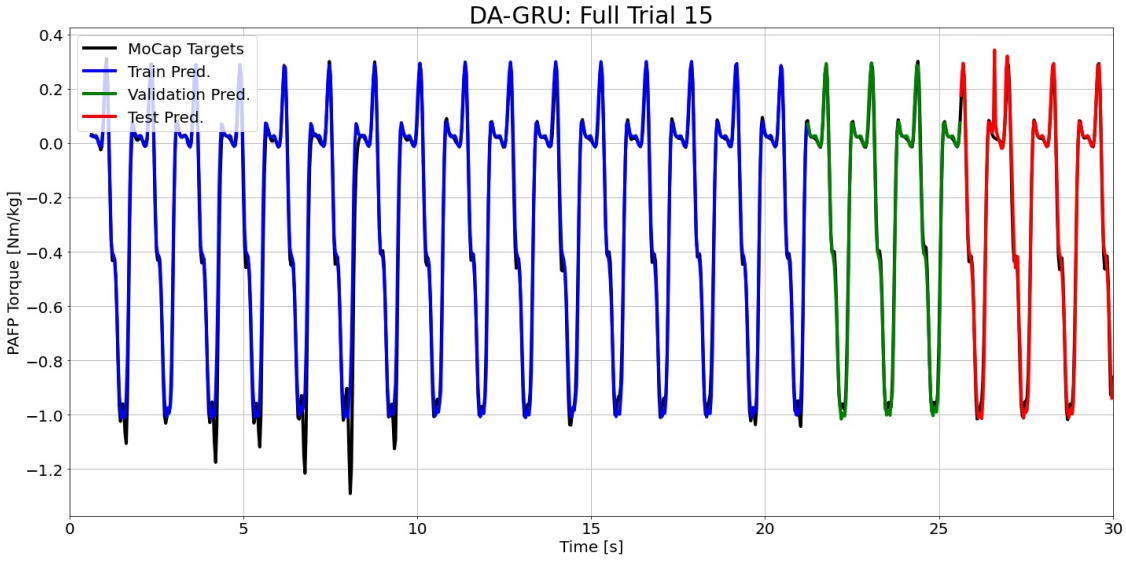


Figure H.125: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 15.

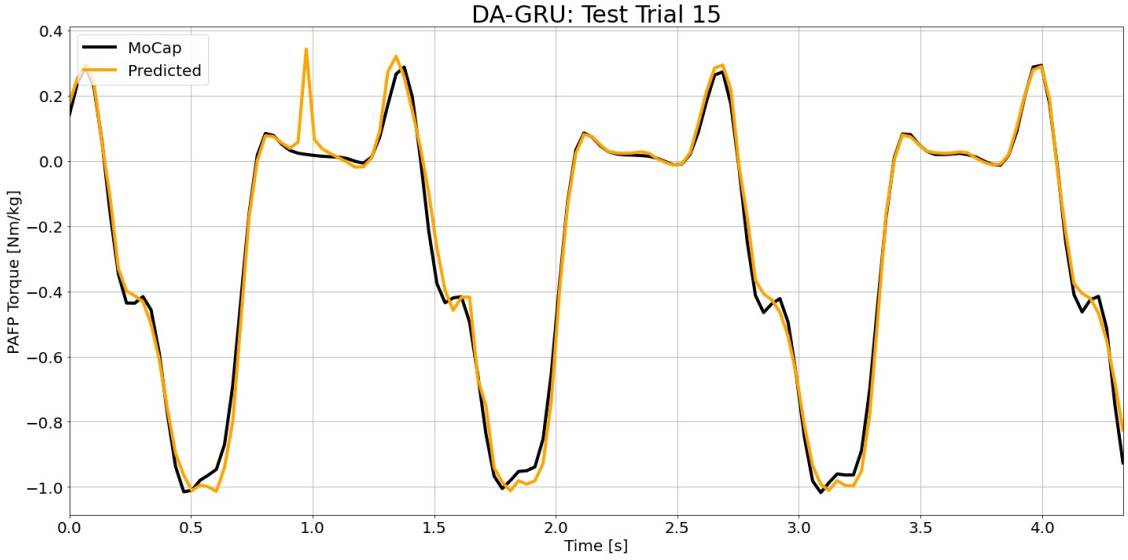


Figure H.126: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 15.

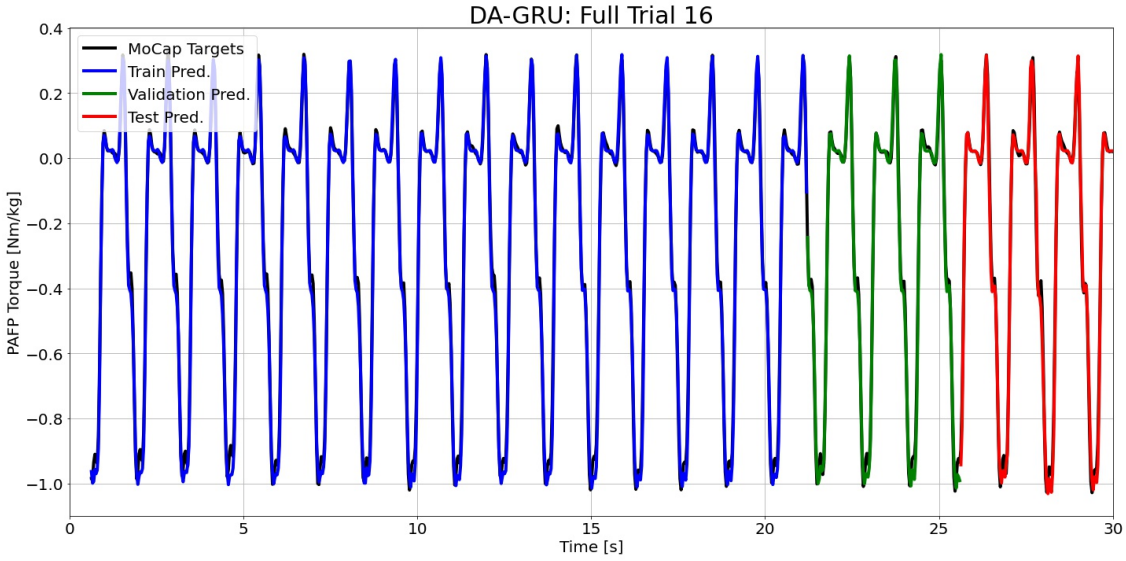


Figure H.127: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 16.

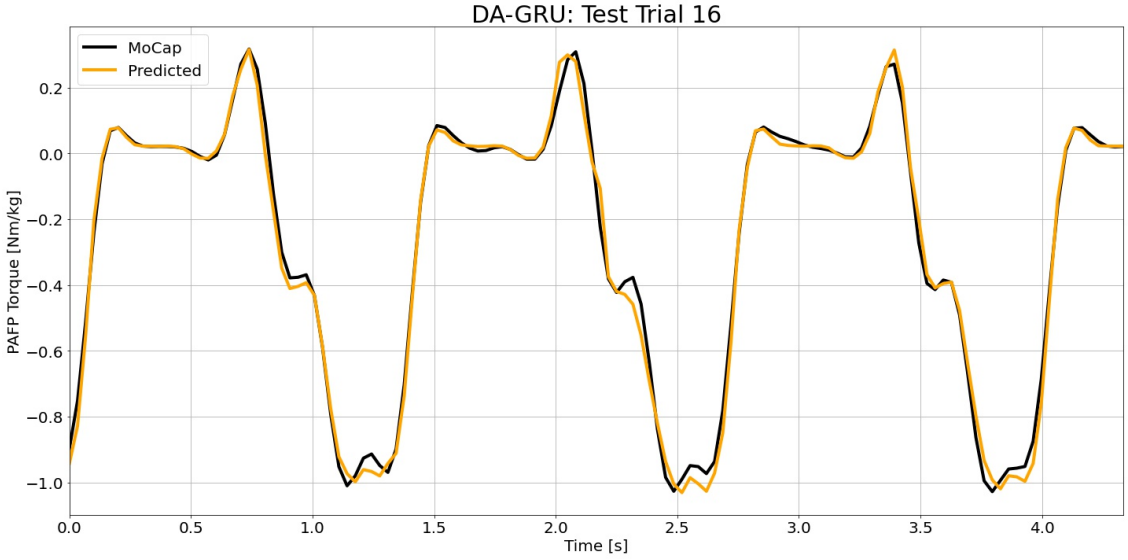


Figure H.128: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 16.

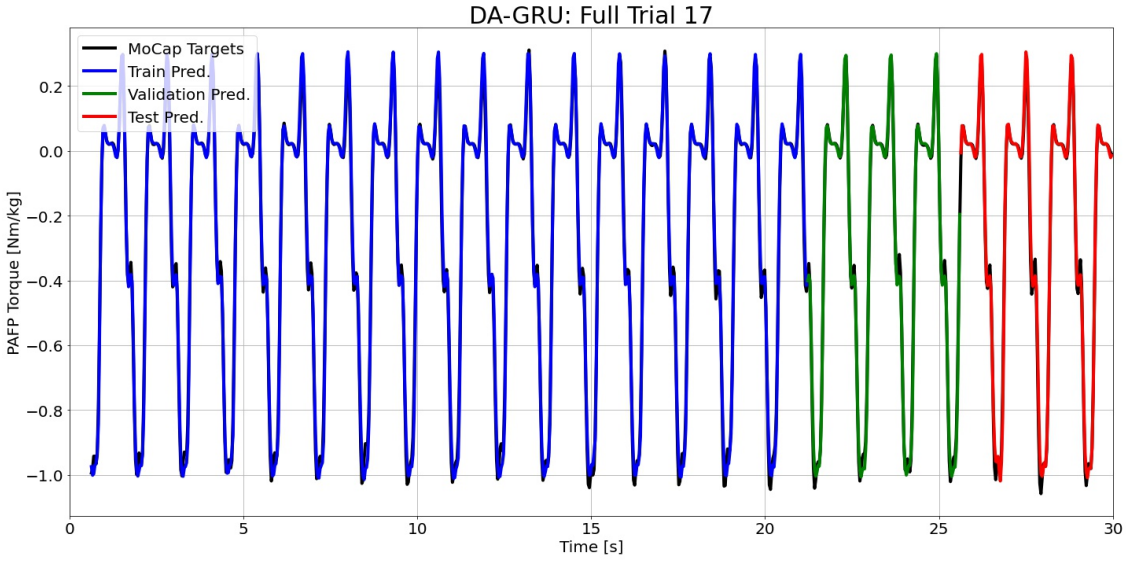


Figure H.129: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 17.

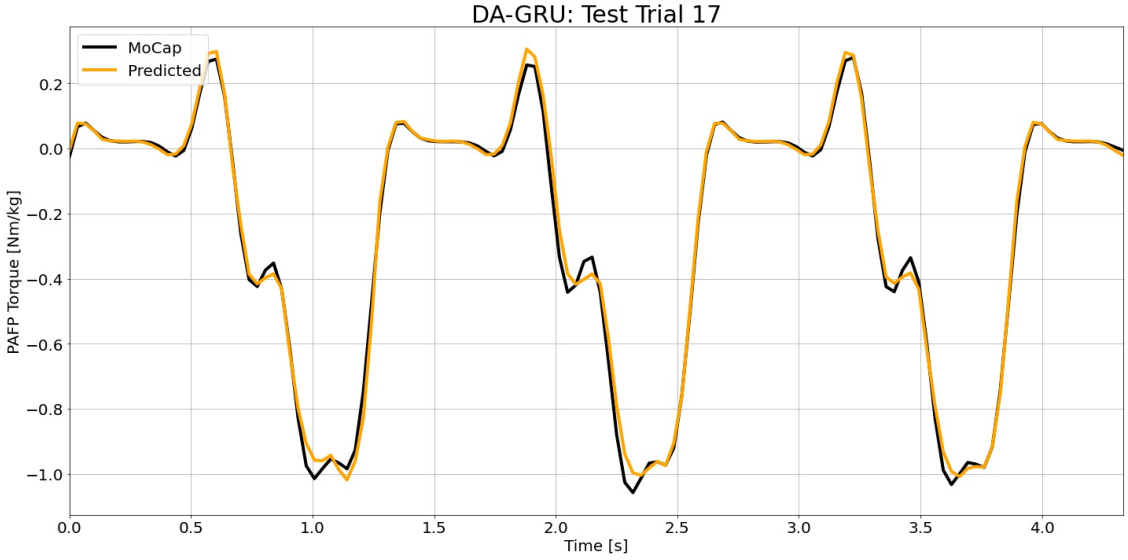


Figure H.130: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 17.

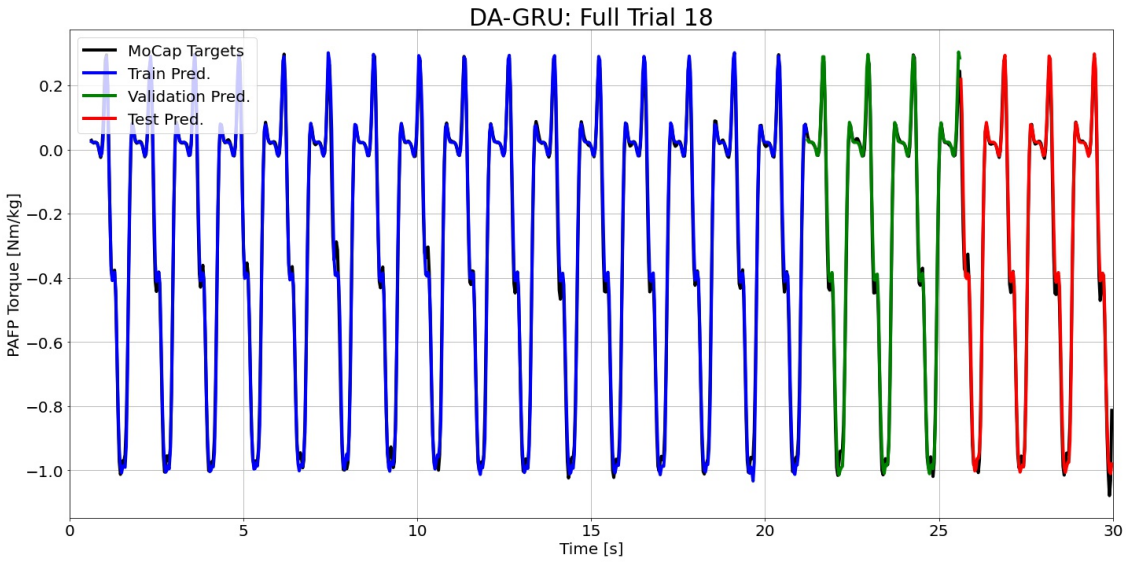


Figure H.131: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 18.

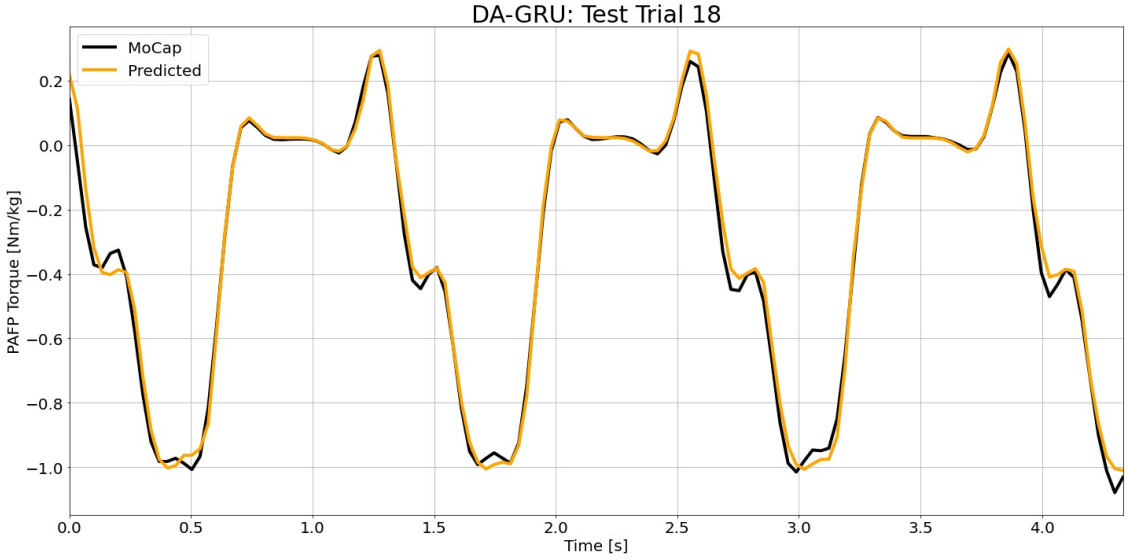


Figure H.132: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 18.

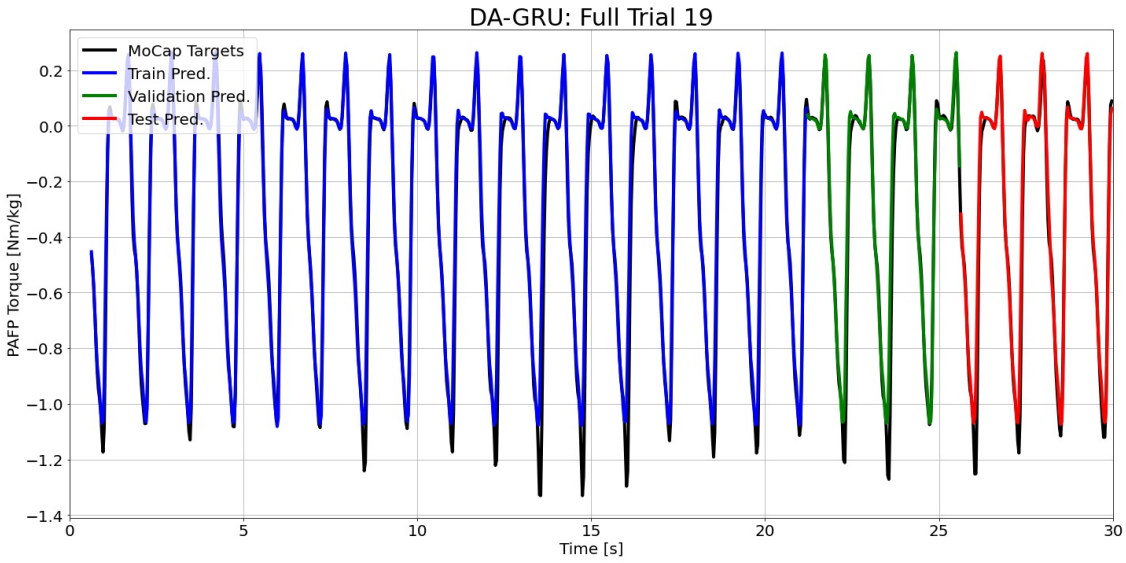


Figure H.133: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 19.

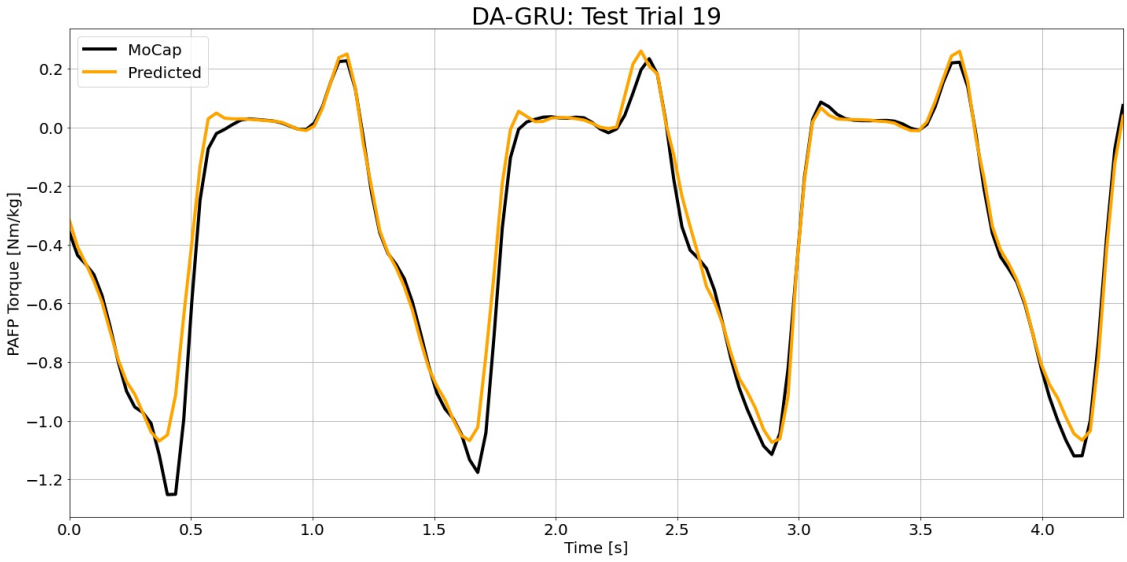


Figure H.134: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 19.

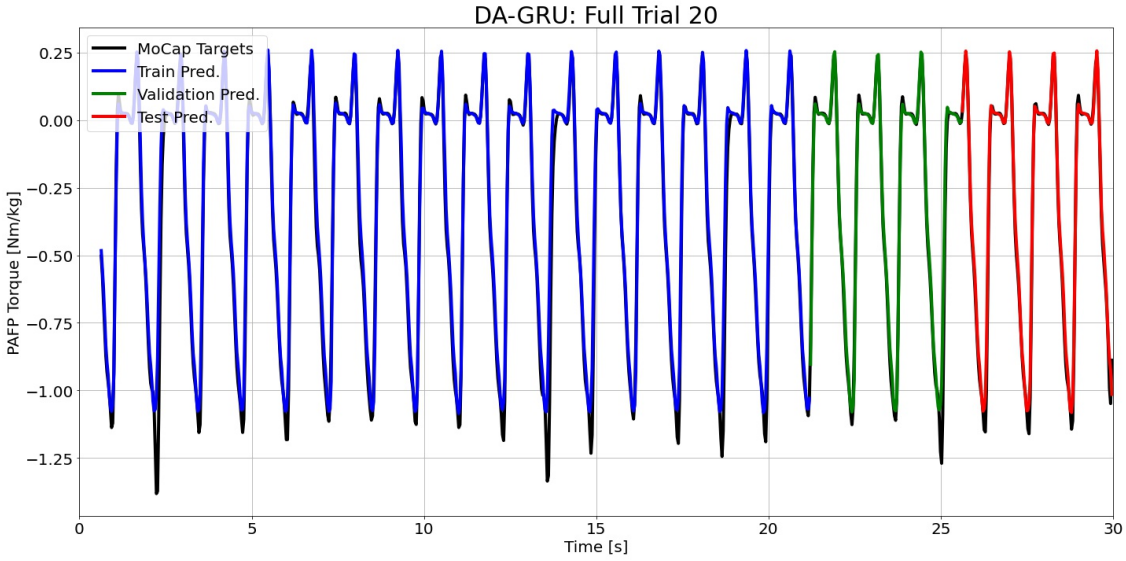


Figure H.135: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 20.

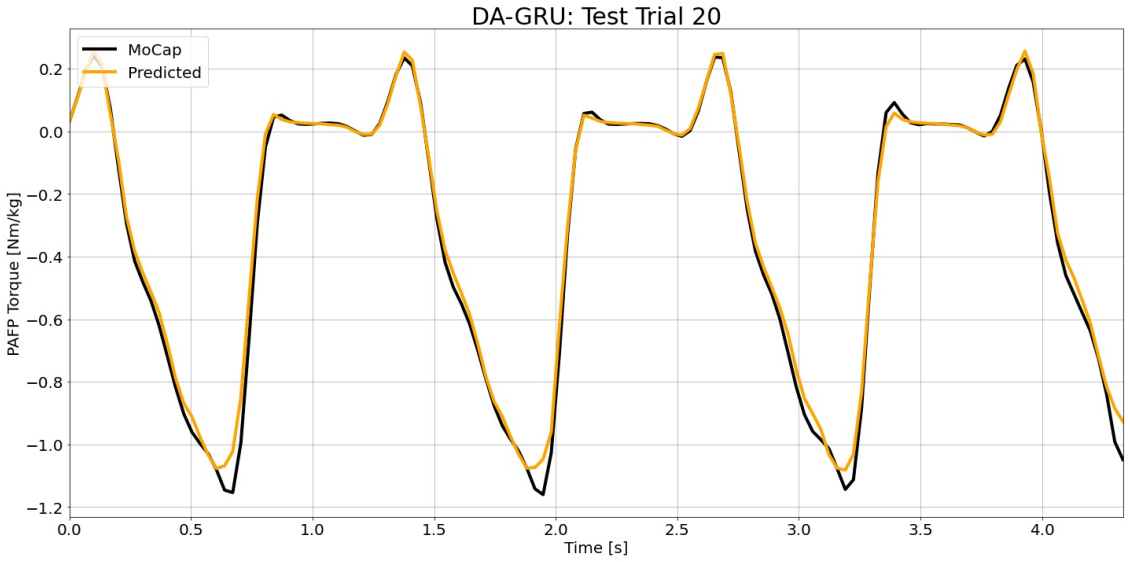


Figure H.136: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 20.

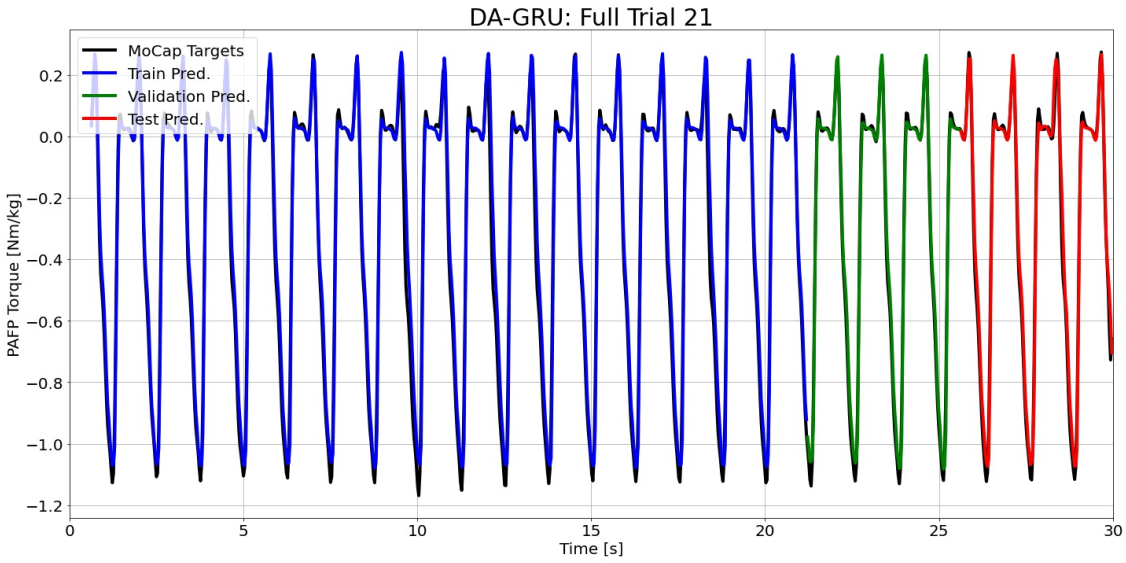


Figure H.137: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 21.

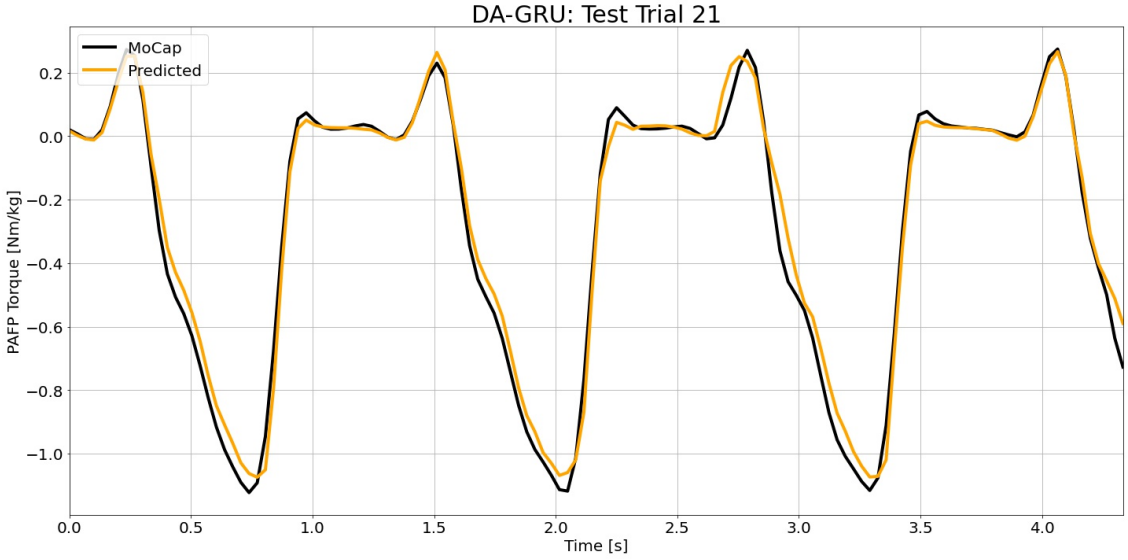


Figure H.138: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 21.

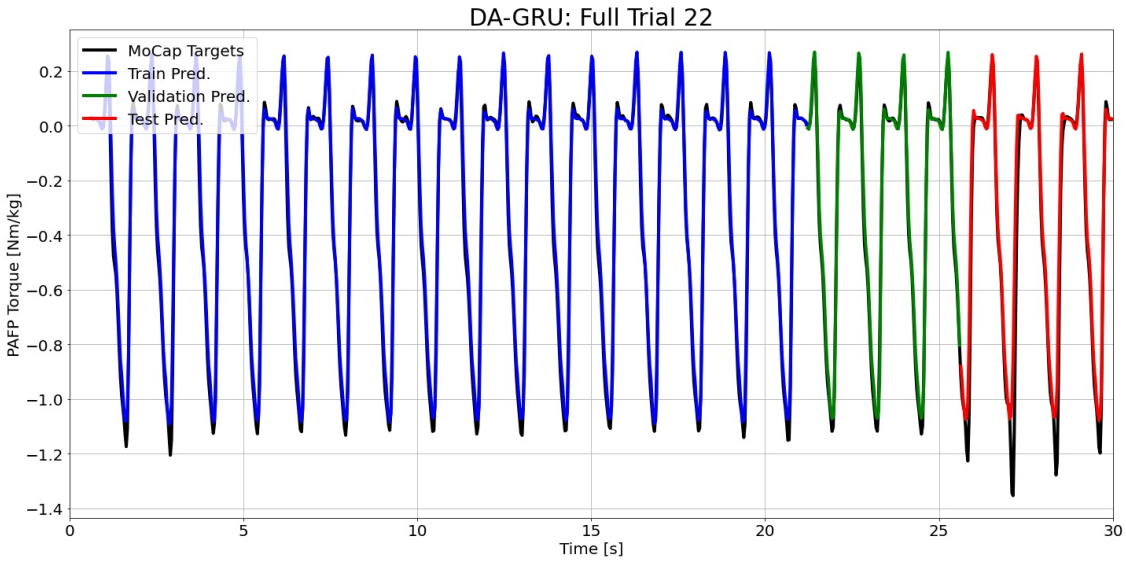


Figure H.139: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 22.

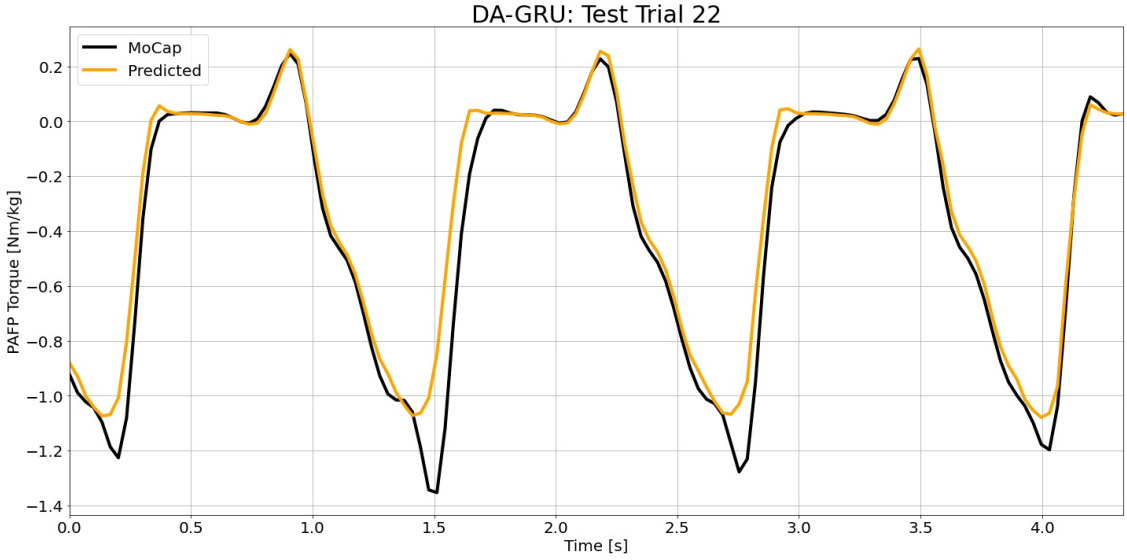


Figure H.140: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 22.

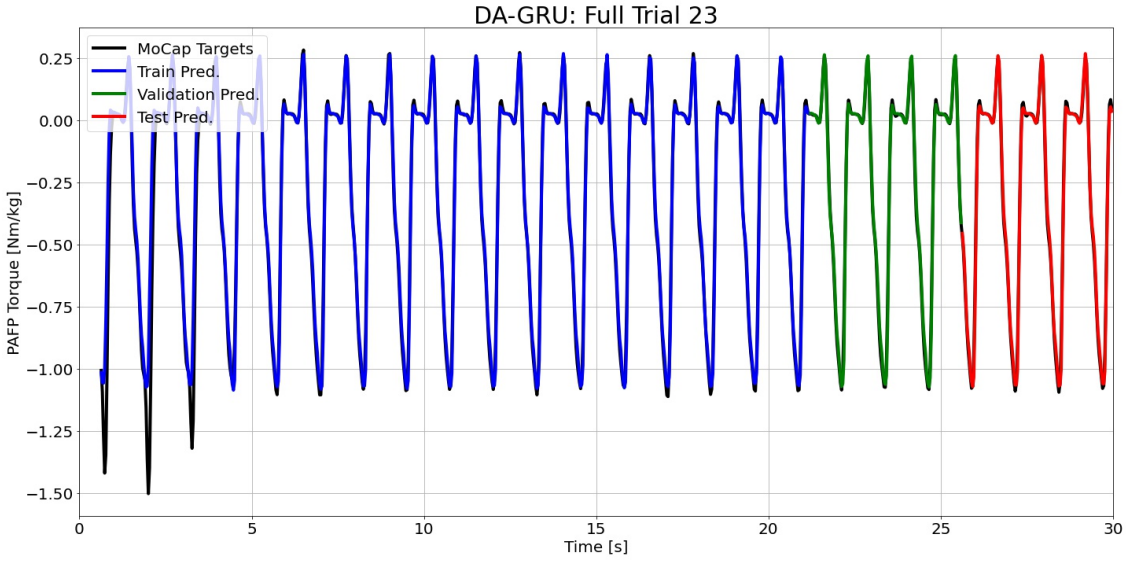


Figure H.141: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 23.

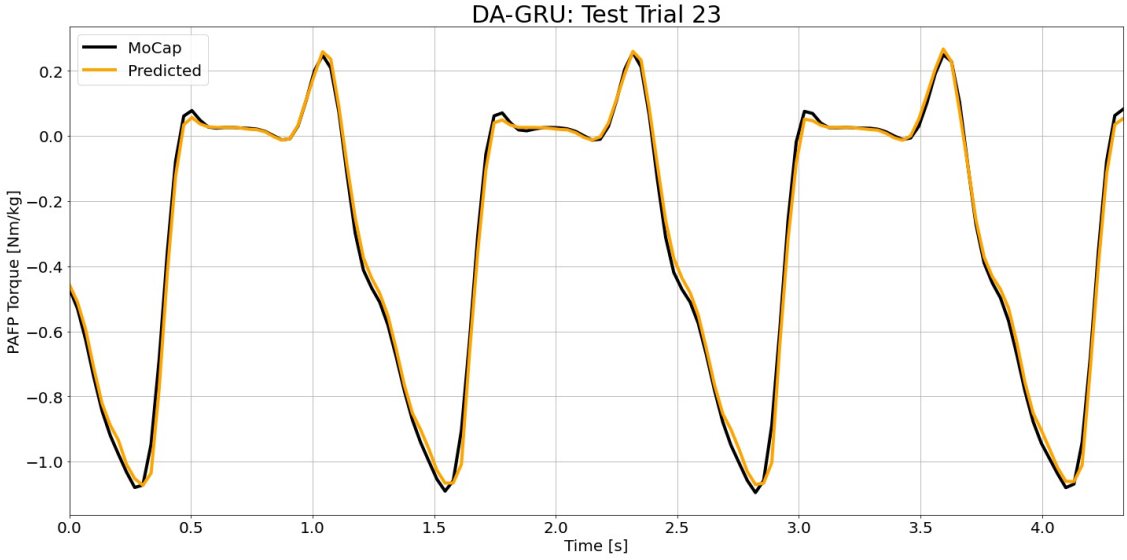


Figure H.142: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 23.

H.4 Analytical Regression Model

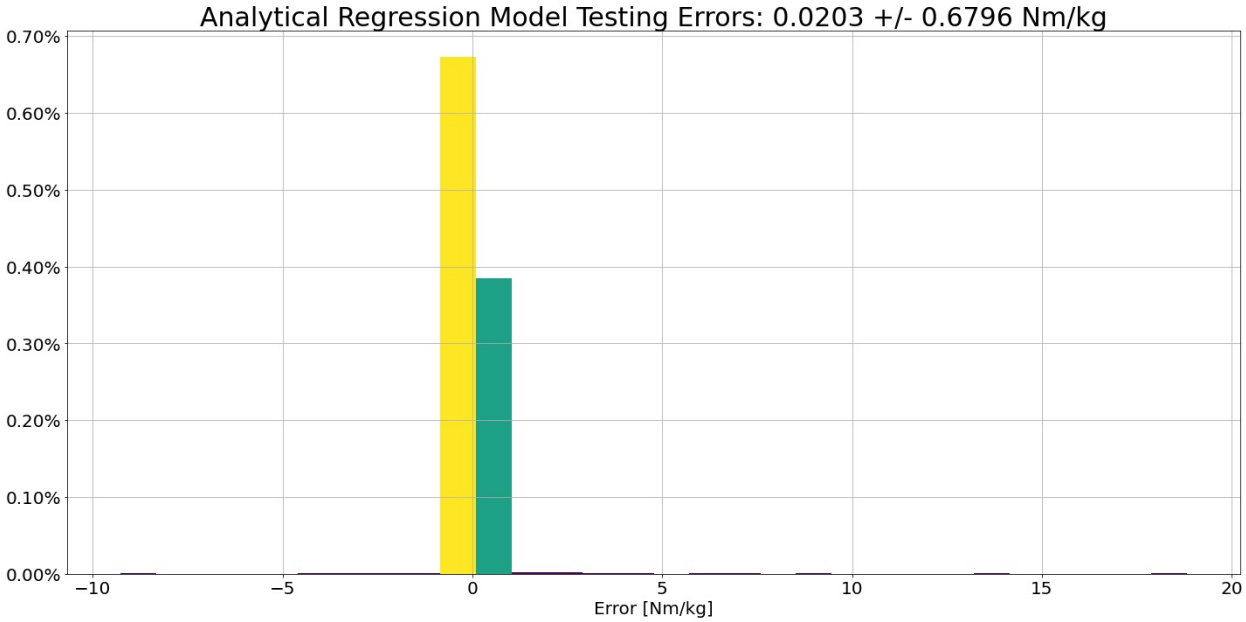


Figure H.143: Analytical regression model prediction error histogram for all test time series data.

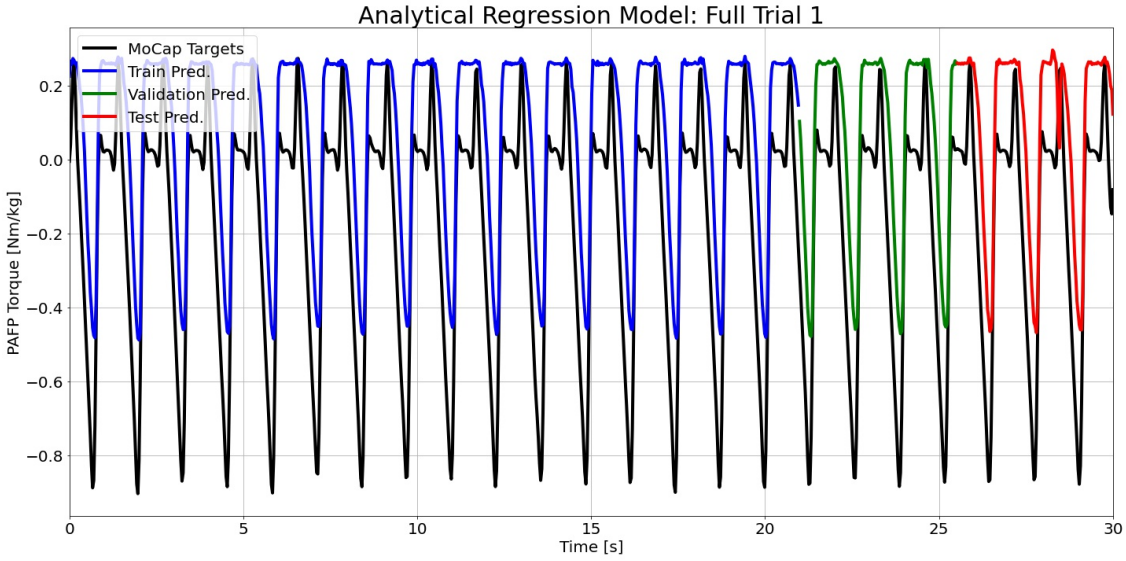


Figure H.144: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 1.

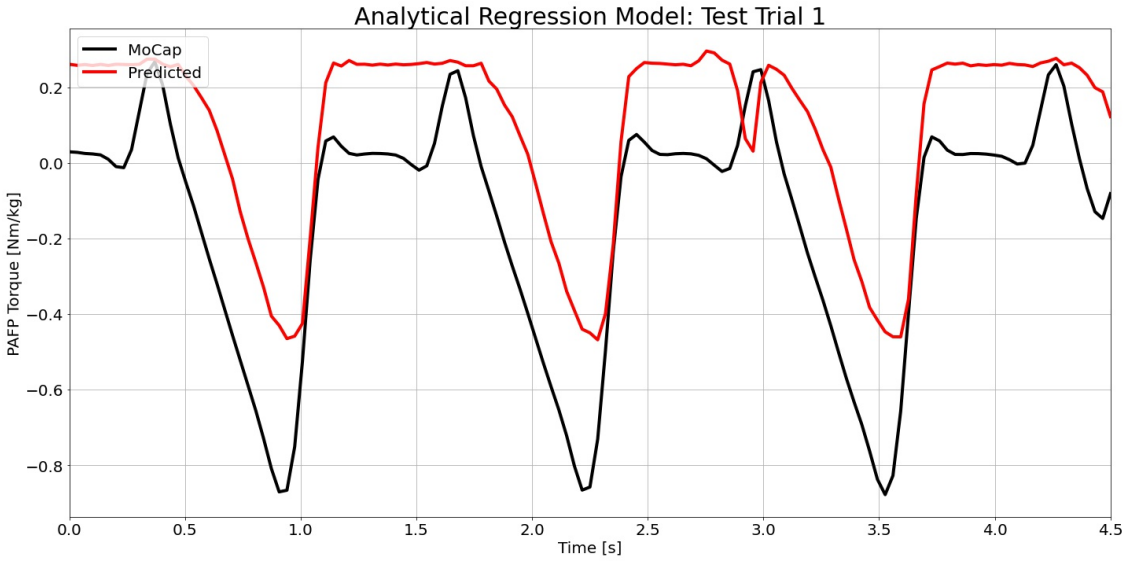


Figure H.145: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 1.

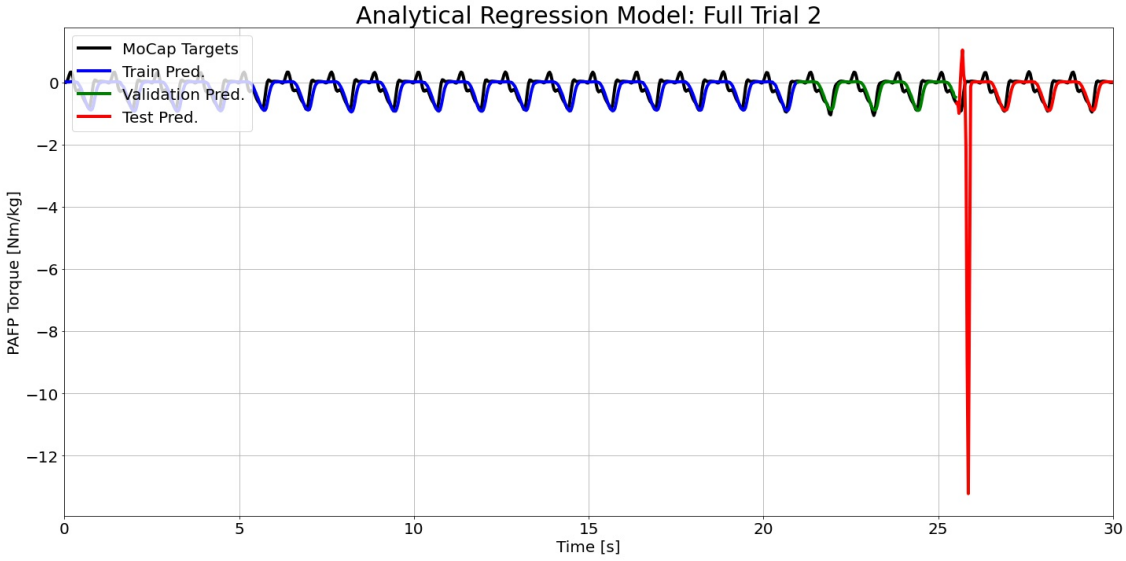


Figure H.146: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 2.

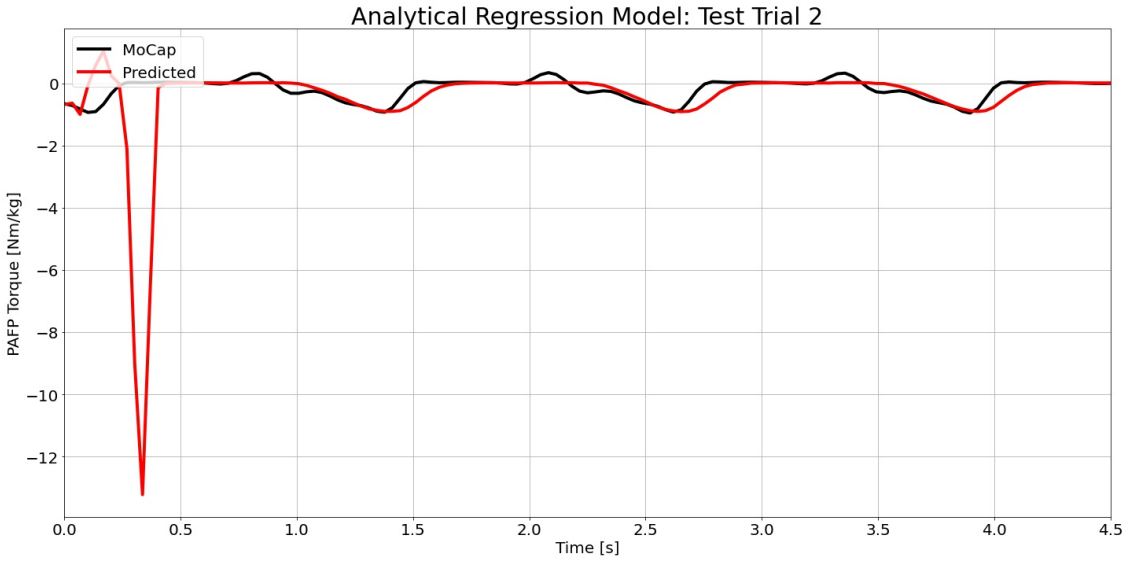


Figure H.147: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 2.

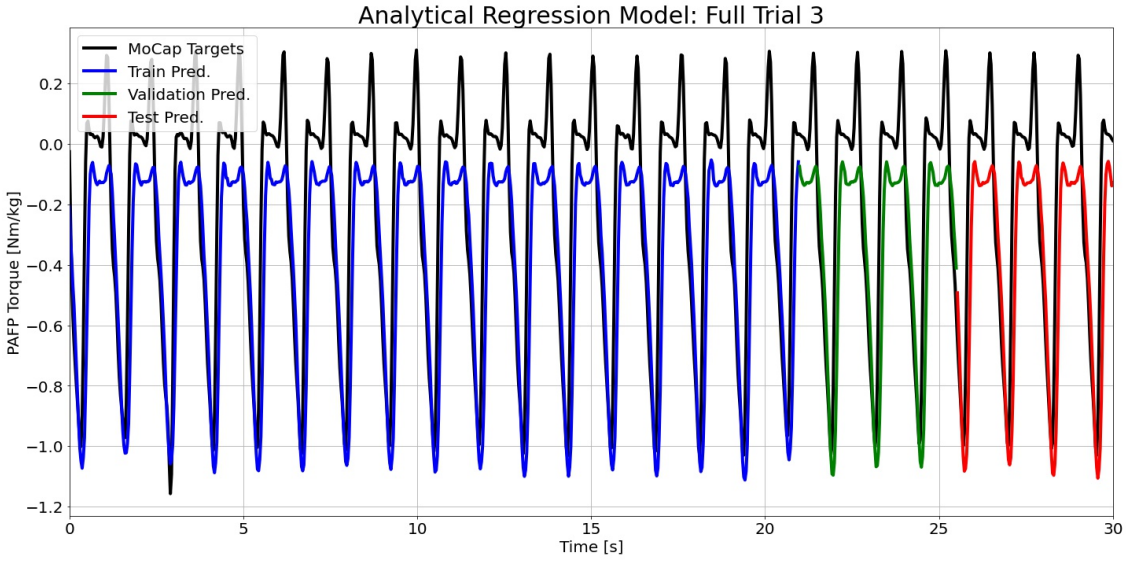


Figure H.148: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 3.

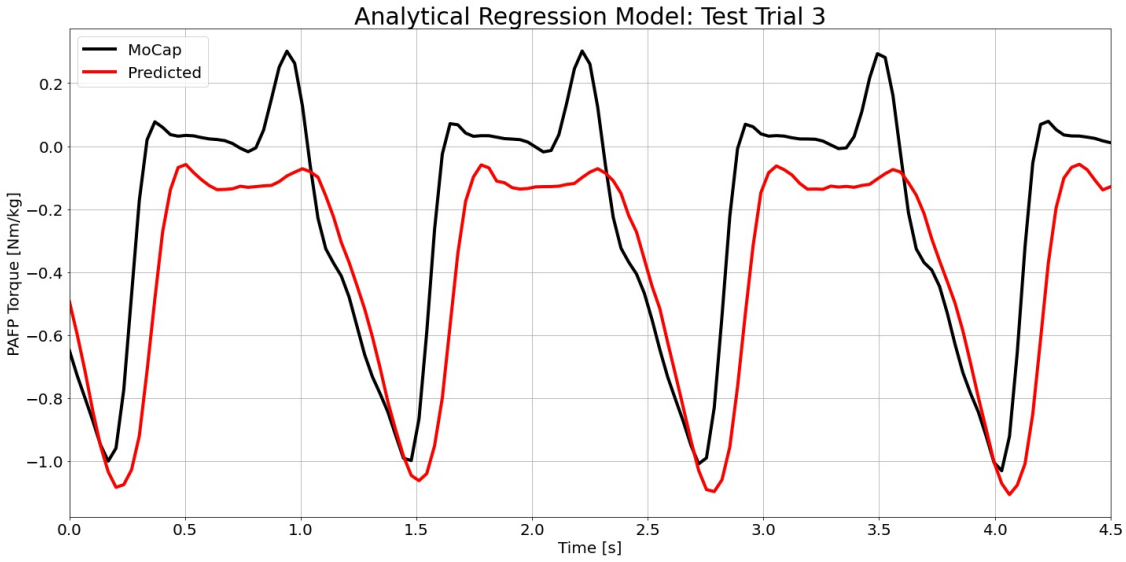


Figure H.149: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 3.

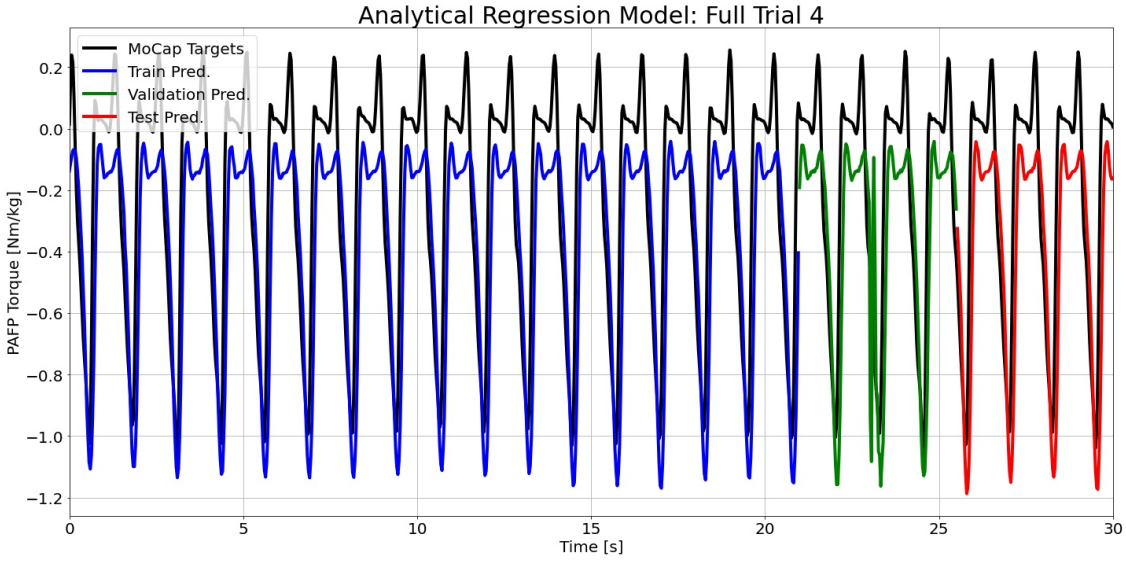


Figure H.150: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 4.

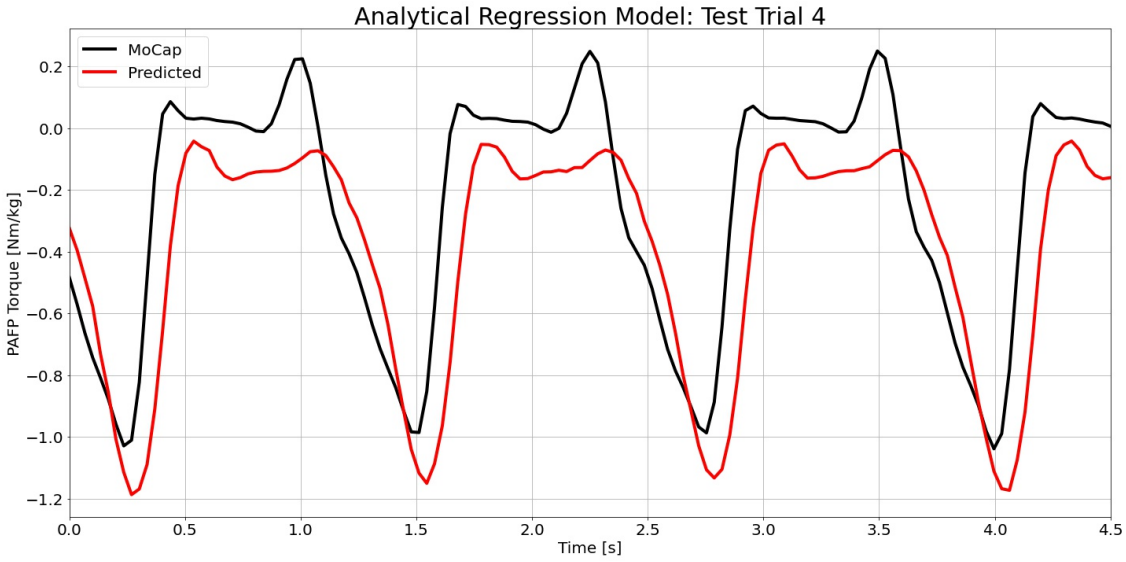


Figure H.151: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 4.

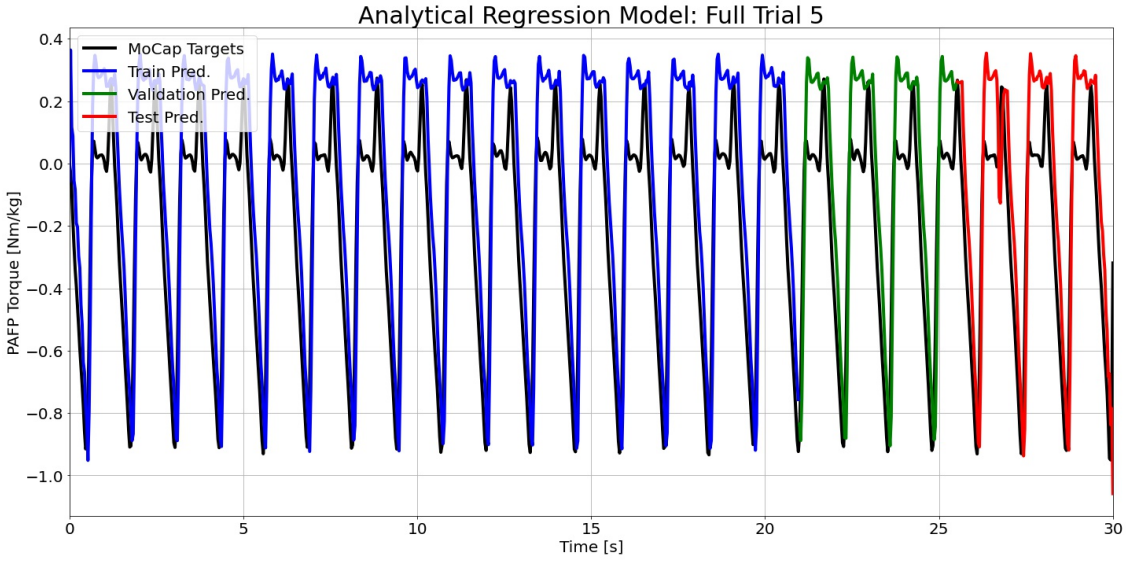


Figure H.152: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 5.

Figure H.153: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 5.

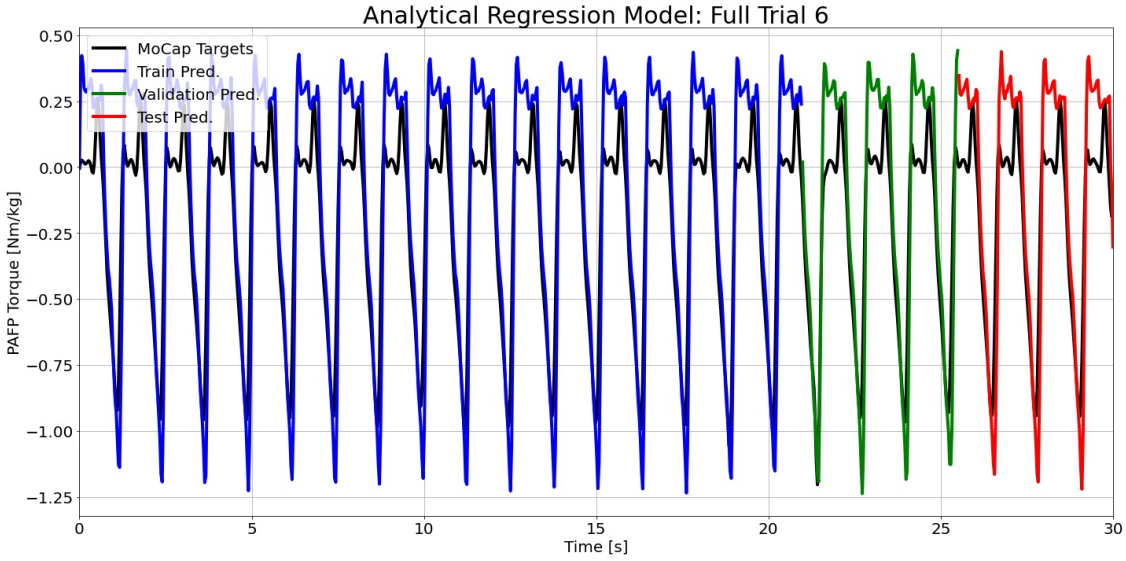


Figure H.154: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 6.

Figure H.155: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 6.

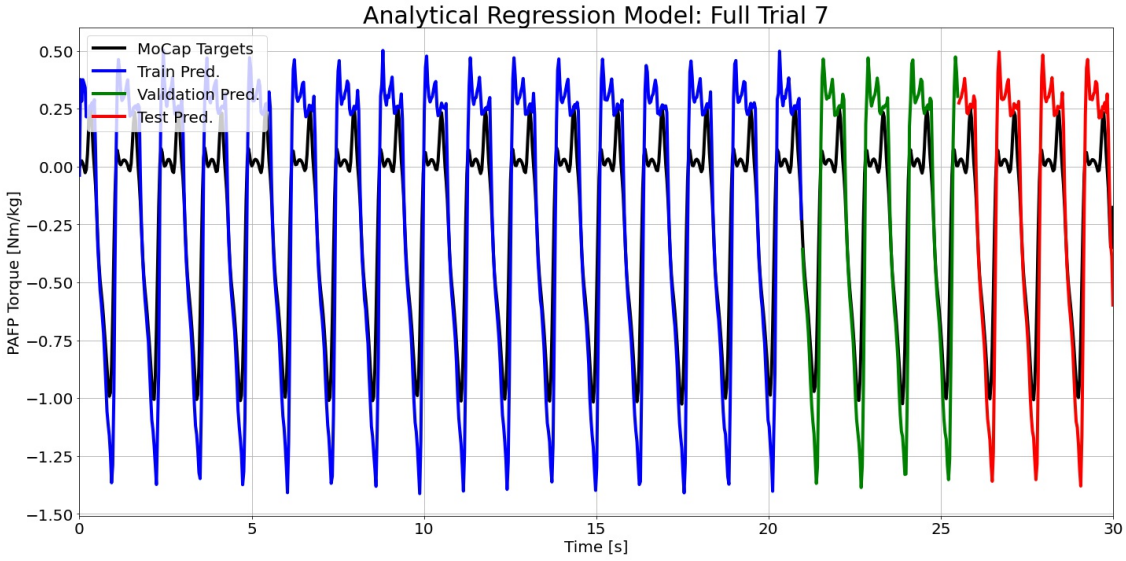


Figure H.156: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 7.

Figure H.157: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 7.

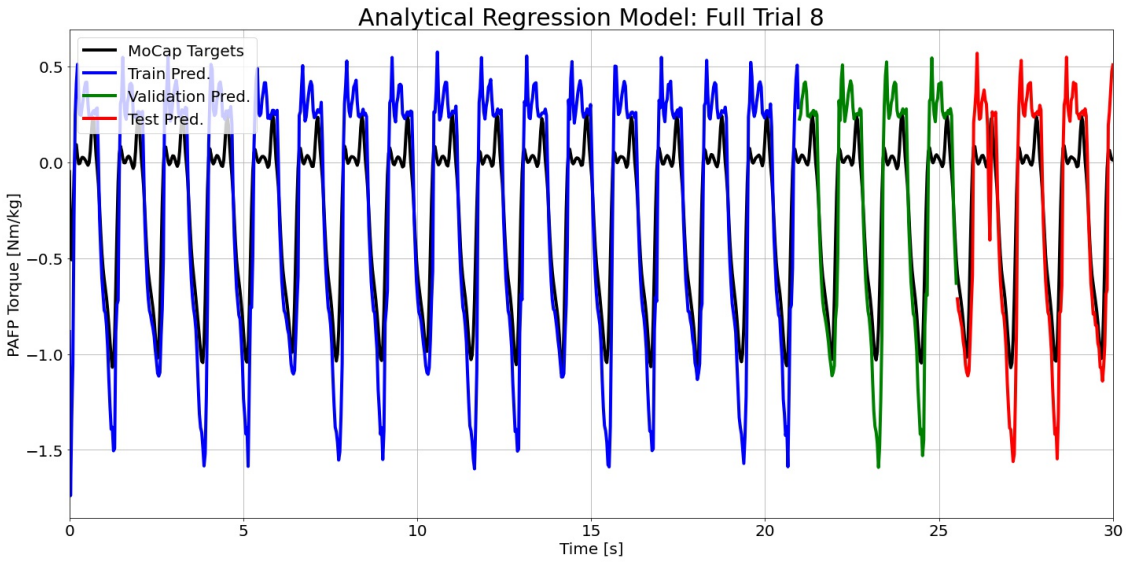


Figure H.158: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 8.

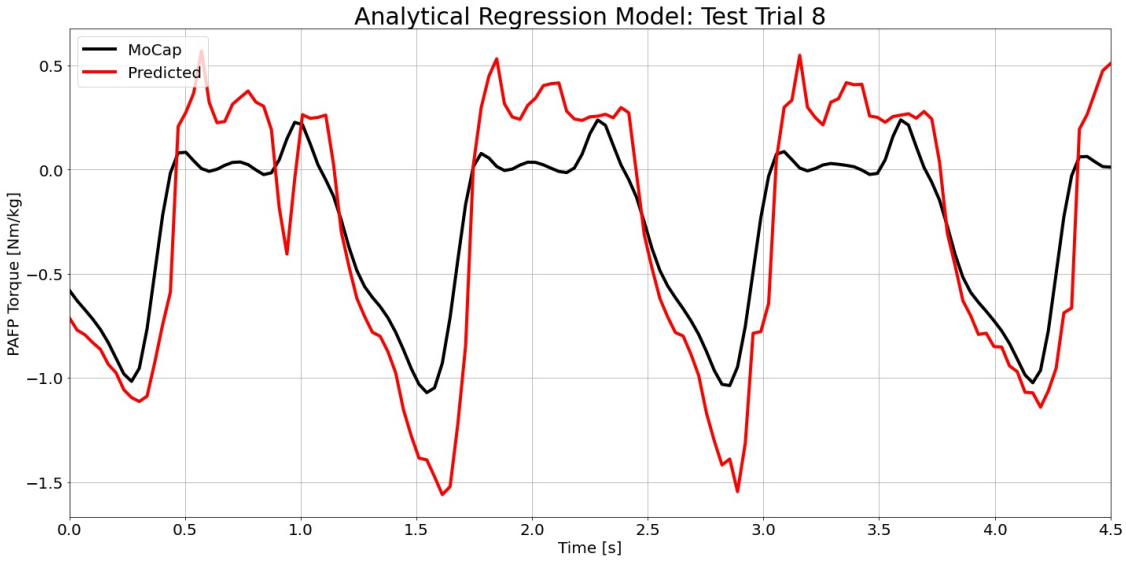


Figure H.159: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 8.

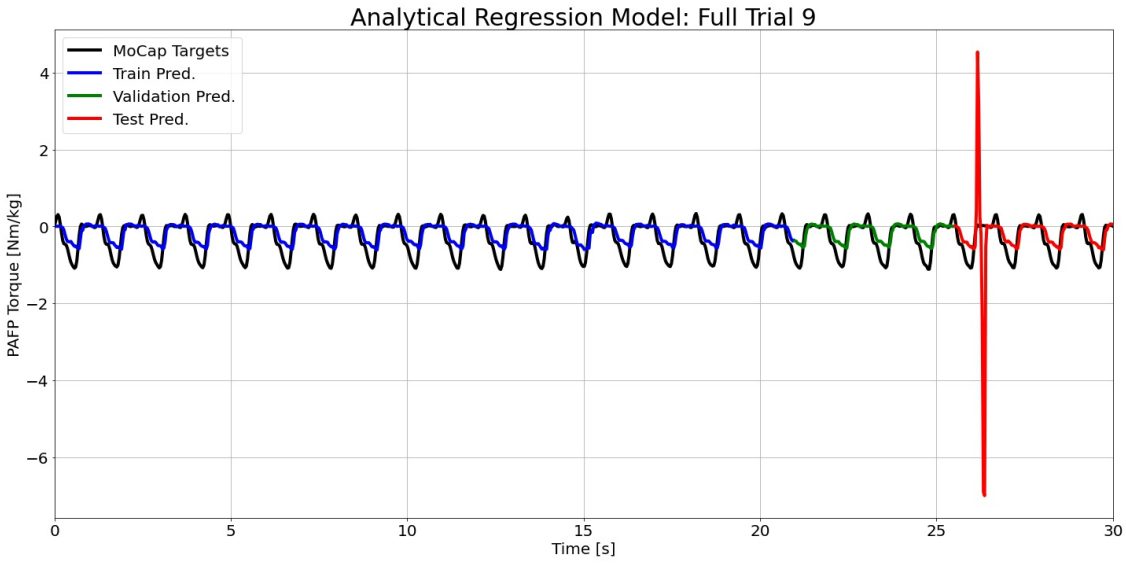


Figure H.160: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 9.

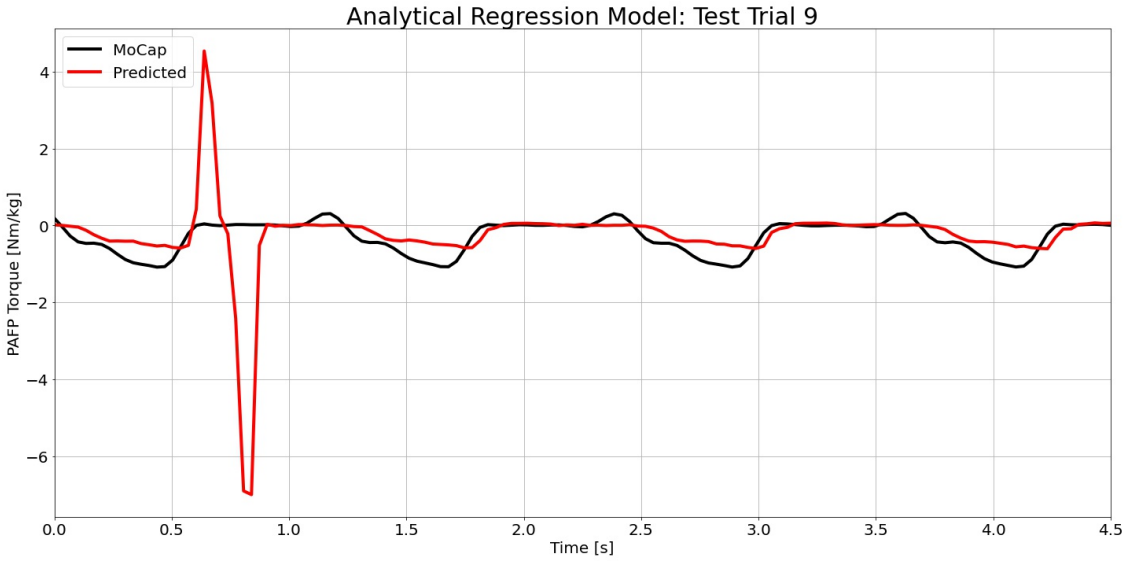


Figure H.161: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 9.

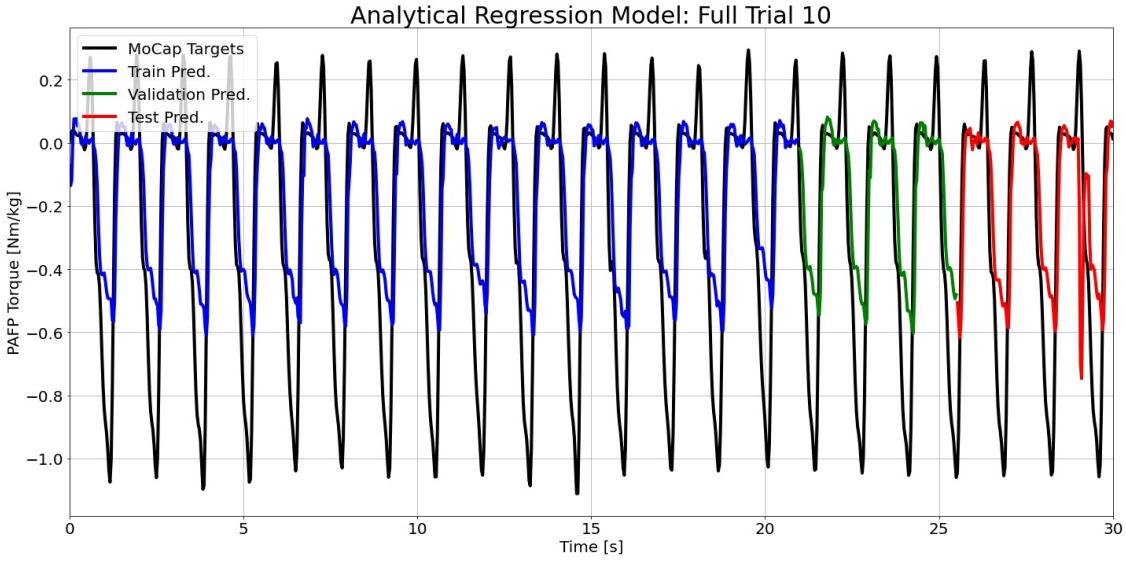


Figure H.162: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 10.

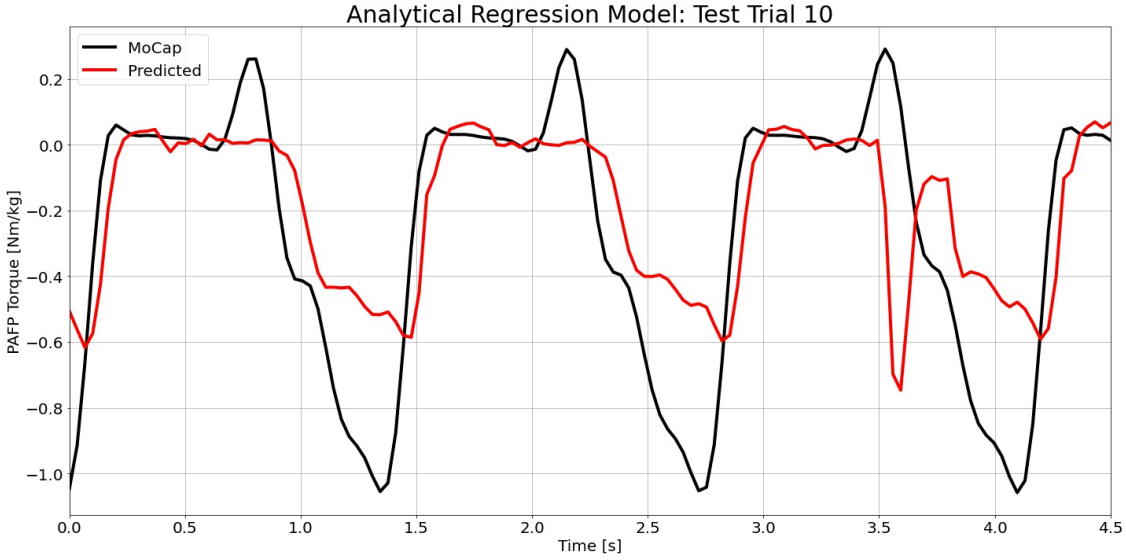


Figure H.163: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 10.

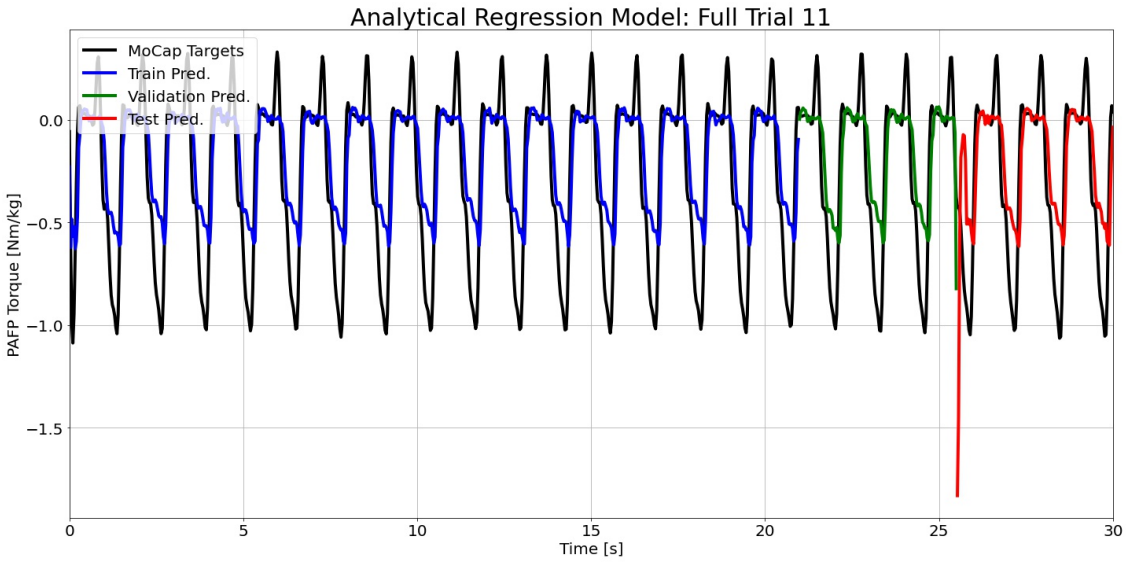


Figure H.164: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 11.

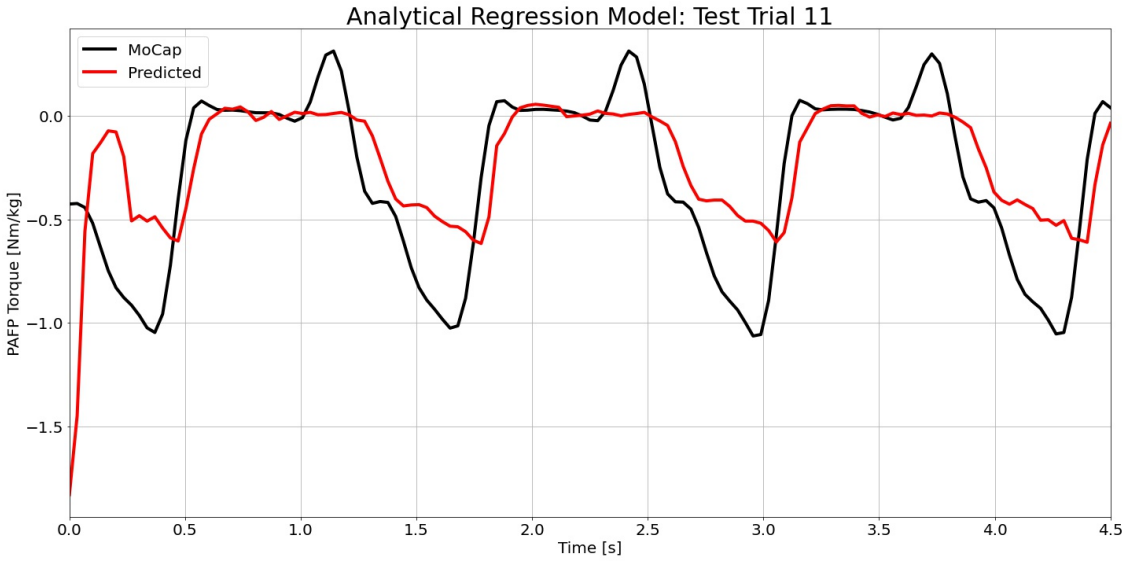


Figure H.165: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 11.

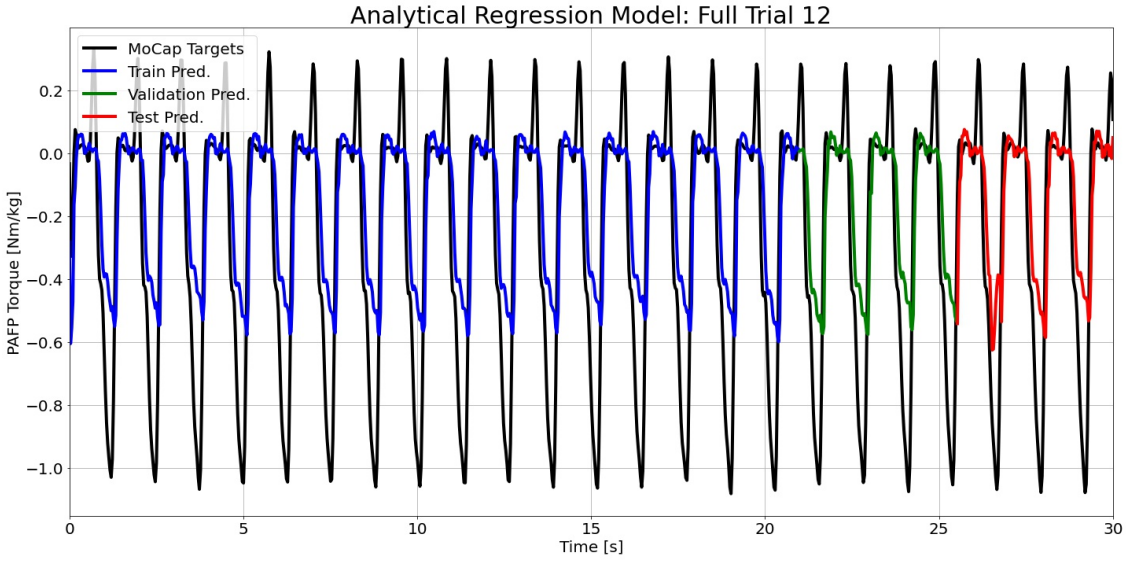


Figure H.166: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 12.

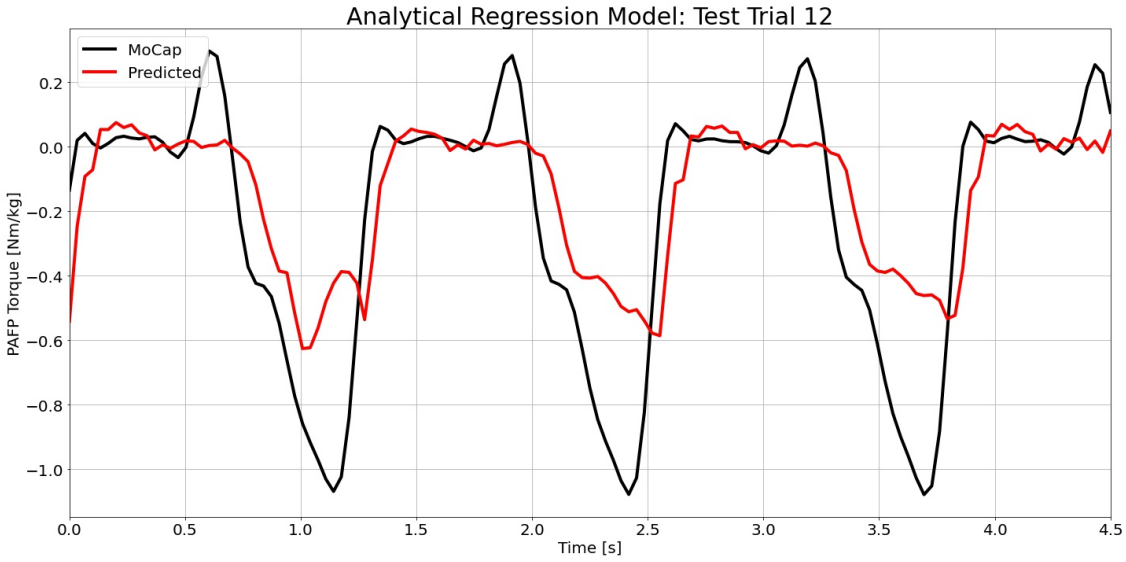


Figure H.167: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 12.

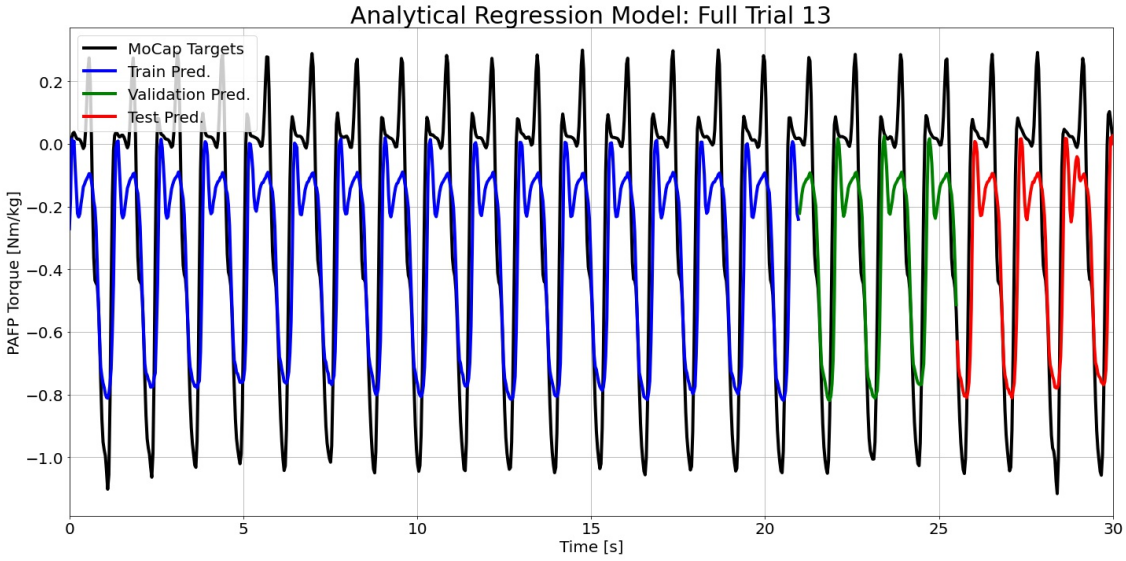


Figure H.168: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 13.

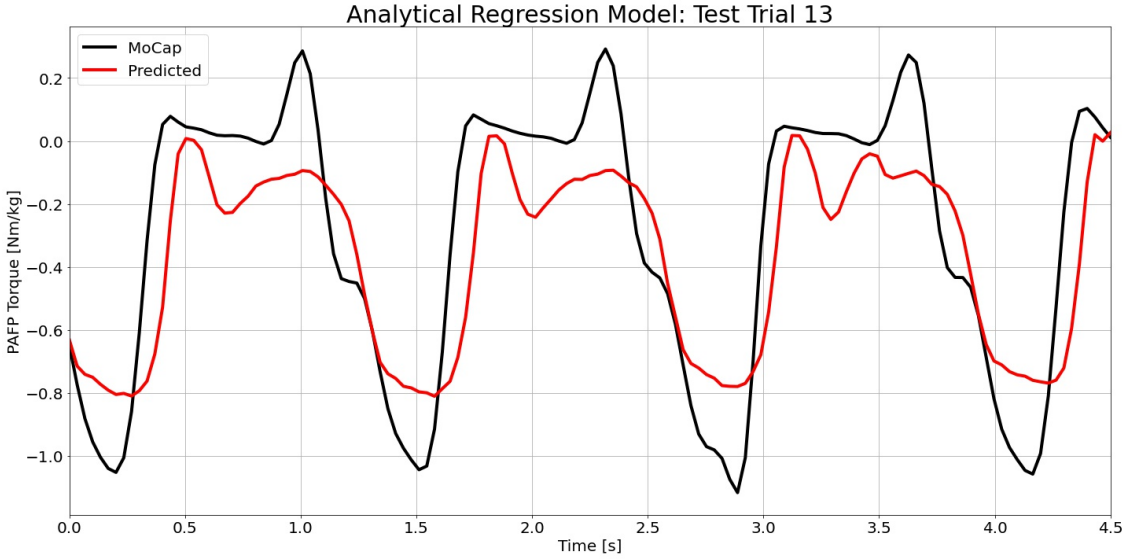


Figure H.169: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 13.

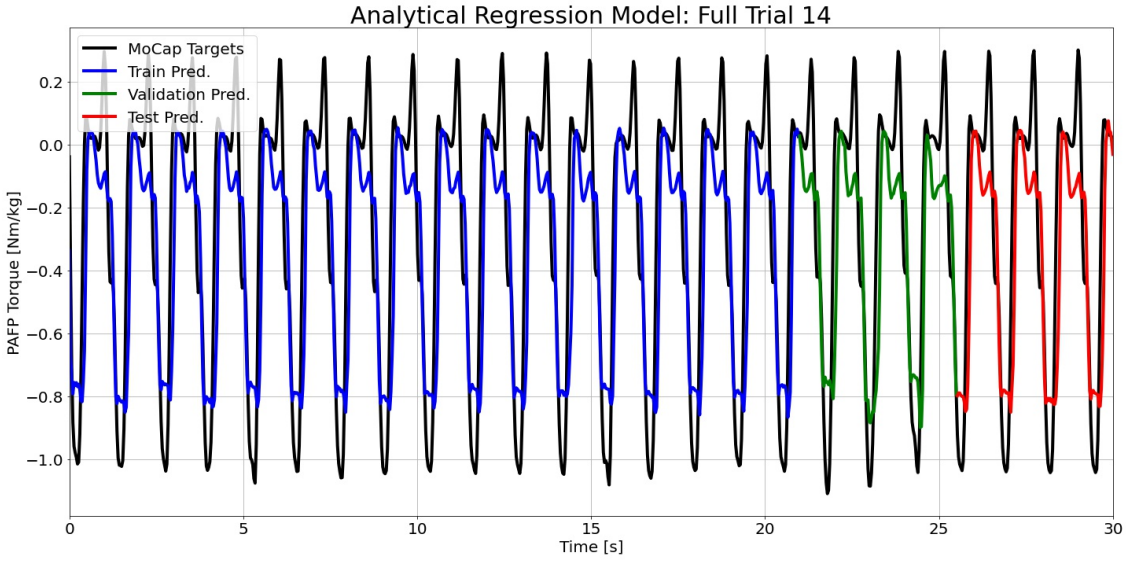


Figure H.170: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 14.

Figure H.171: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 14.

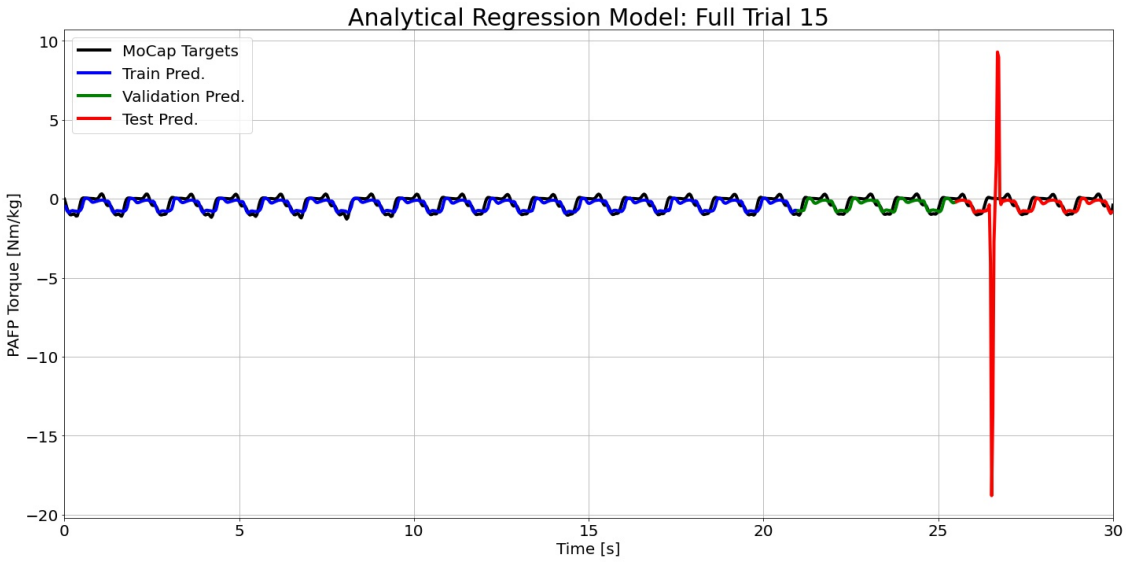


Figure H.172: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 15.

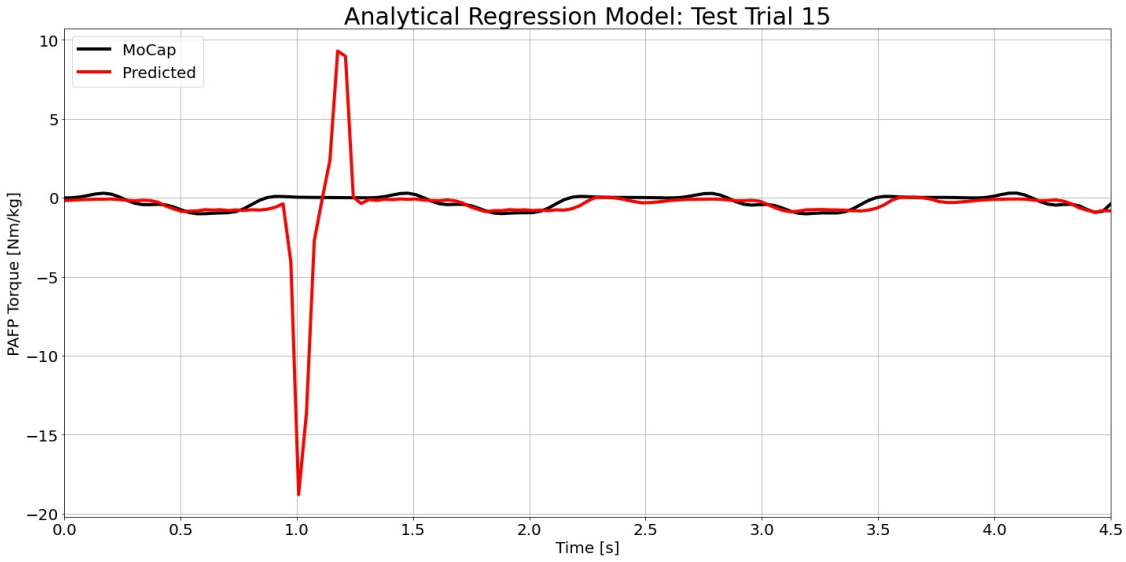


Figure H.173: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 15.

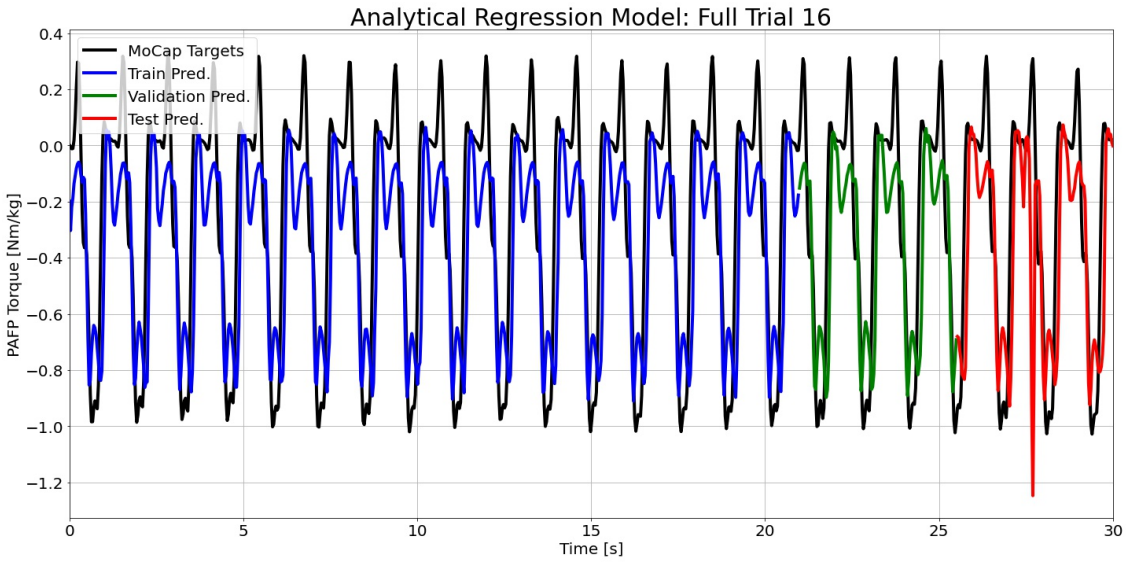


Figure H.174: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 16.

Figure H.175: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 16.

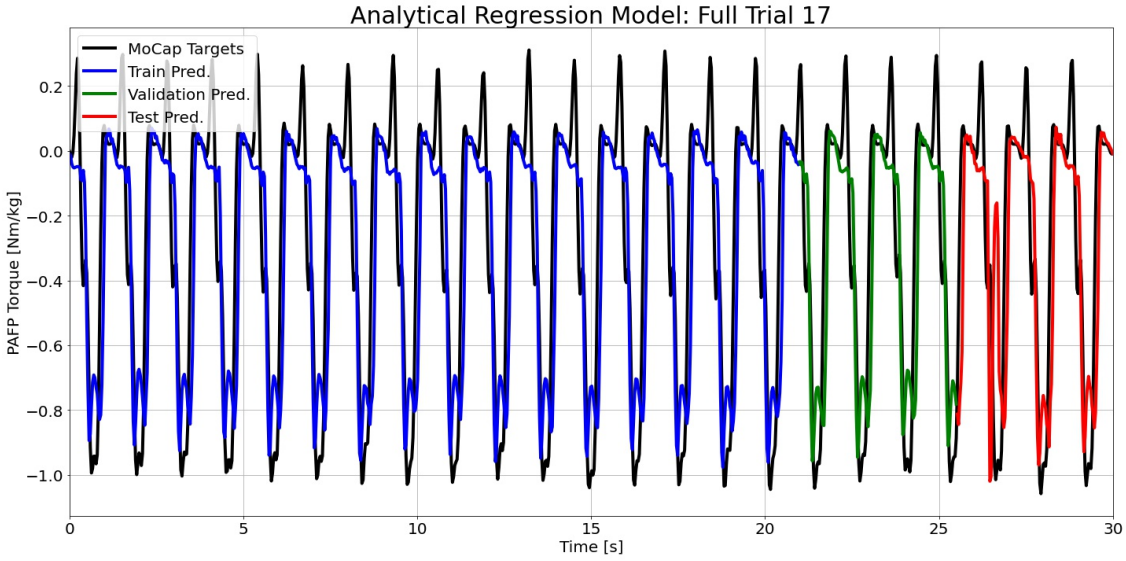


Figure H.176: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 17.

Figure H.177: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 17.

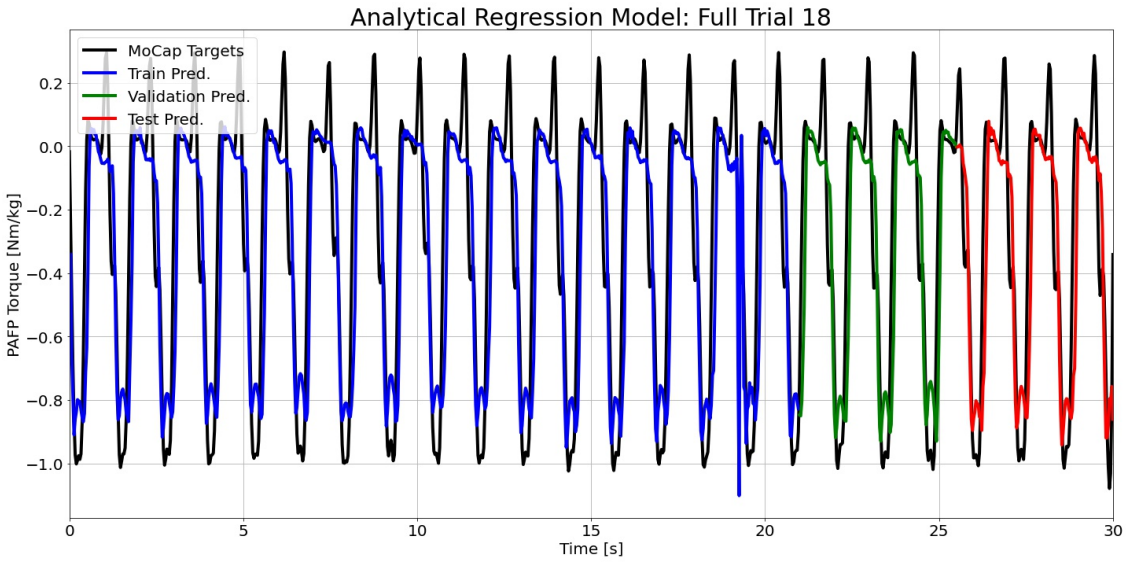


Figure H.178: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 18.

Figure H.179: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 18.

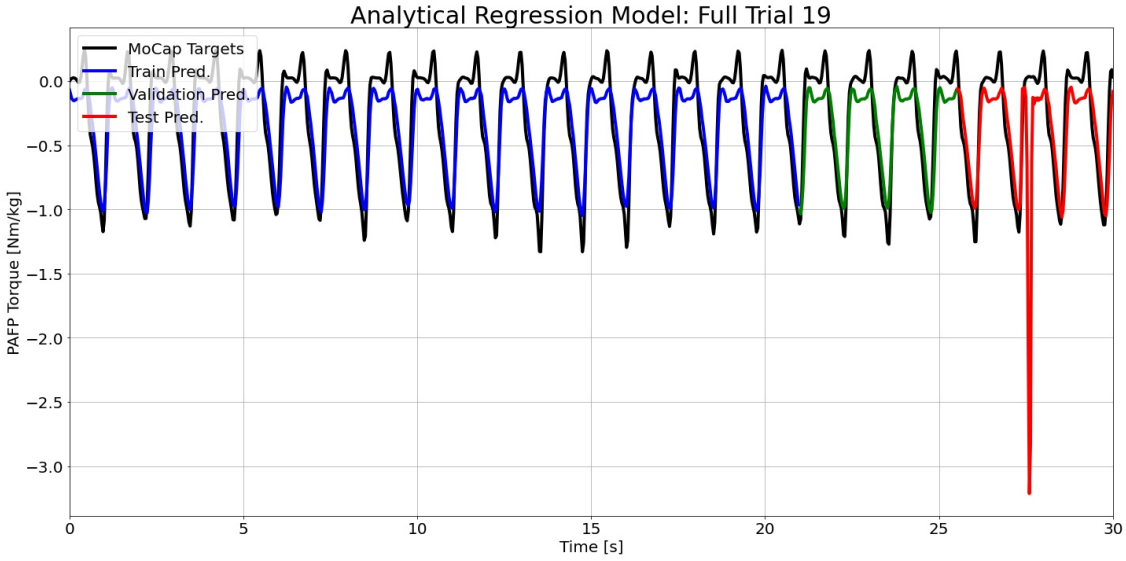


Figure H.180: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 19.

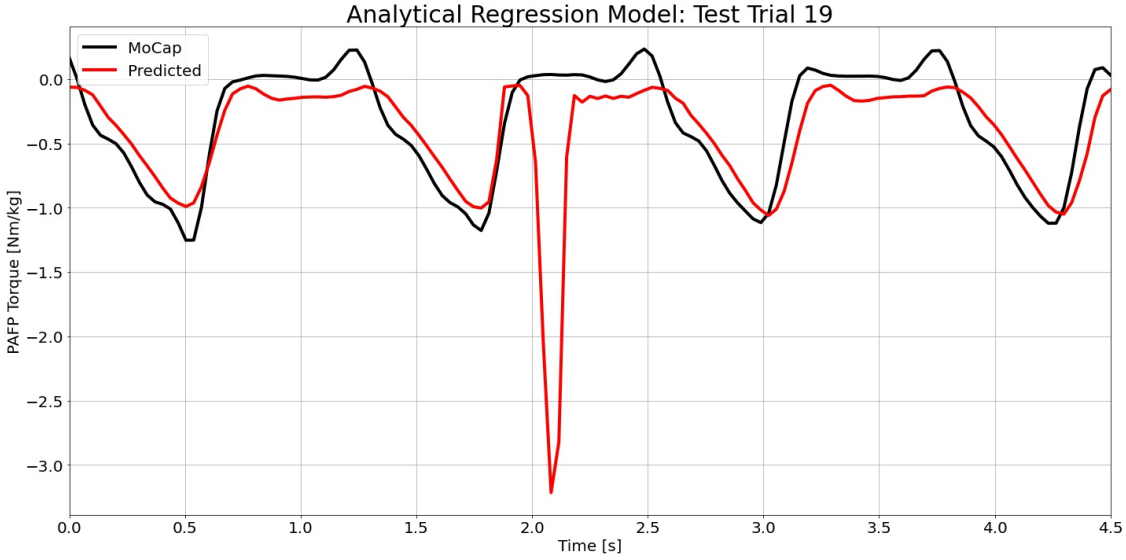


Figure H.181: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 19.

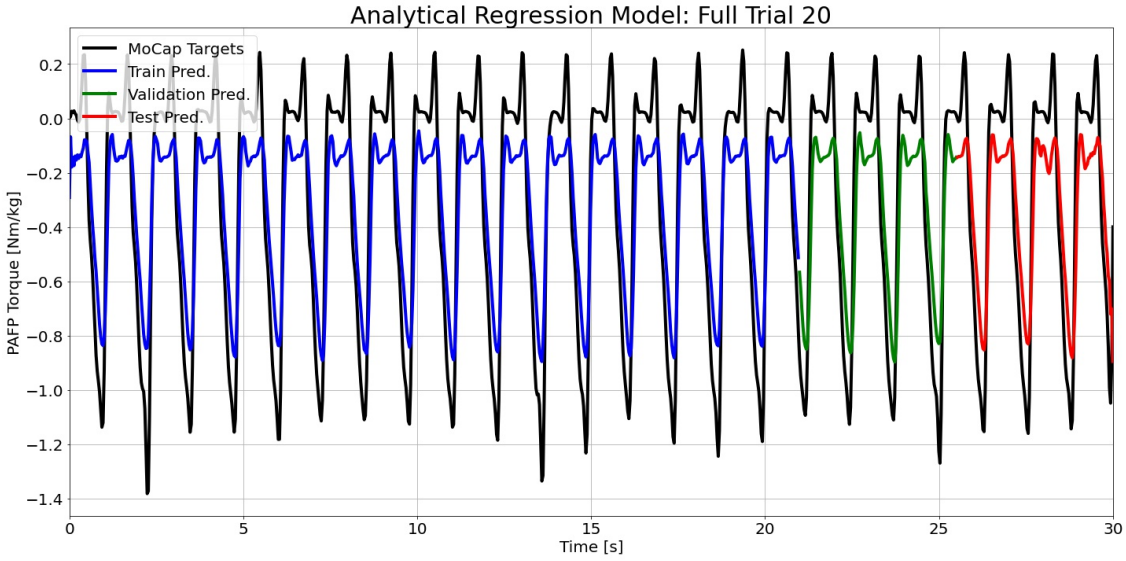


Figure H.182: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 20.

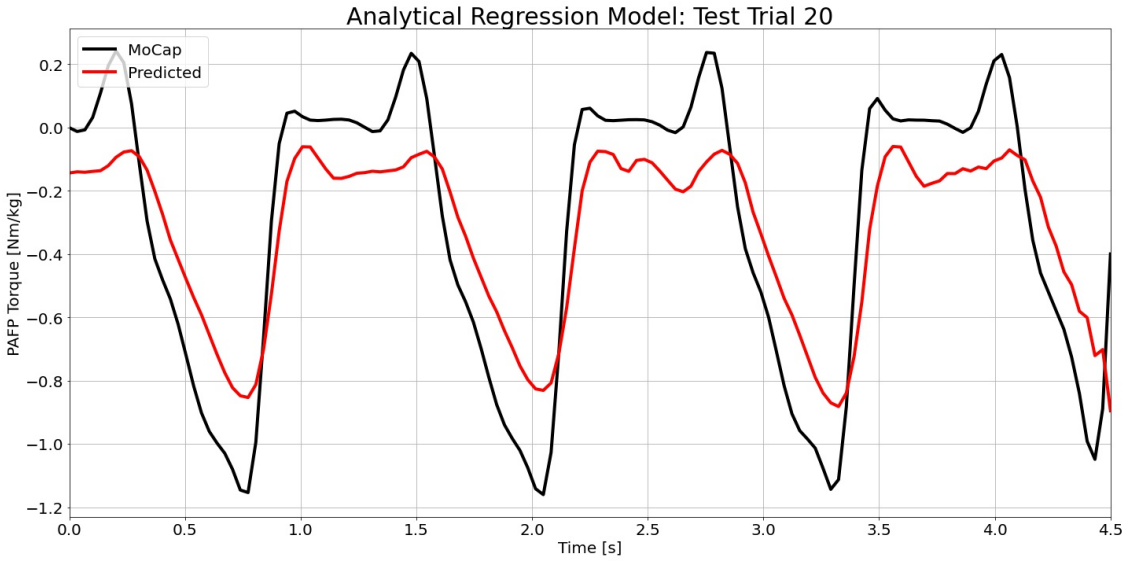


Figure H.183: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 20.

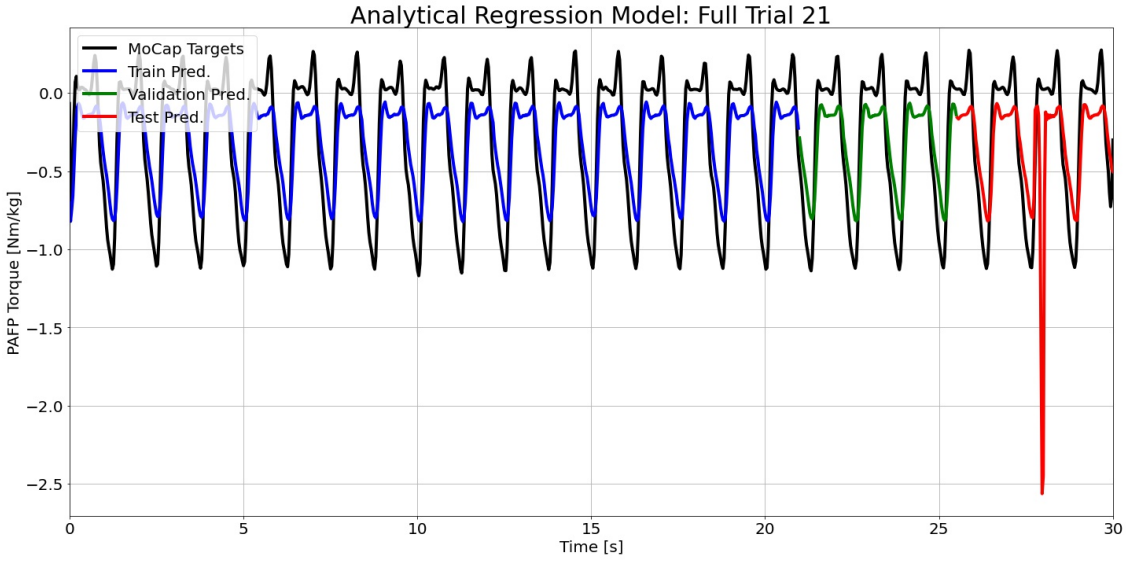


Figure H.184: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 21.

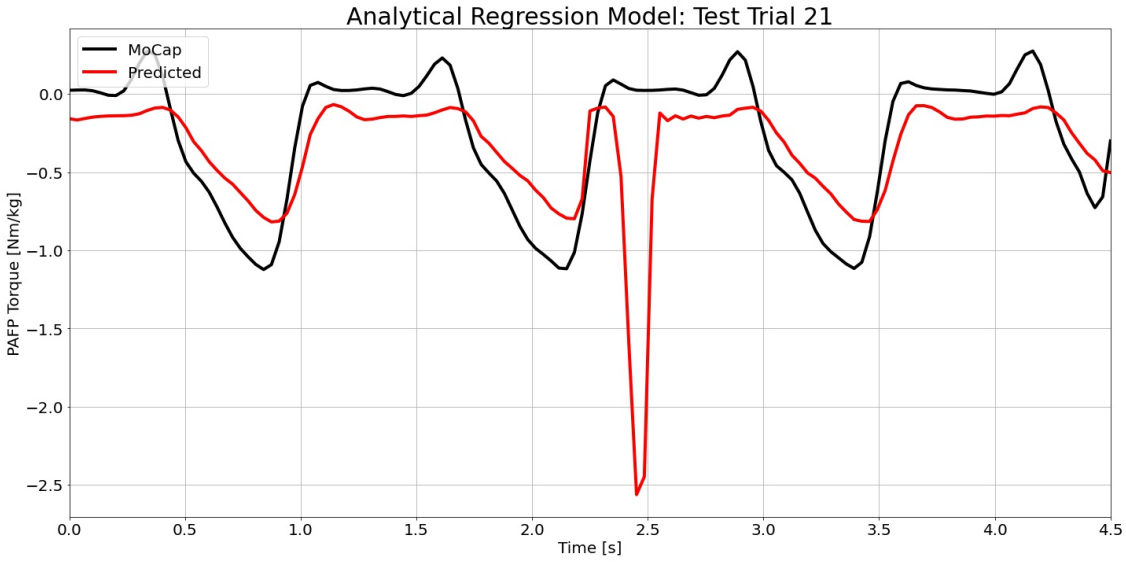


Figure H.185: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 21.

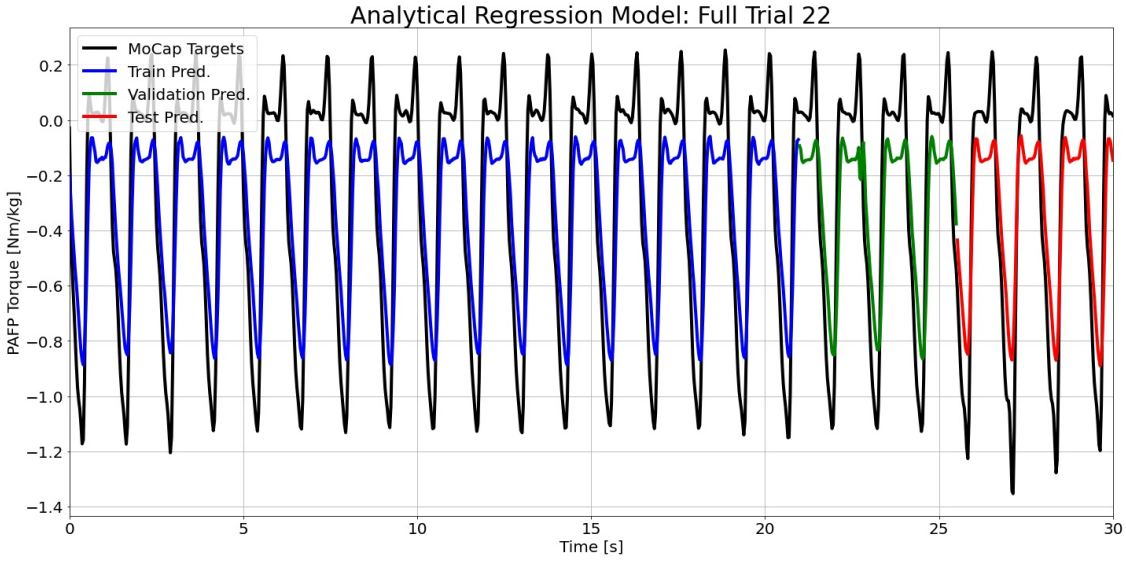


Figure H.186: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 22.

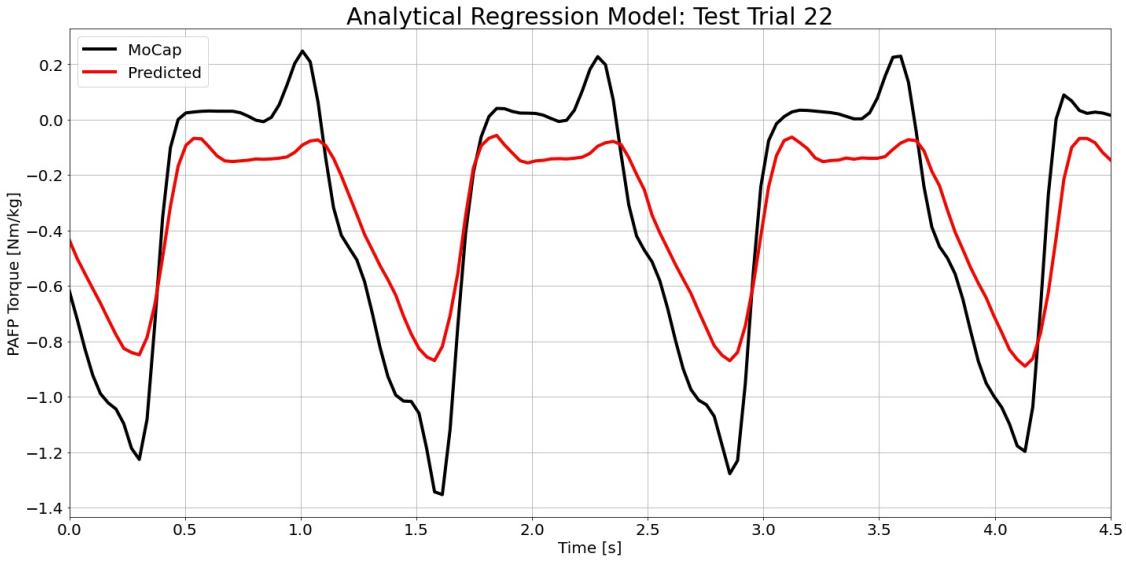


Figure H.187: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 22.

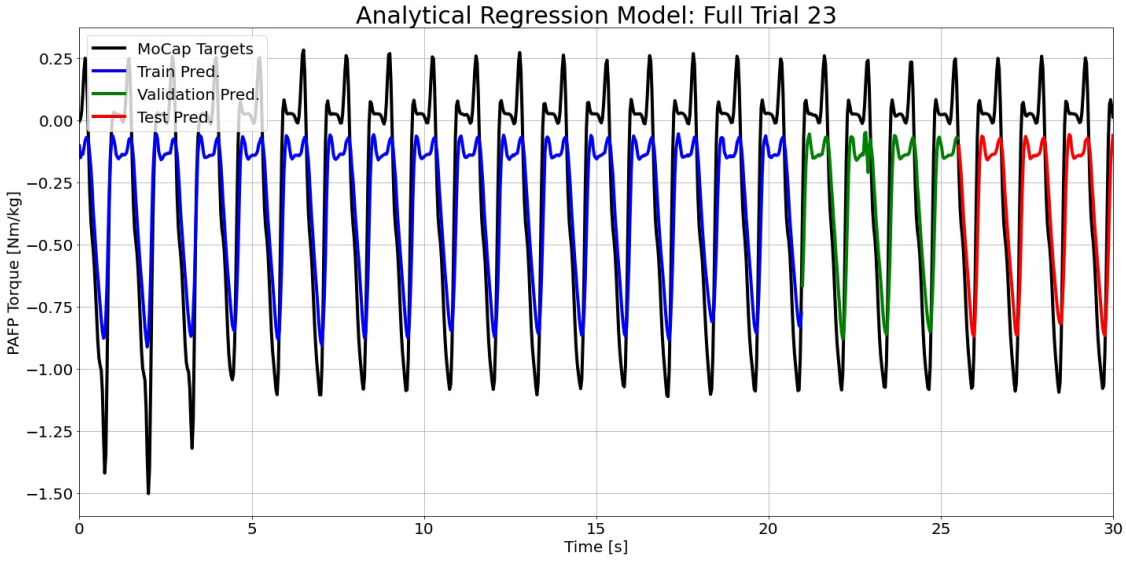


Figure H.188: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for full time series 23.

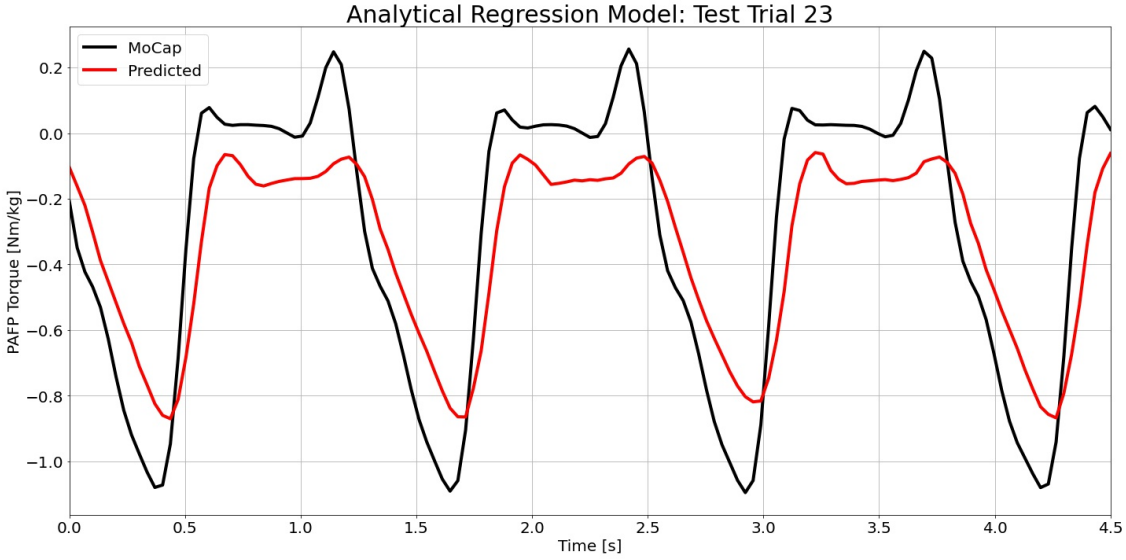


Figure H.189: Analytical model predictions compared to the MoCap inverse dynamics ground truth values for test time series 23.

Appendix I

TWENTY-SAMPLE AHEAD TIME SERIES PREDICTION RESULTS

This appendix contains the full time series and test results for twenty-sample ahead model predictions. The four models and training procedures are discussed in Chapter 6.

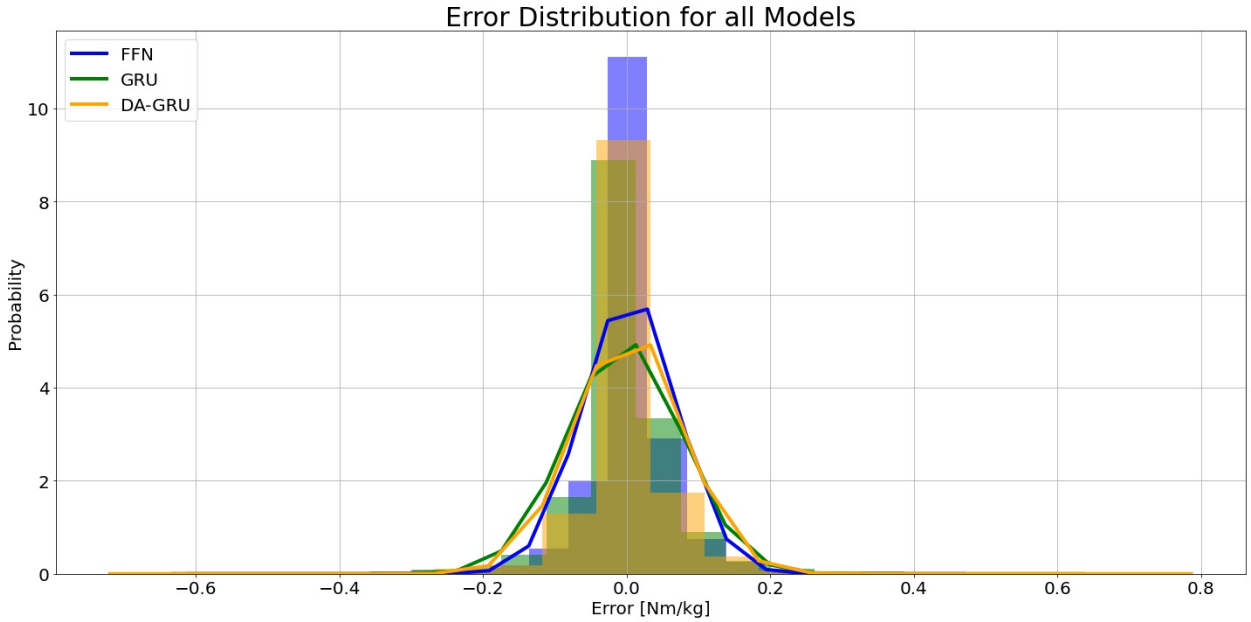


Figure I.1: Prediction error histogram for all DNN models across all test time series data.

I.1 Feedforward Neural Network

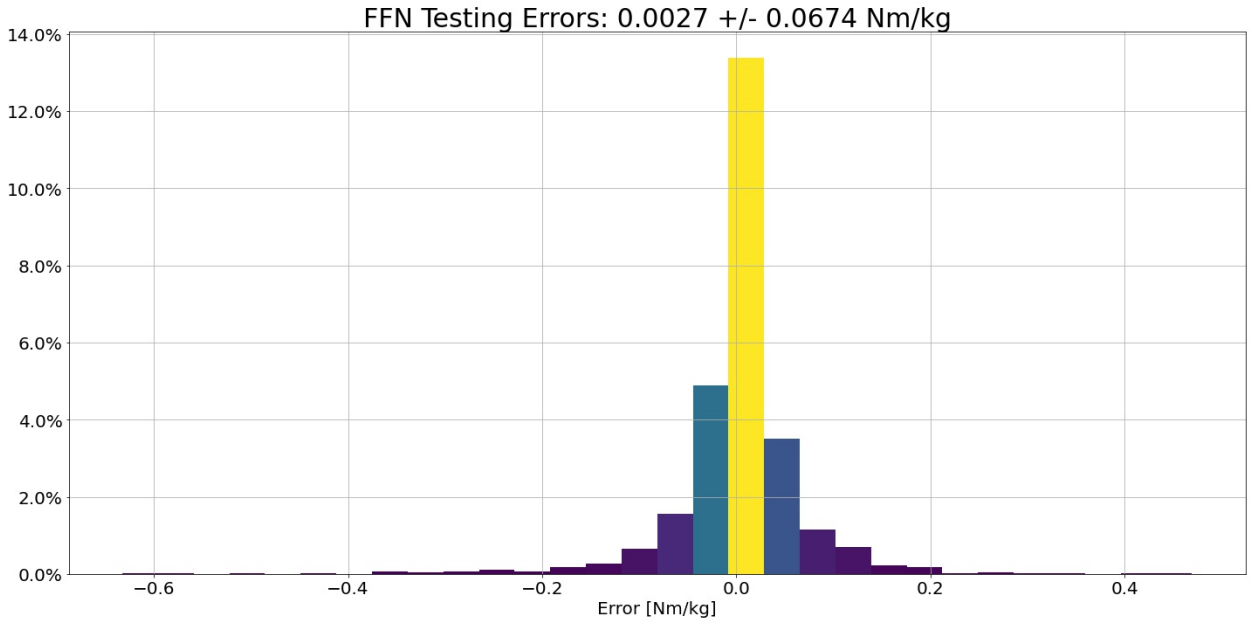


Figure I.2: FFN prediction error histogram for all test time series data.

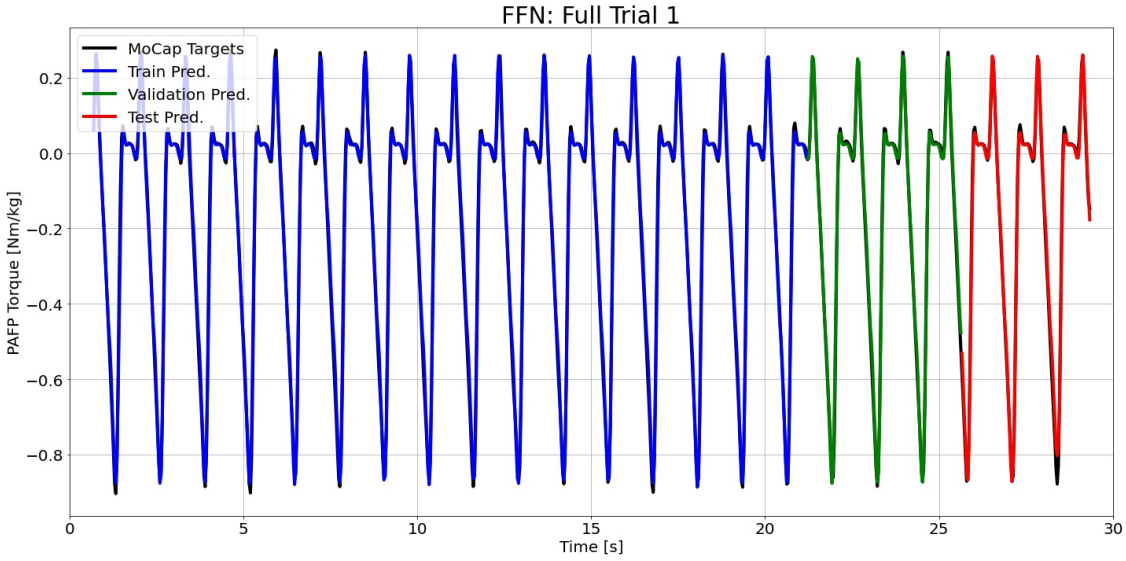


Figure I.3: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 1.

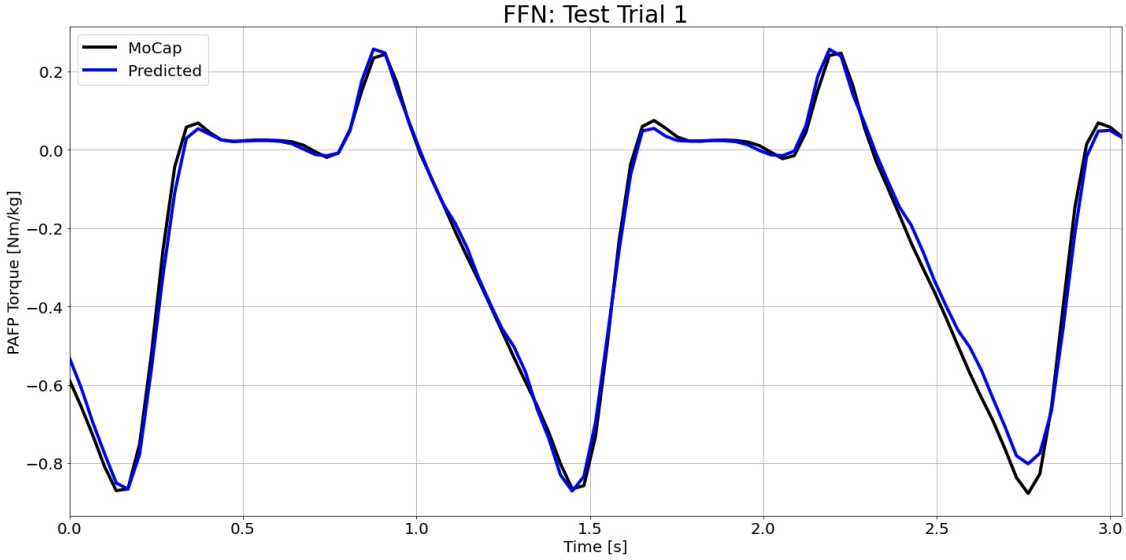


Figure I.4: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 1.

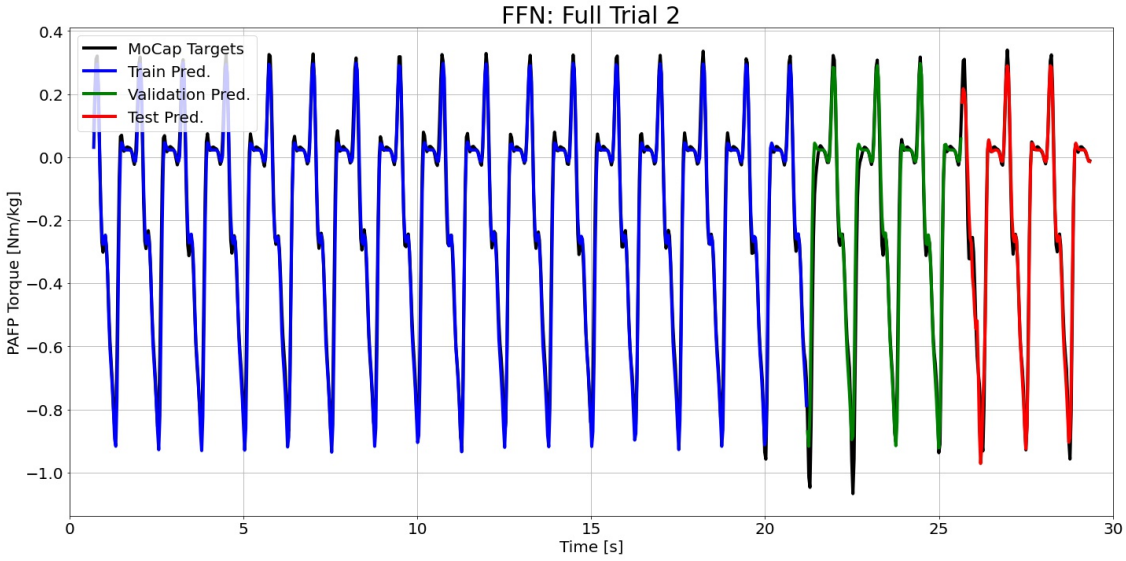


Figure I.5: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 2.

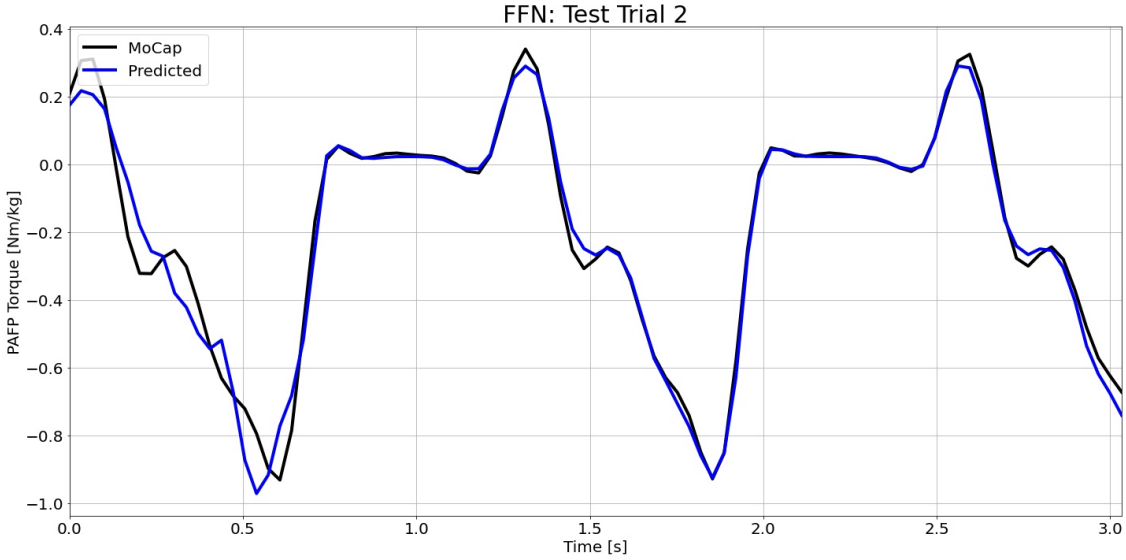


Figure I.6: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 2.

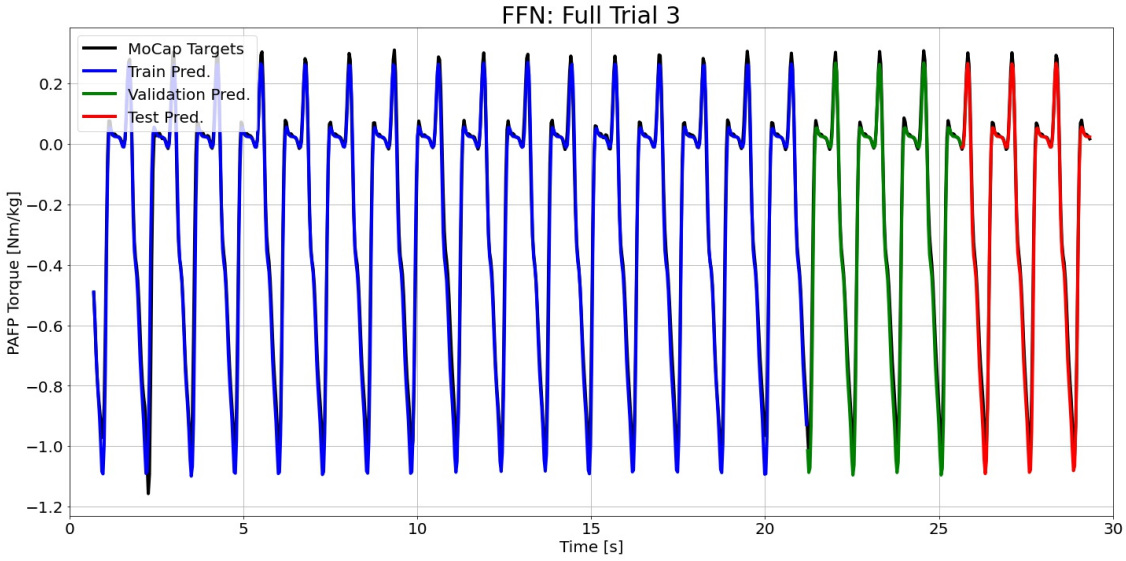


Figure I.7: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 3.

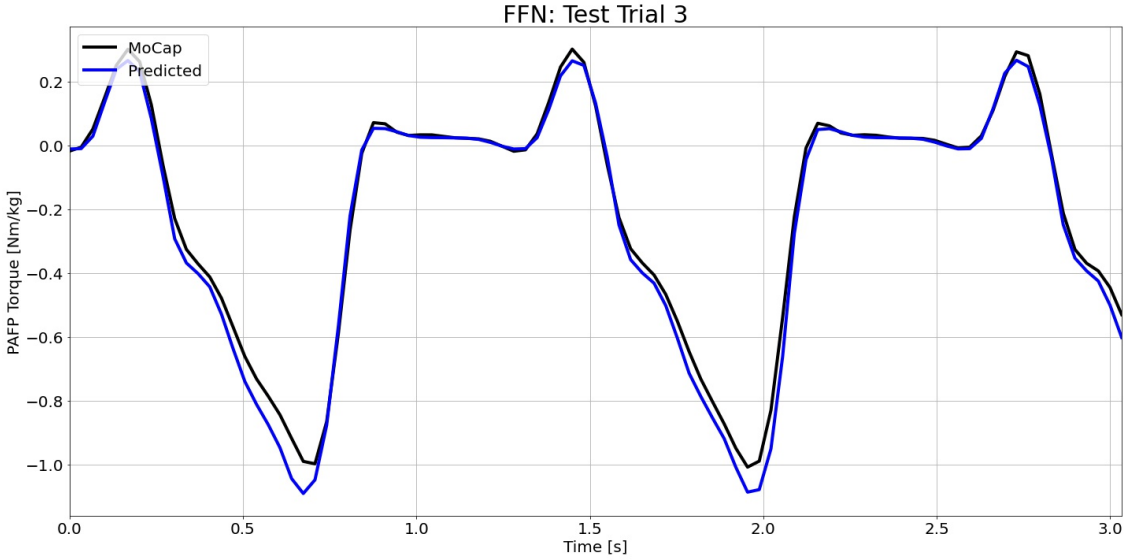


Figure I.8: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 3.

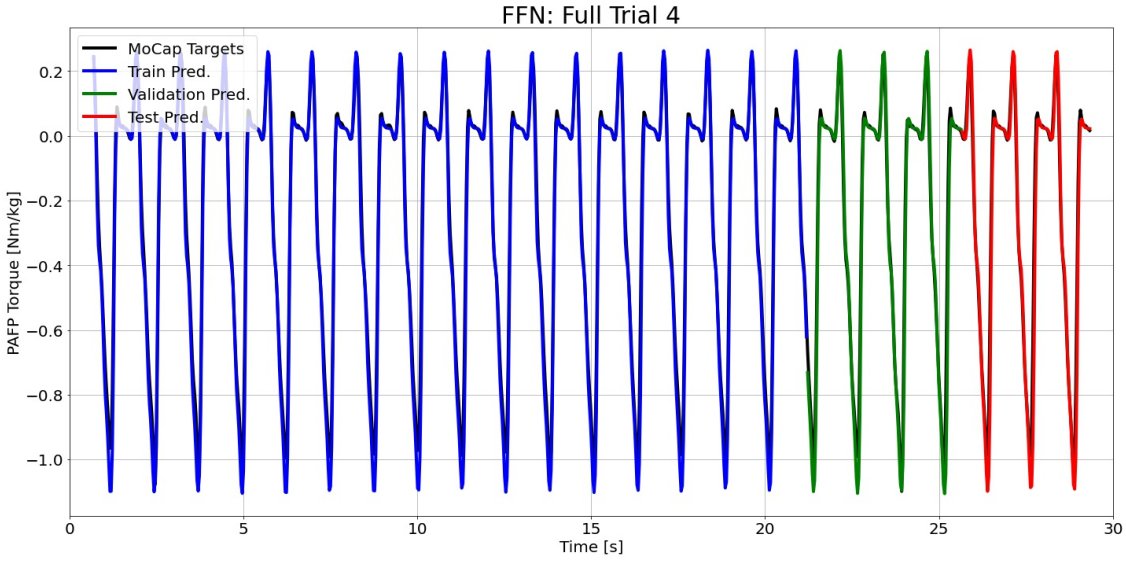


Figure I.9: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 4.

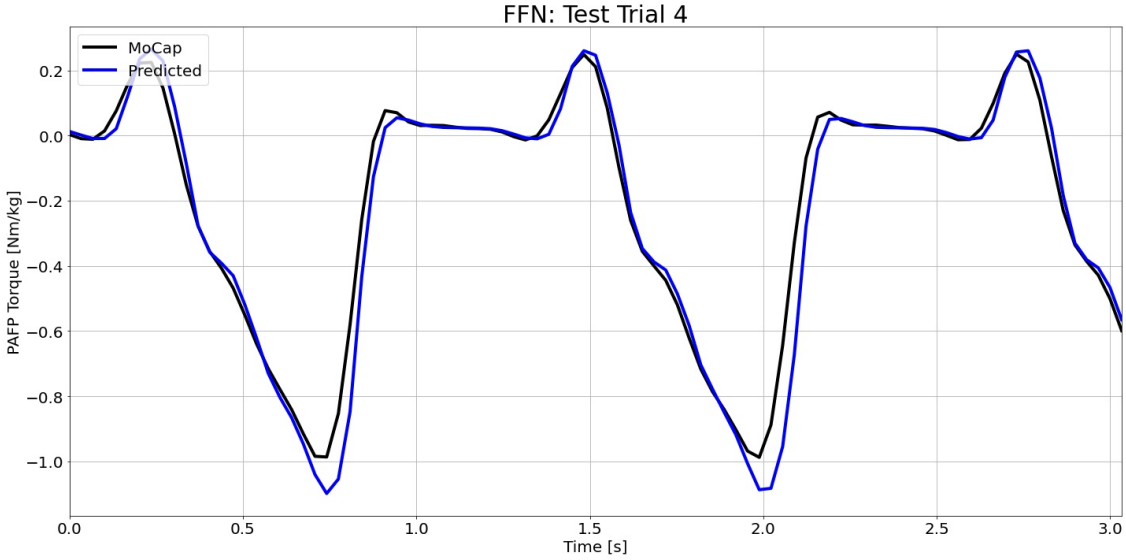


Figure I.10: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 4.

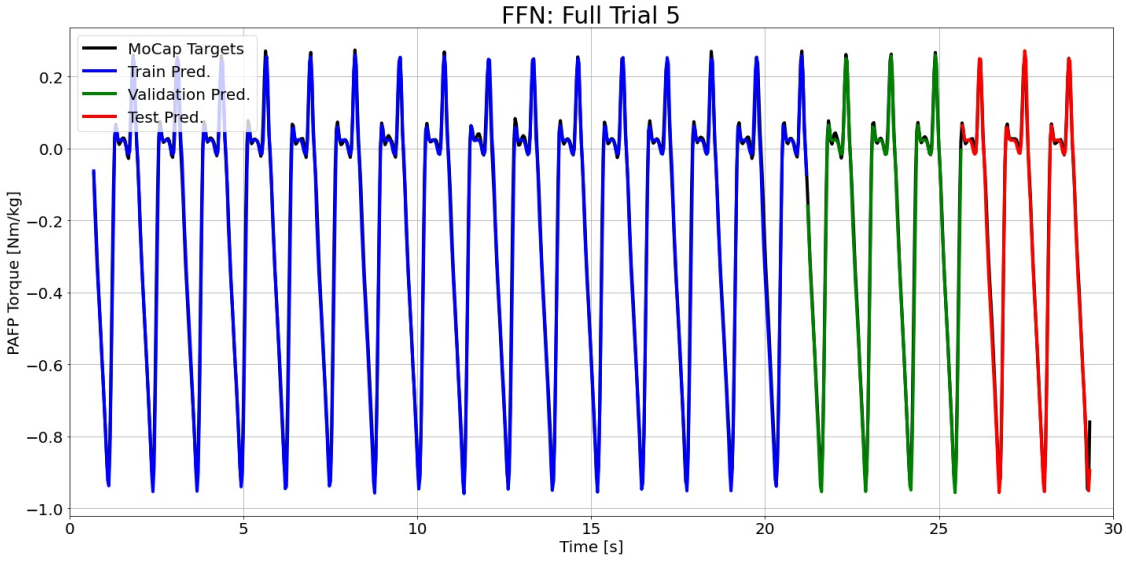


Figure I.11: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 5.

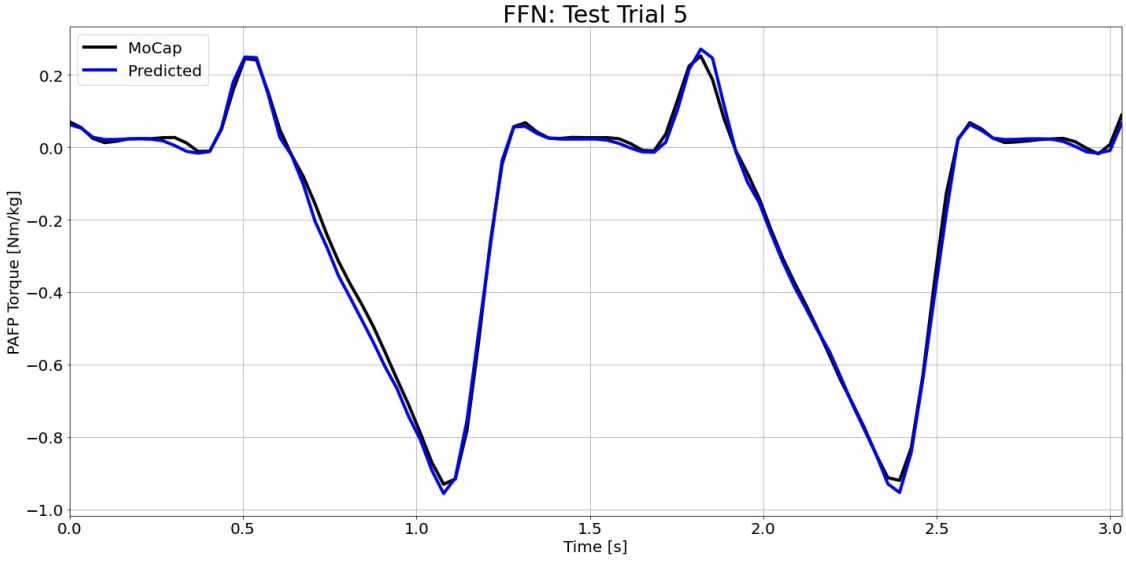


Figure I.12: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 5.

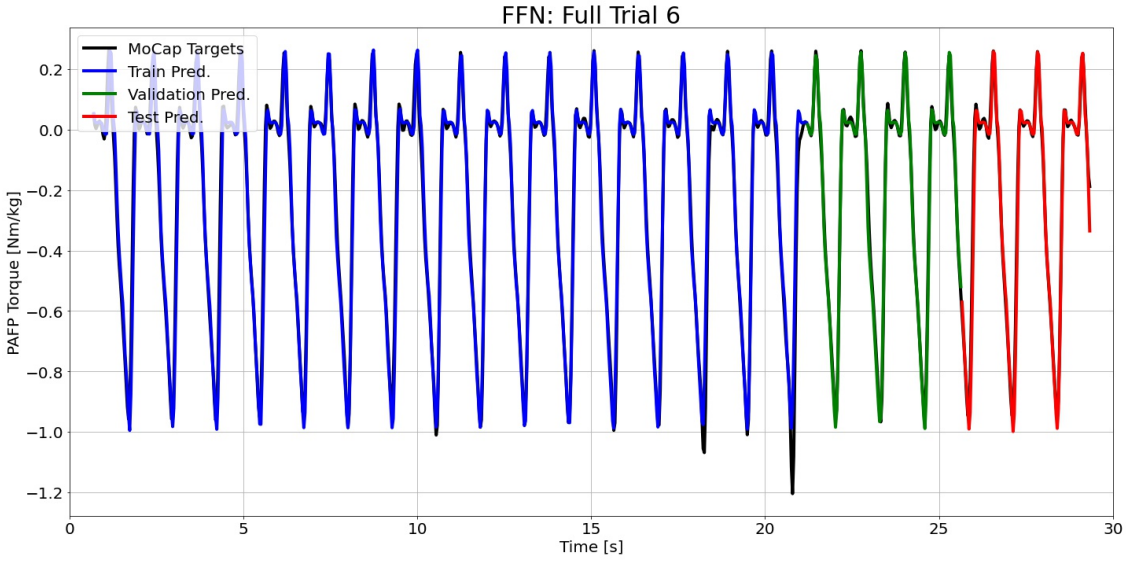


Figure I.13: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 6.

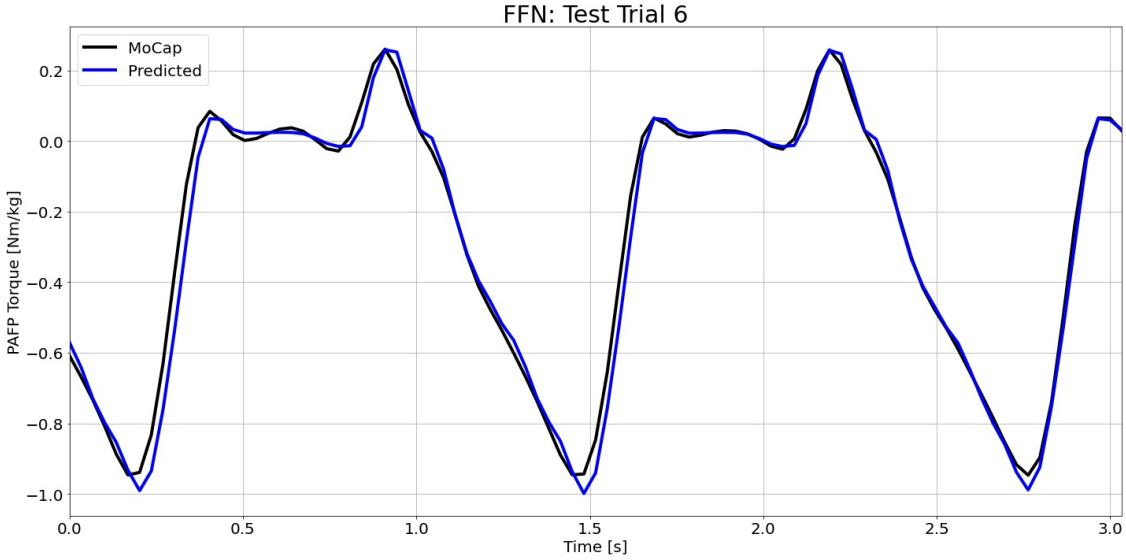


Figure I.14: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 6.

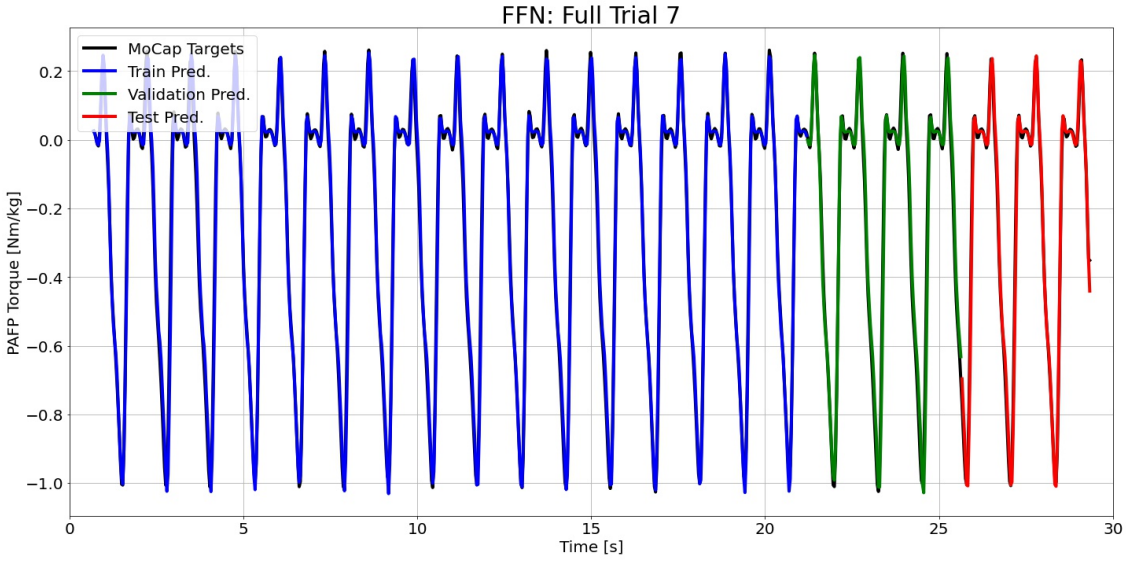


Figure I.15: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 7.

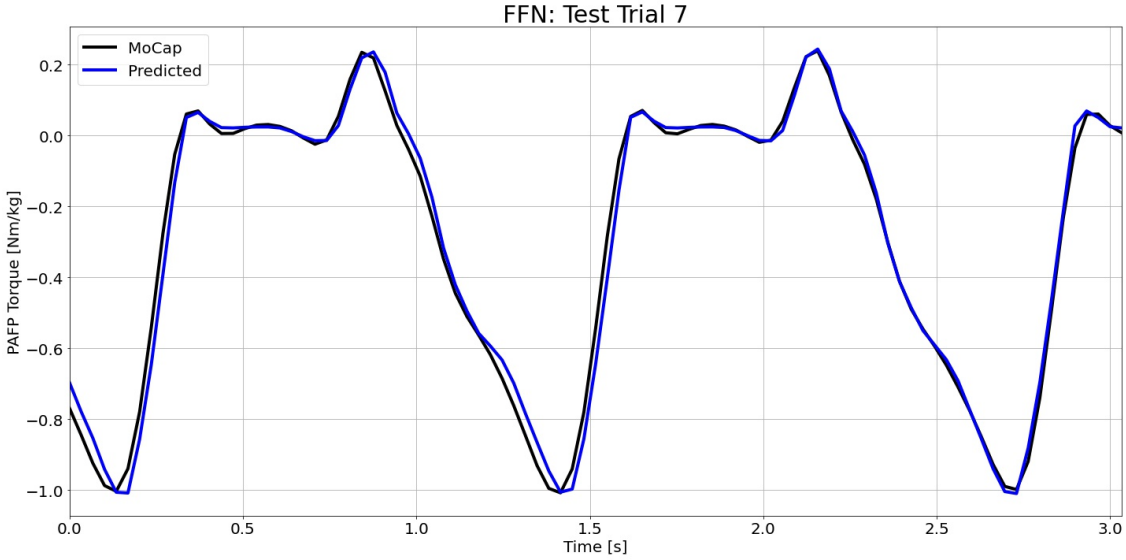


Figure I.16: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 7.

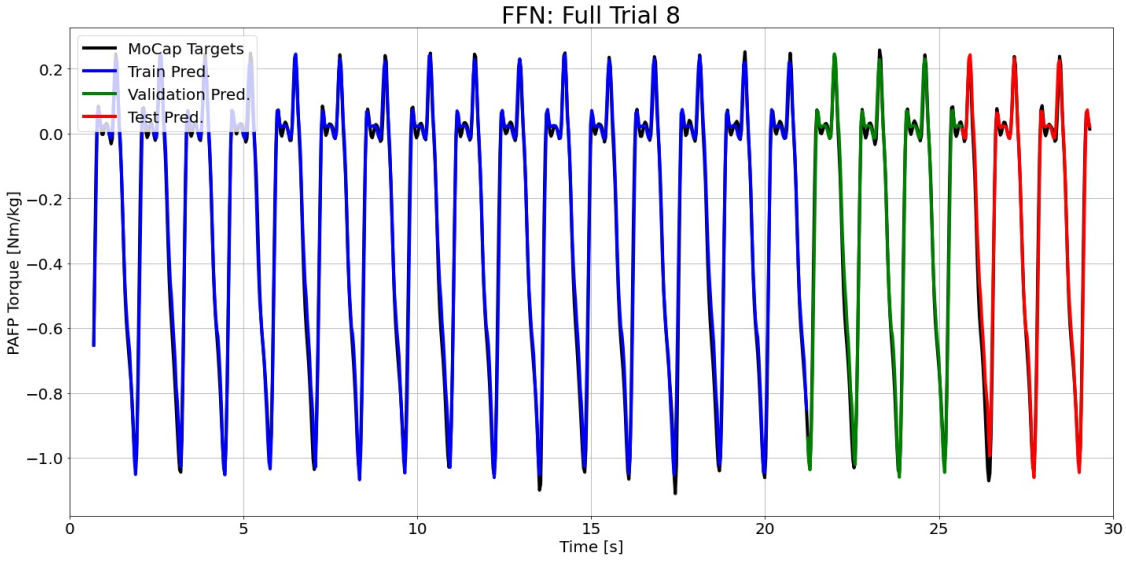


Figure I.17: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 8.

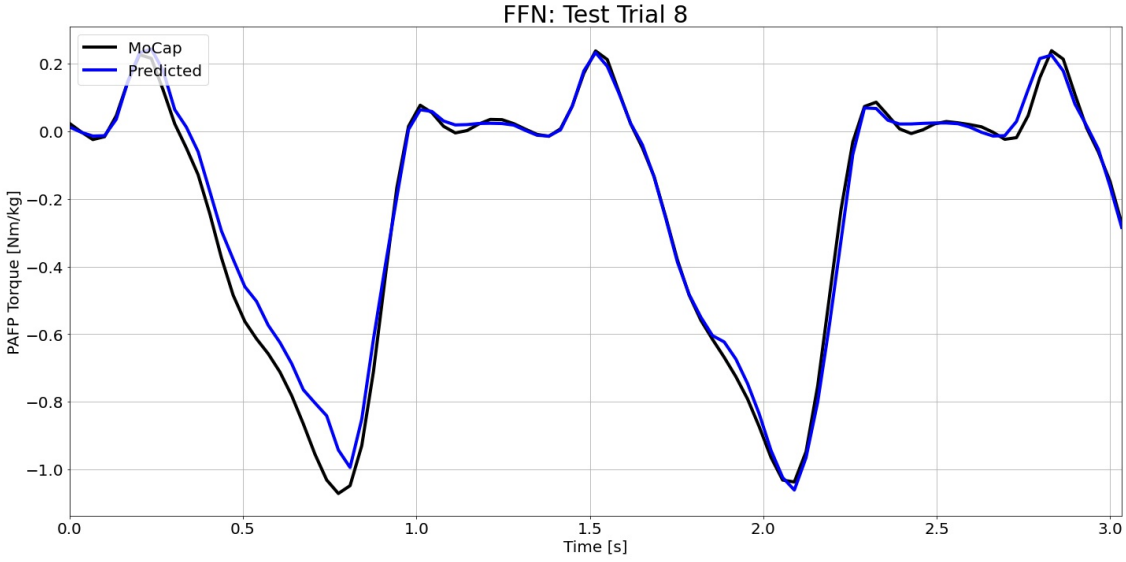


Figure I.18: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 8.

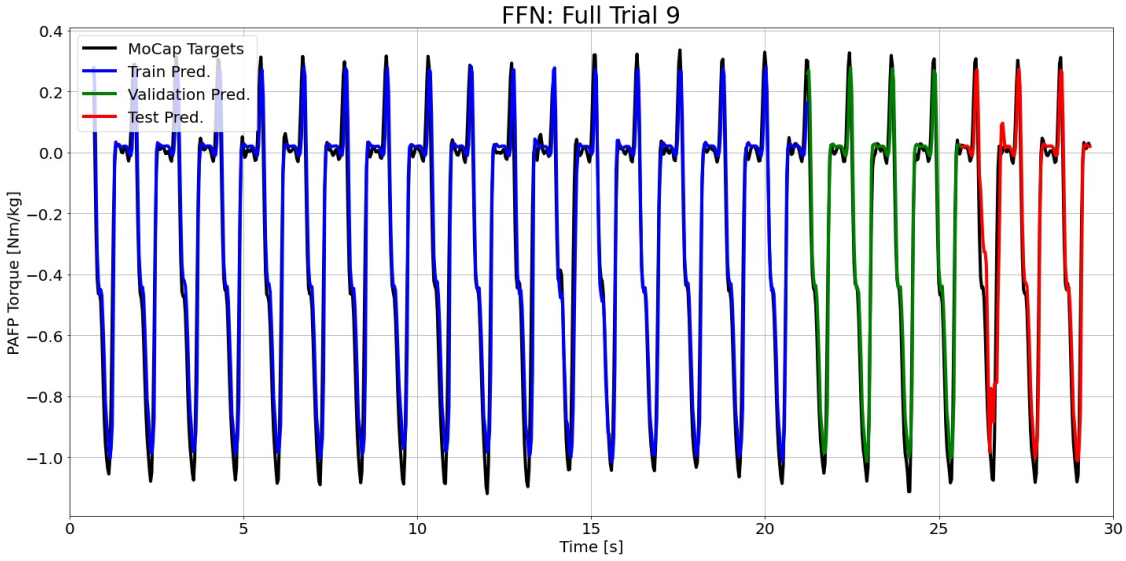


Figure I.19: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 9.

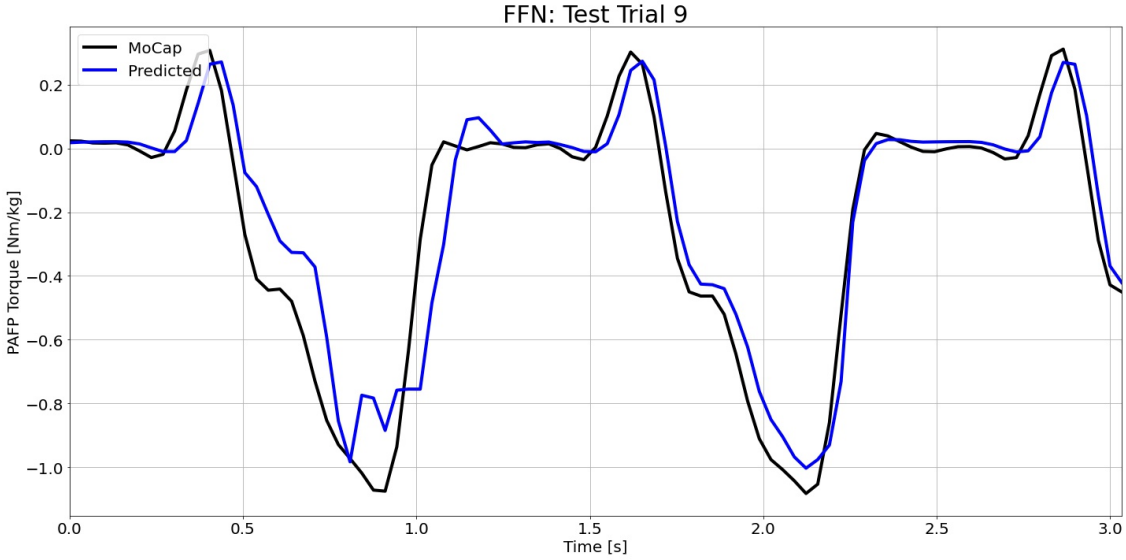


Figure I.20: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 9.

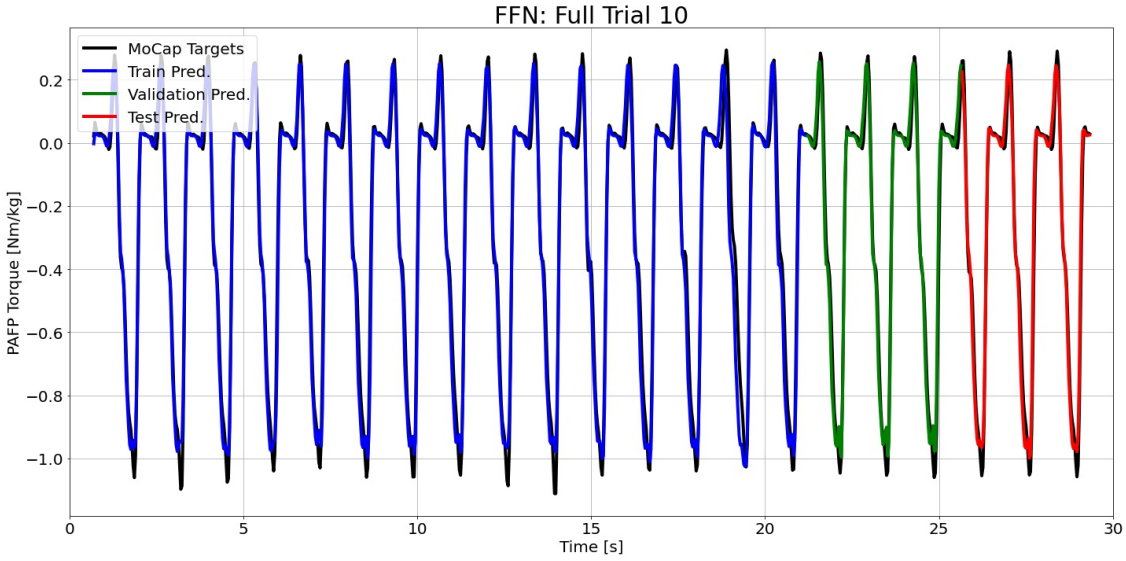


Figure I.21: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 10.

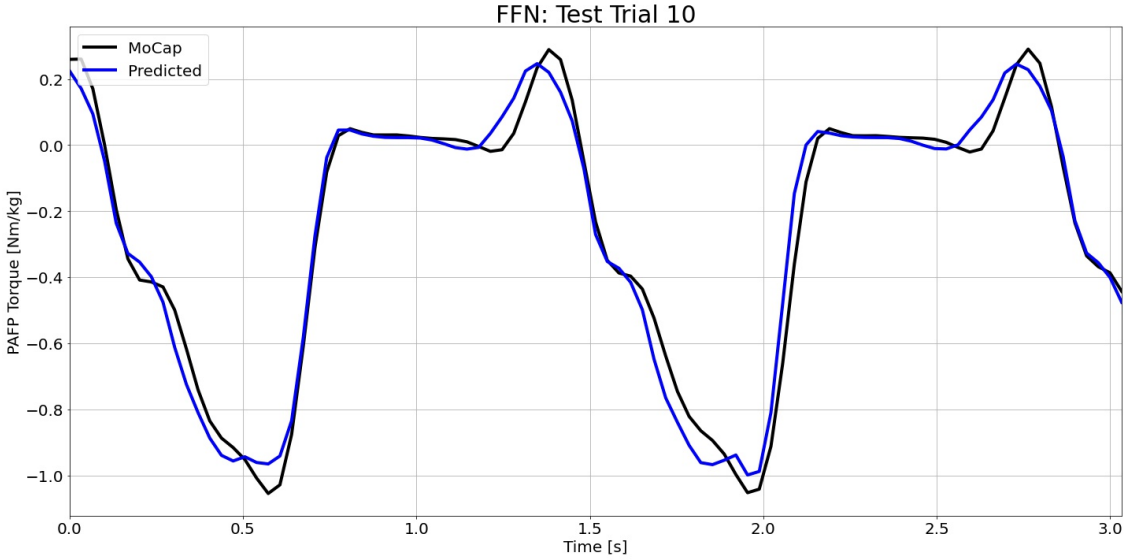


Figure I.22: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 10.

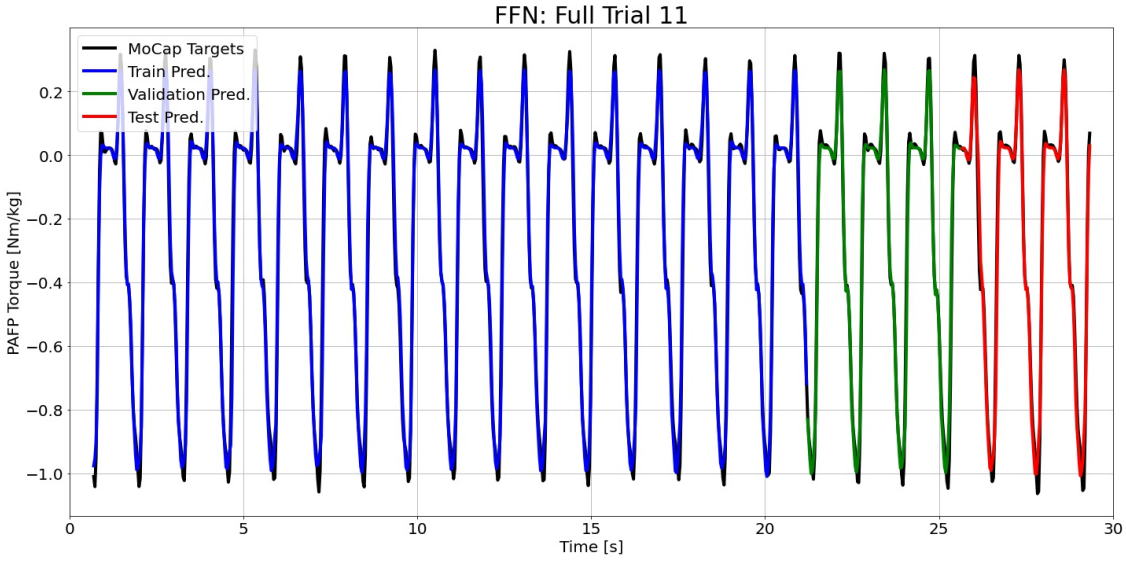


Figure I.23: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 11.

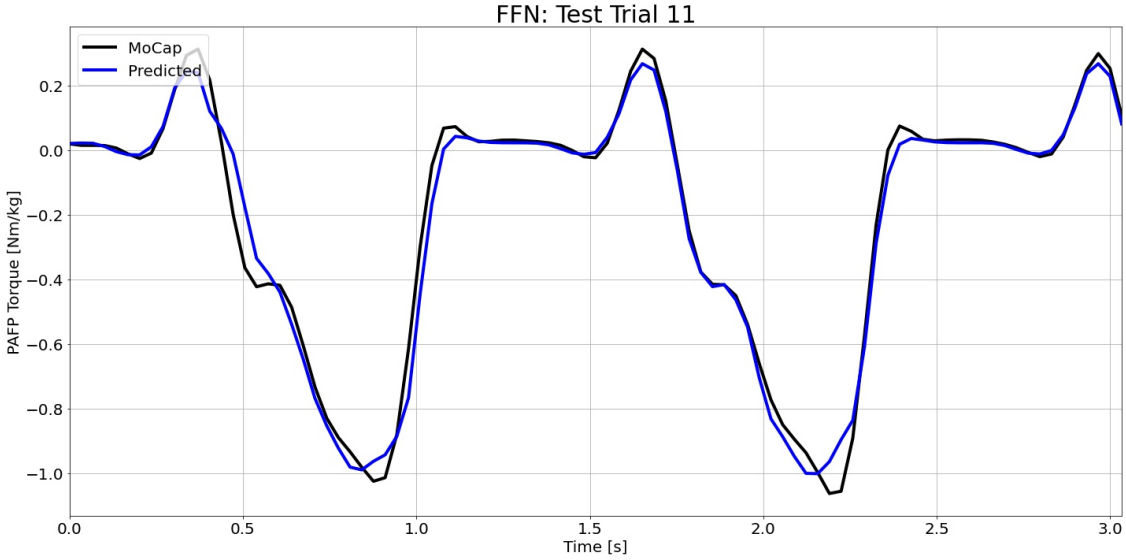


Figure I.24: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 11.

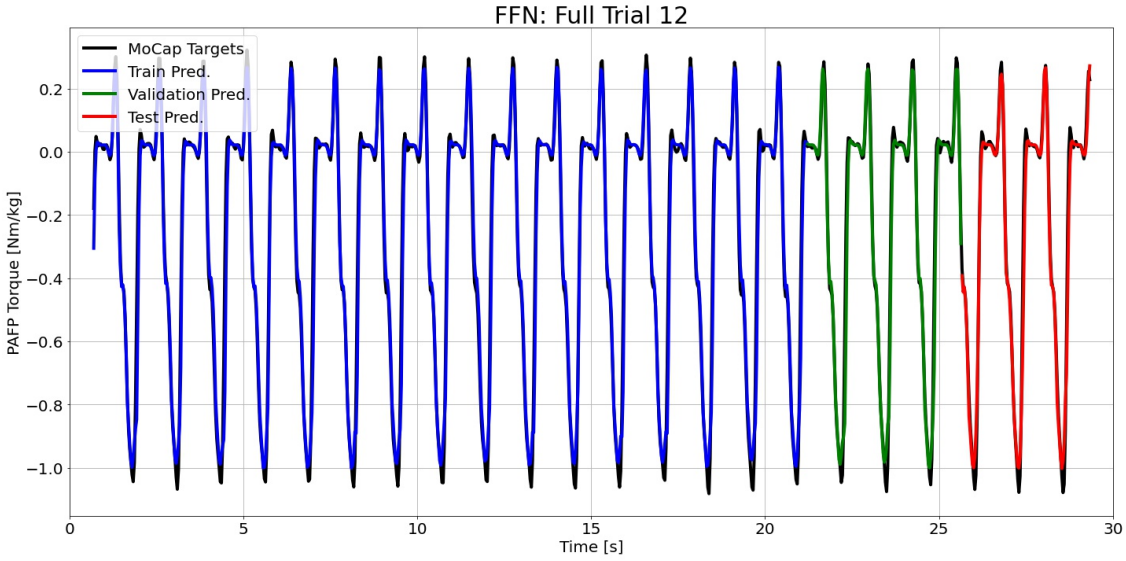


Figure I.25: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 12.

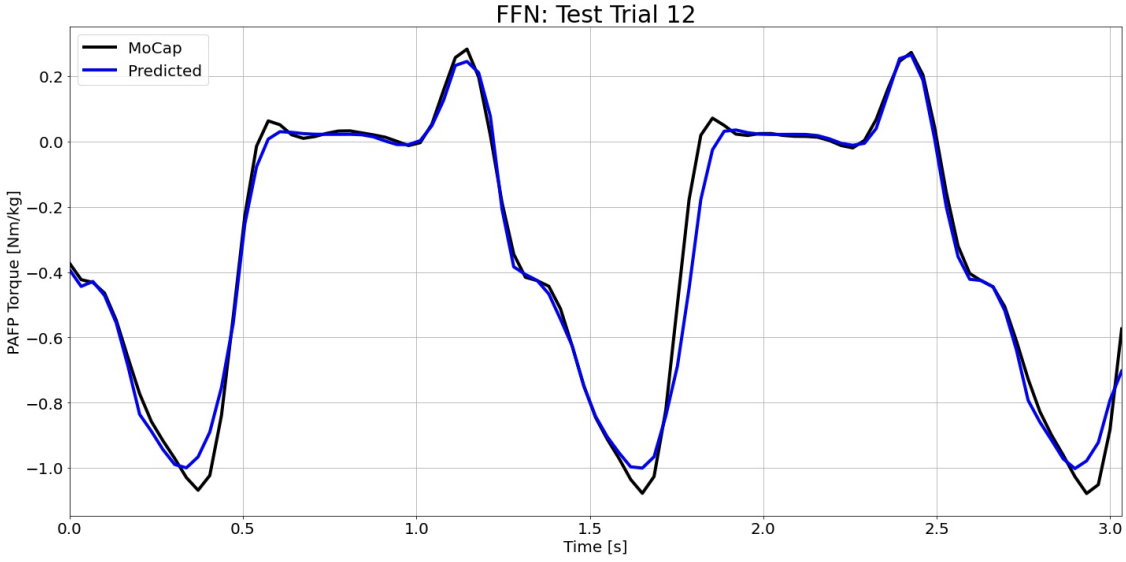


Figure I.26: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 12.

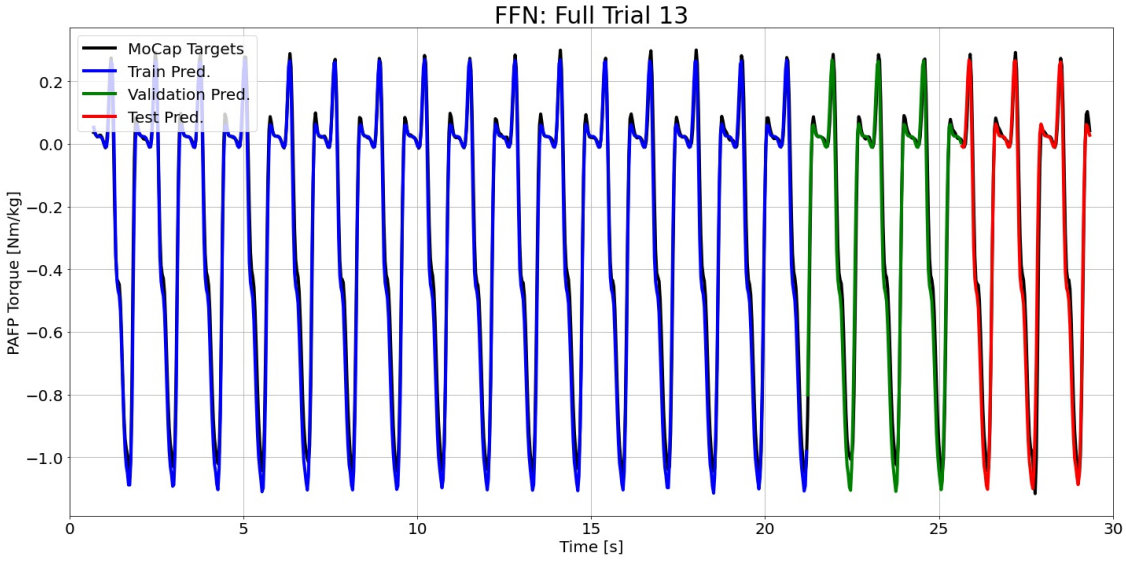


Figure I.27: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 13.

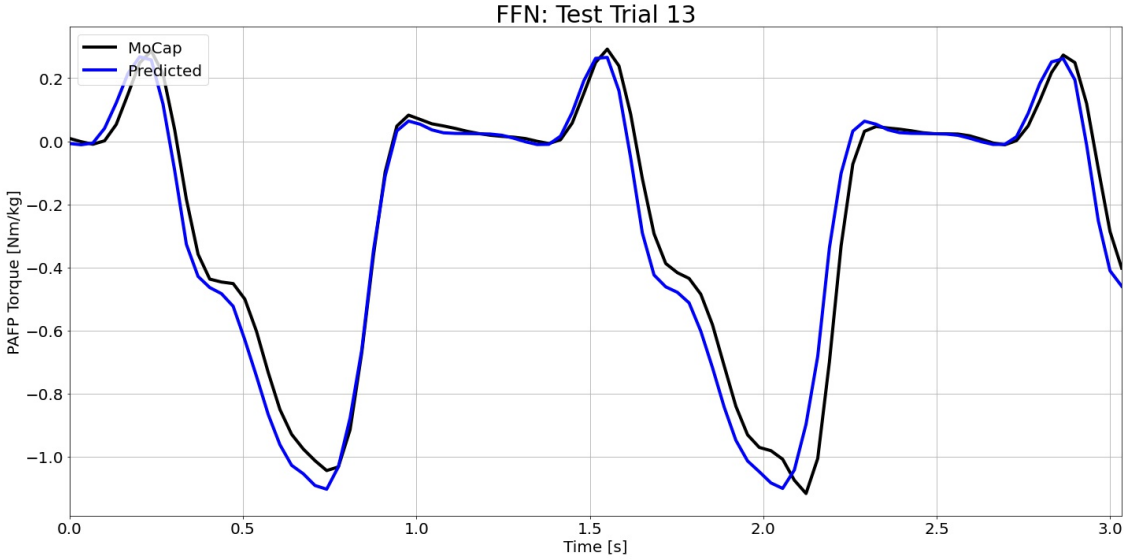


Figure I.28: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 13.

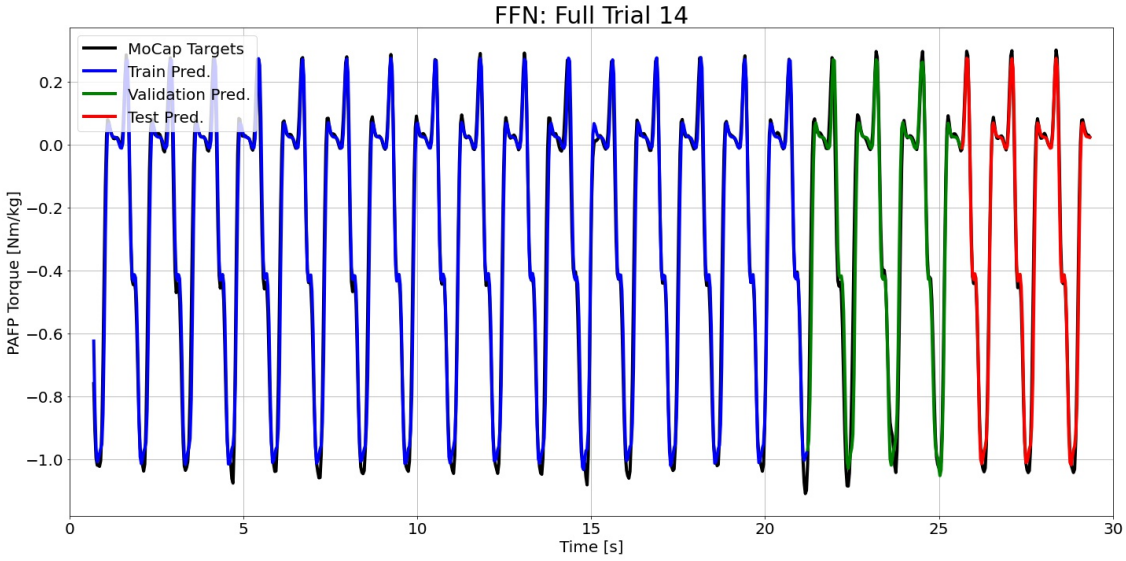


Figure I.29: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 14.

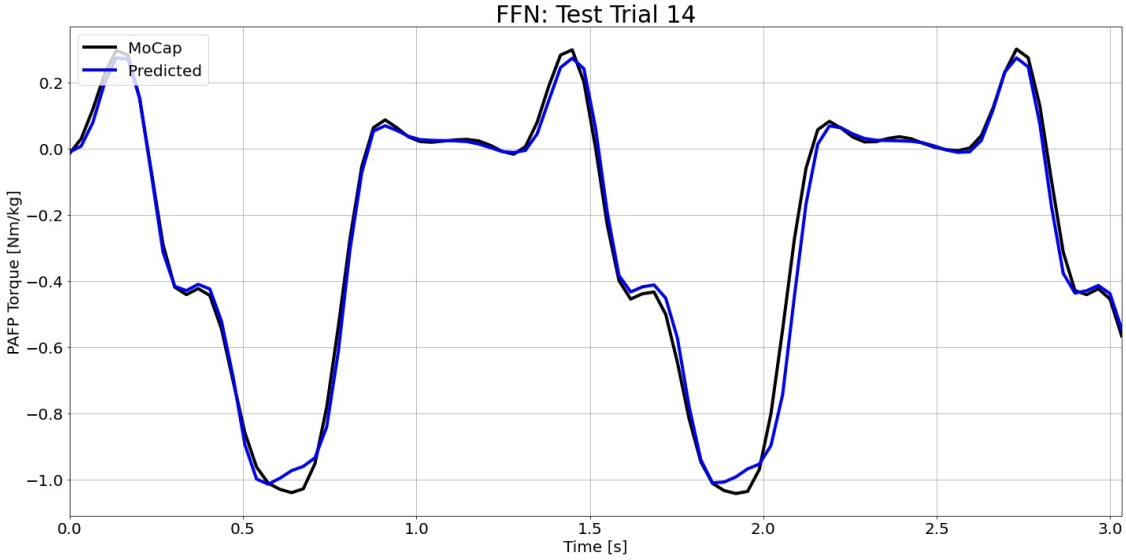


Figure I.30: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 14.

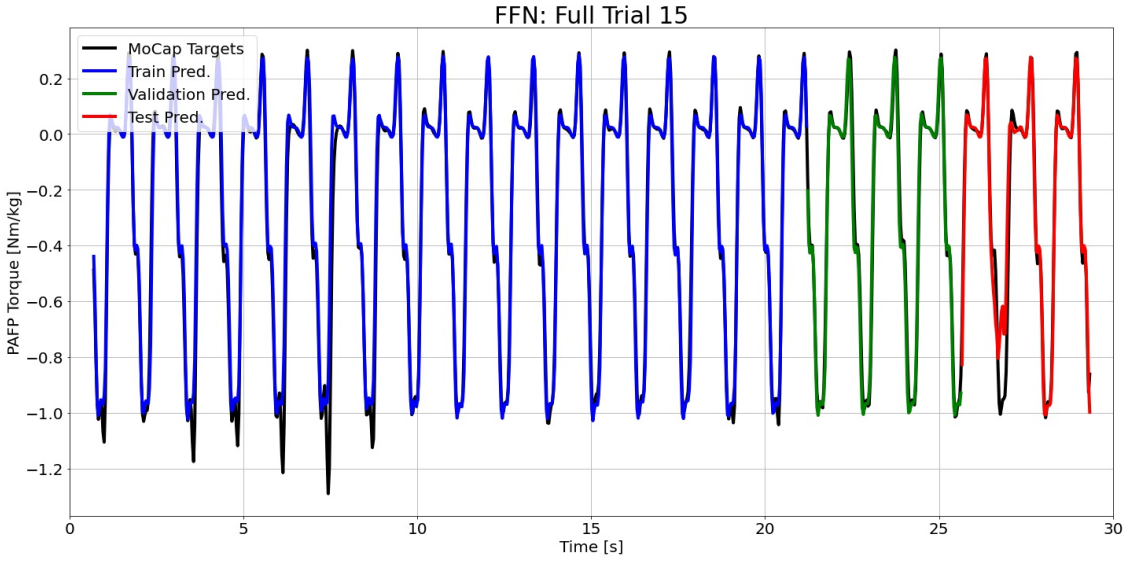


Figure I.31: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 15.

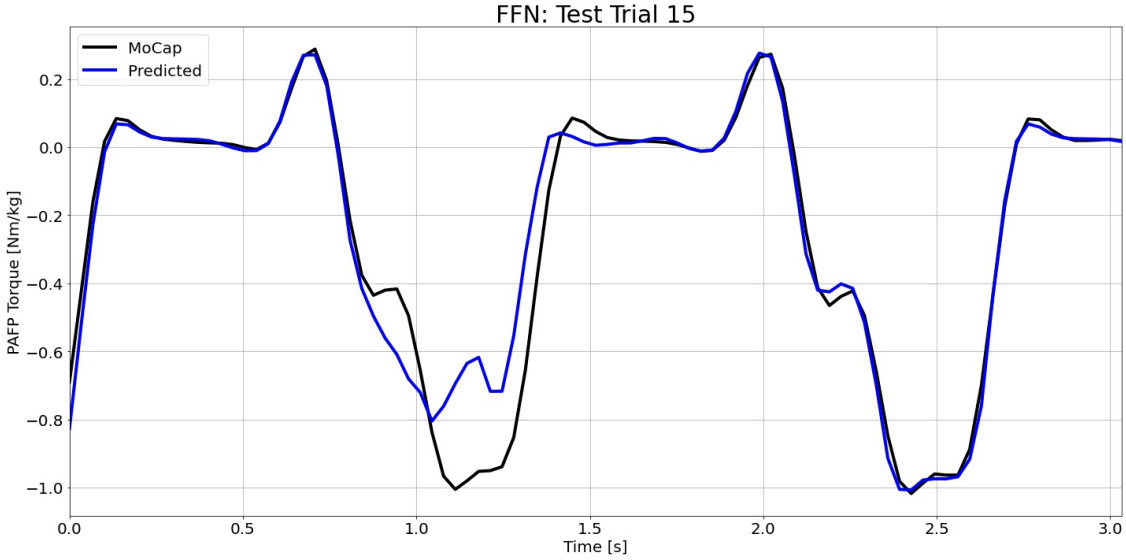


Figure I.32: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 15.

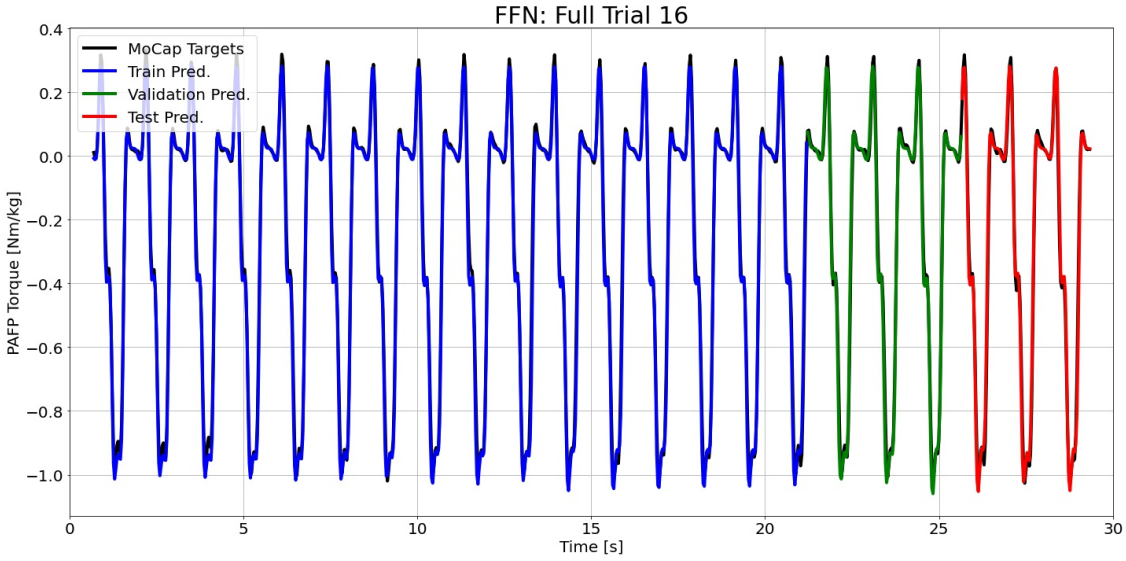


Figure I.33: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 16.

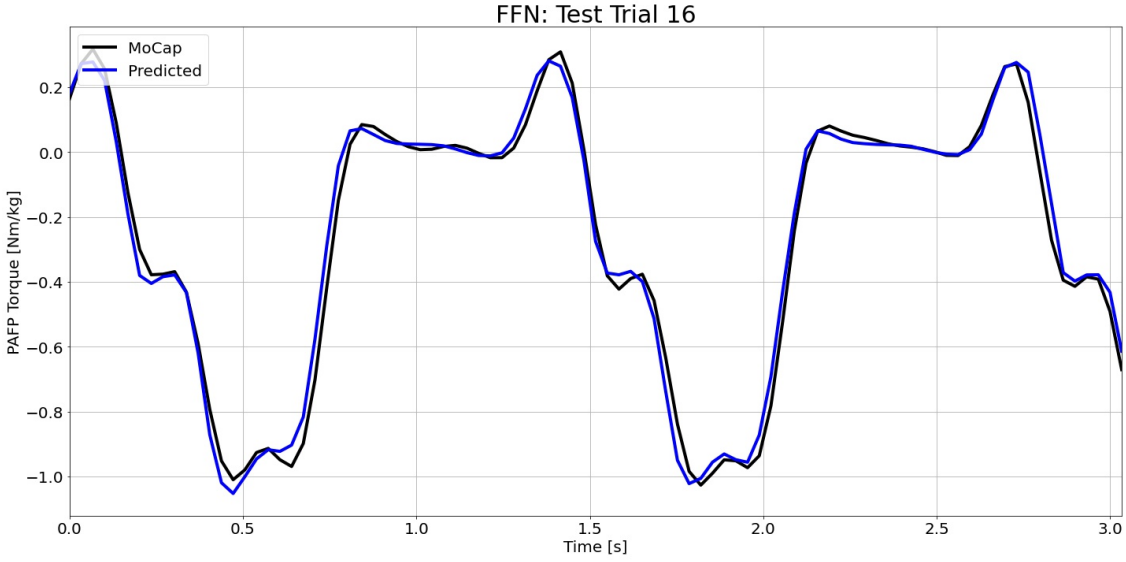


Figure I.34: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 16.

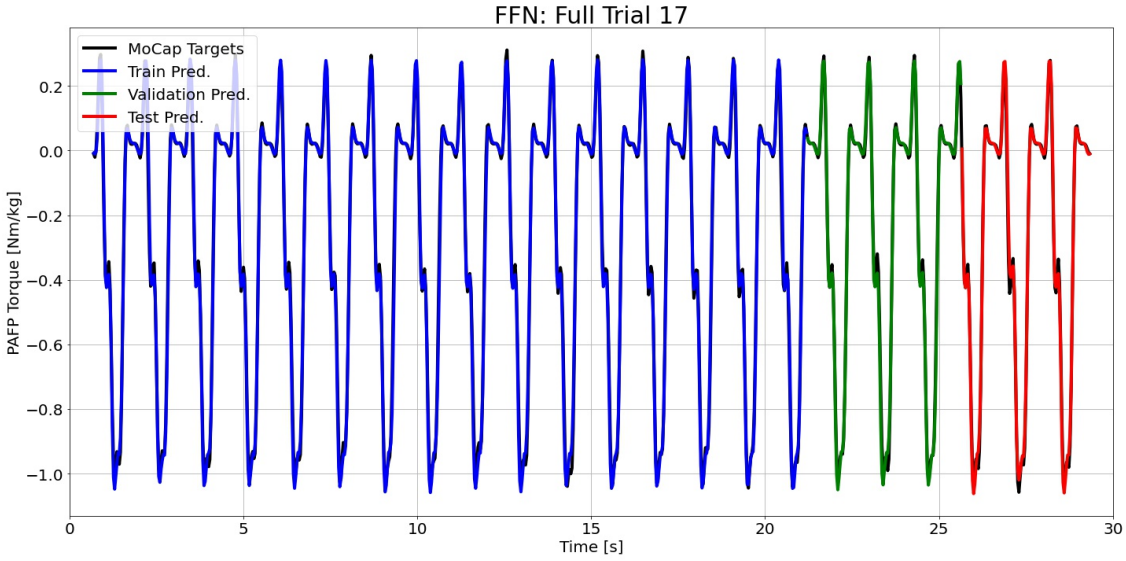


Figure I.35: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 17.

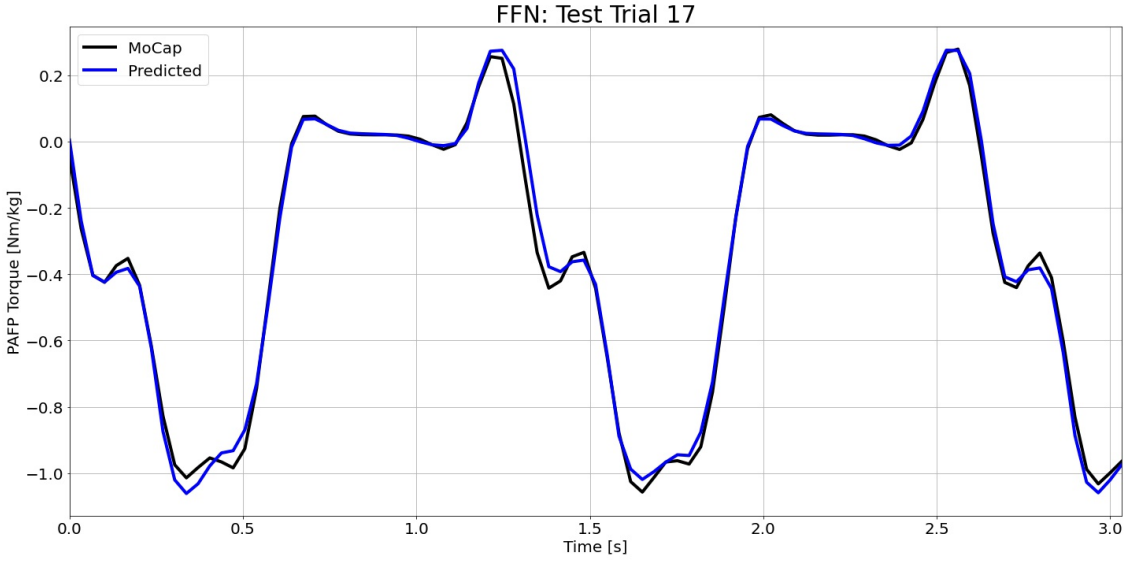


Figure I.36: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 17.

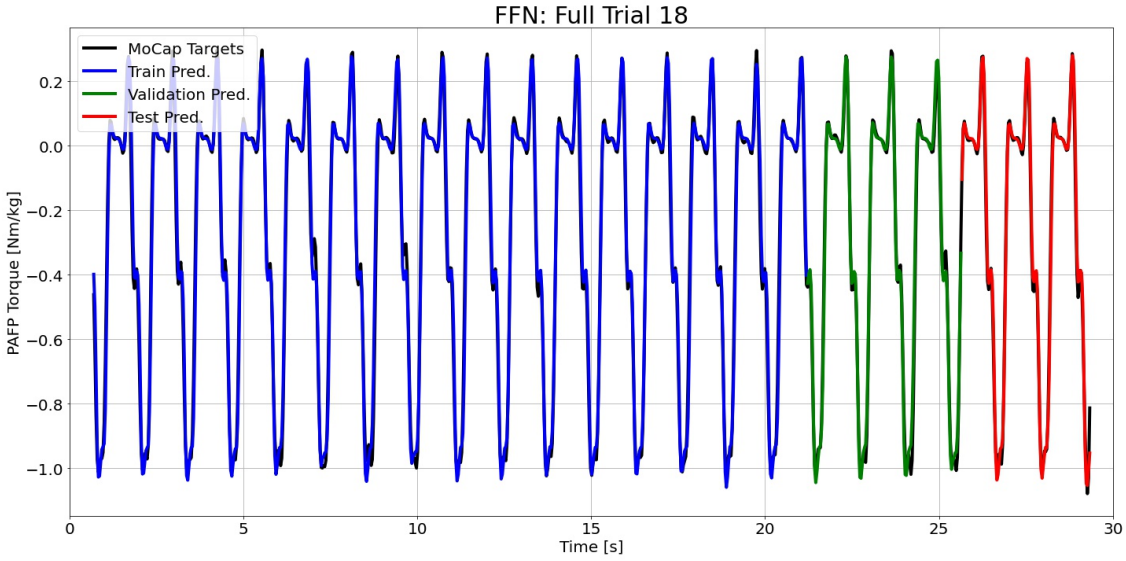


Figure I.37: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 18.

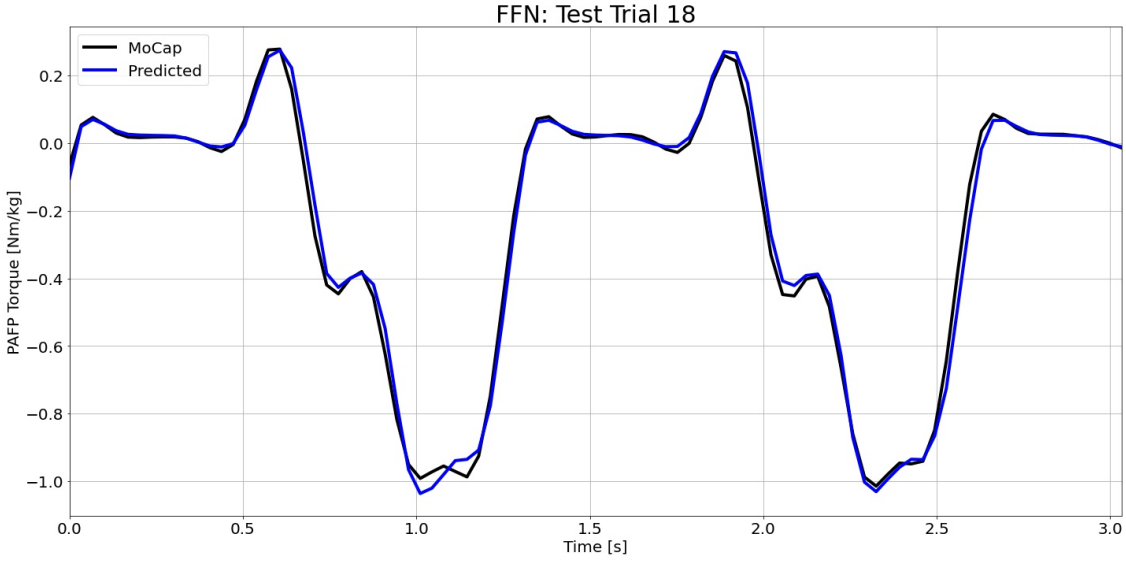


Figure I.38: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 18.

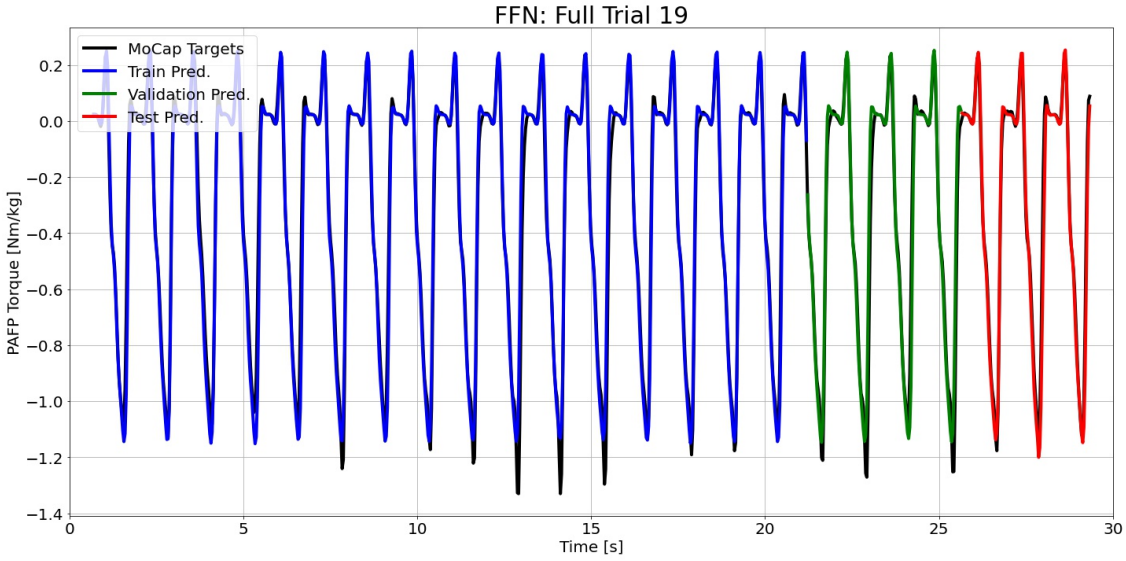


Figure I.39: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 19.

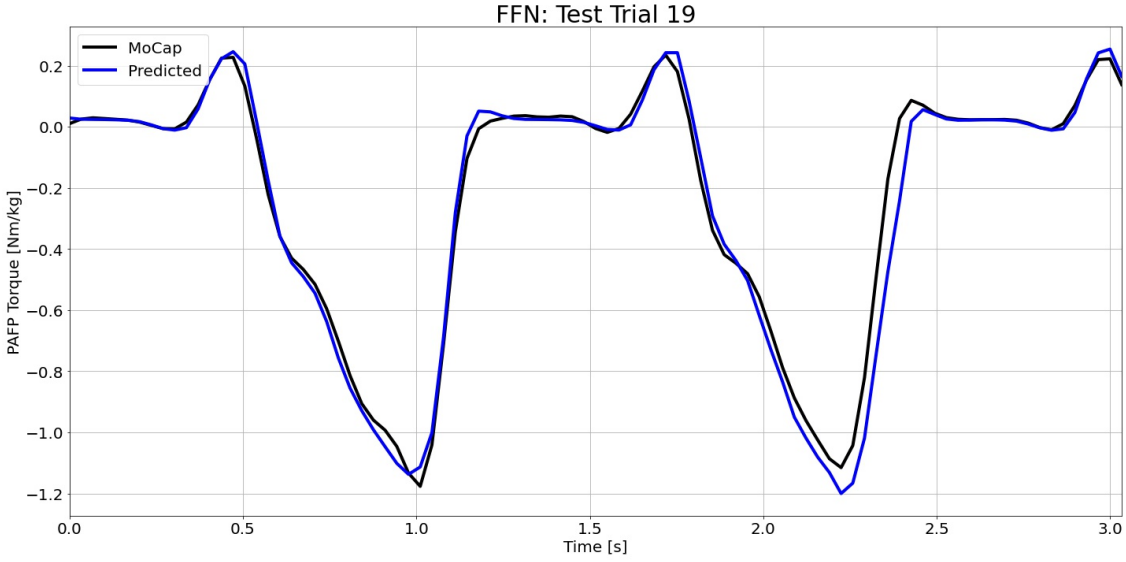


Figure I.40: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 19.

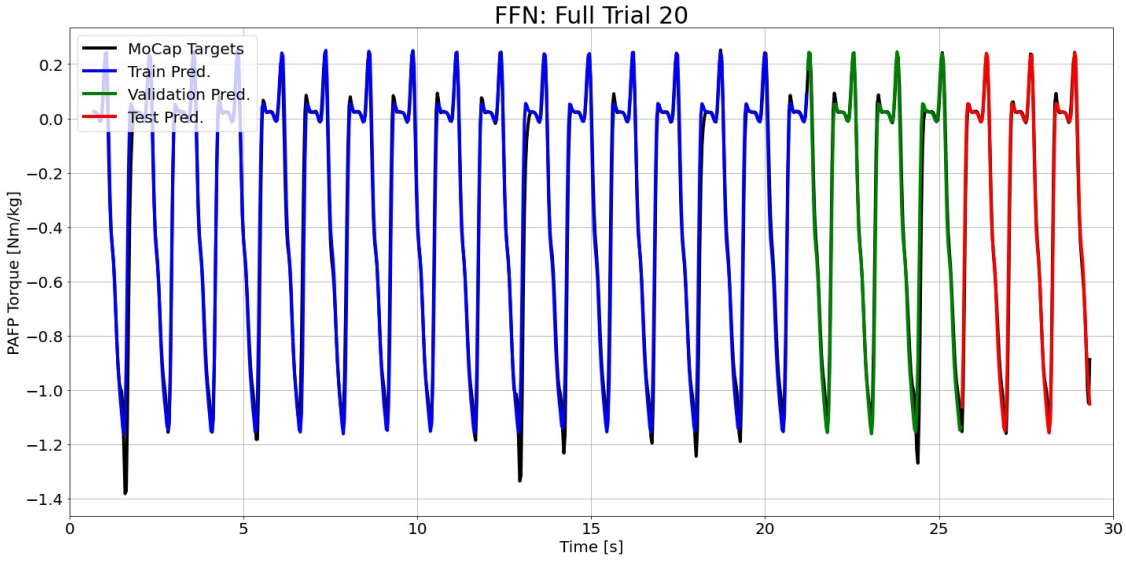


Figure I.41: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 20.

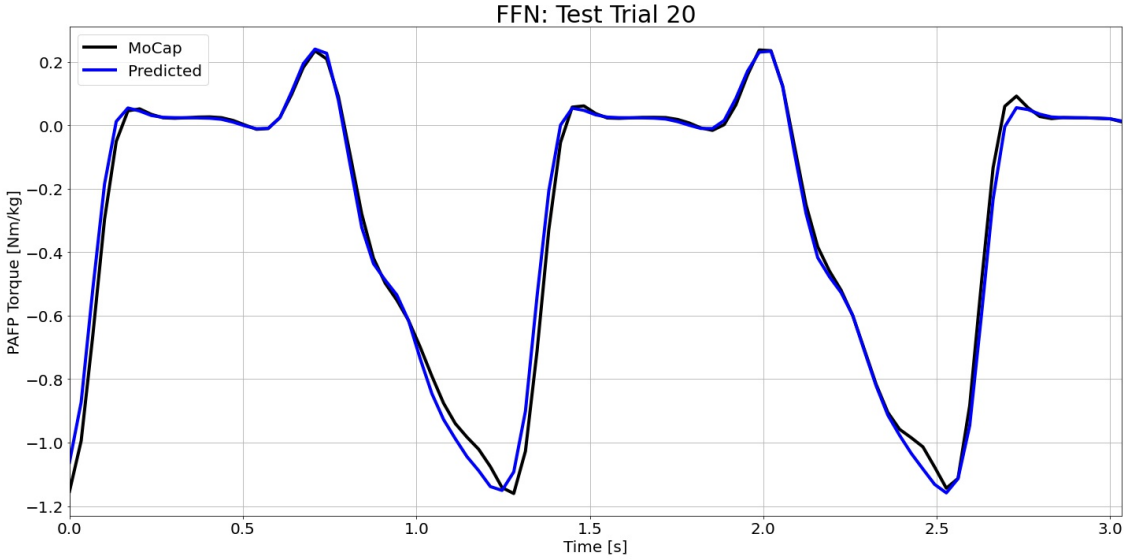


Figure I.42: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 20.

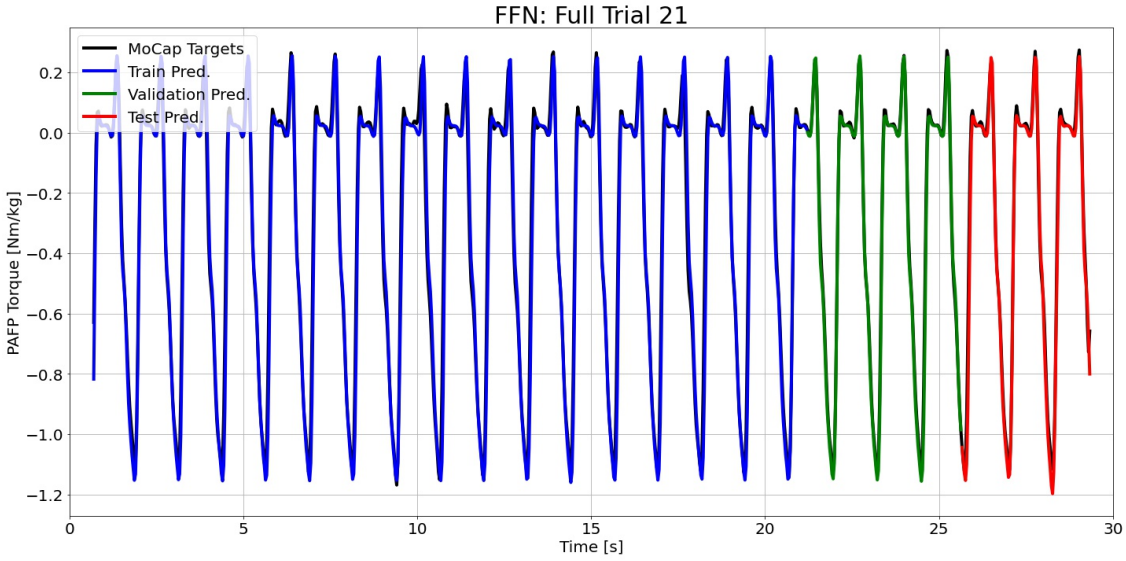


Figure I.43: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 21.

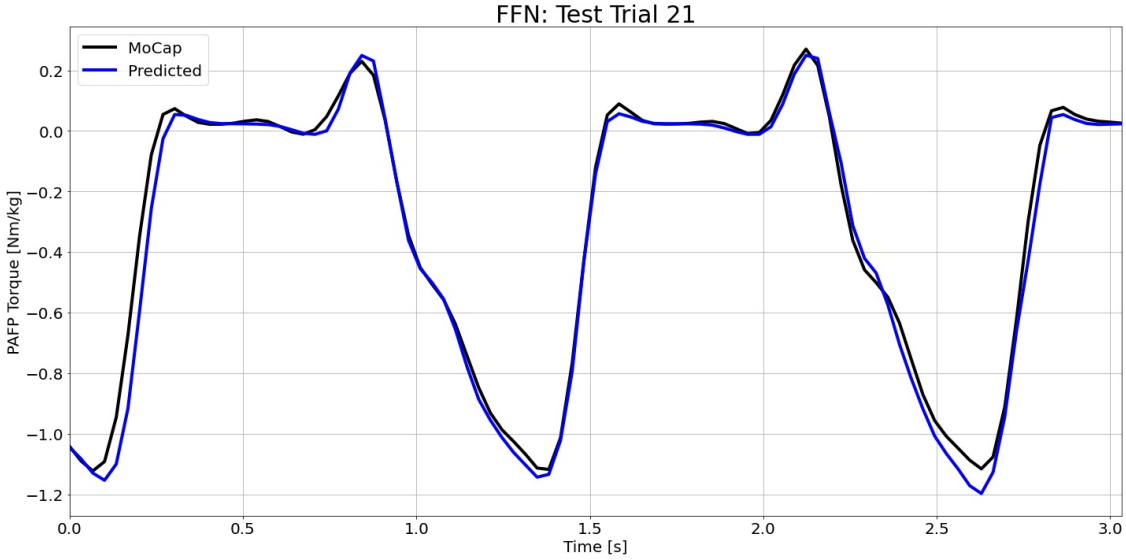


Figure I.44: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 21.

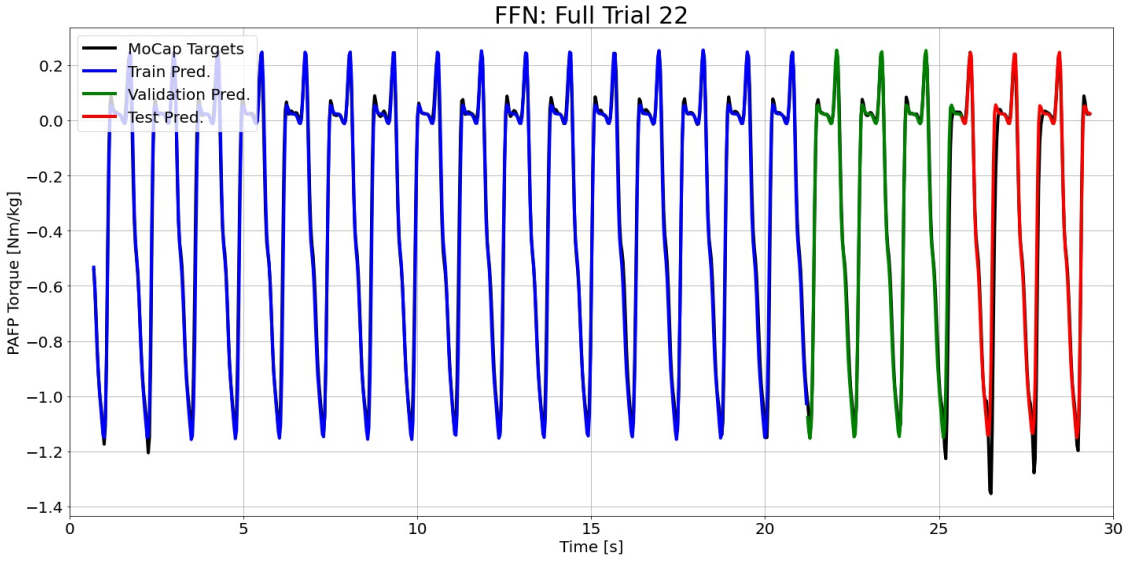


Figure I.45: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 22.

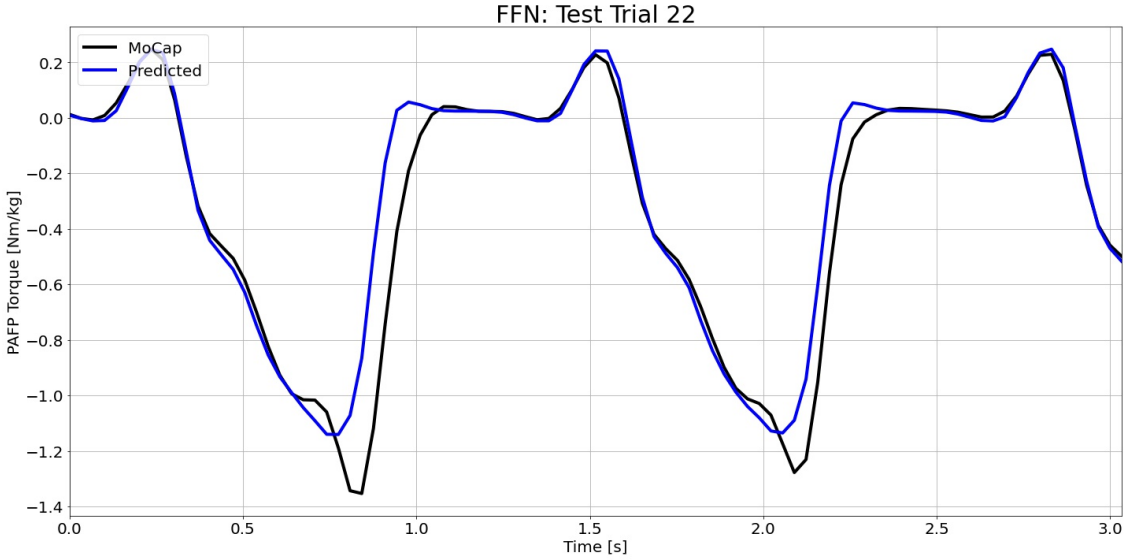


Figure I.46: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 22.

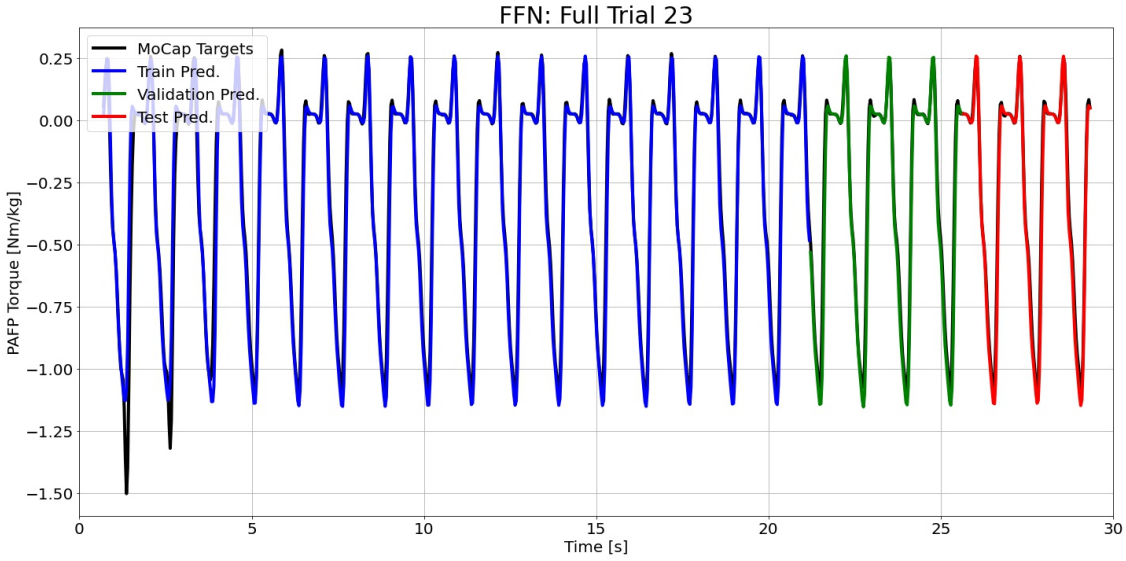


Figure I.47: FFN predictions compared to the MoCap inverse dynamics ground truth values for full time series 23.

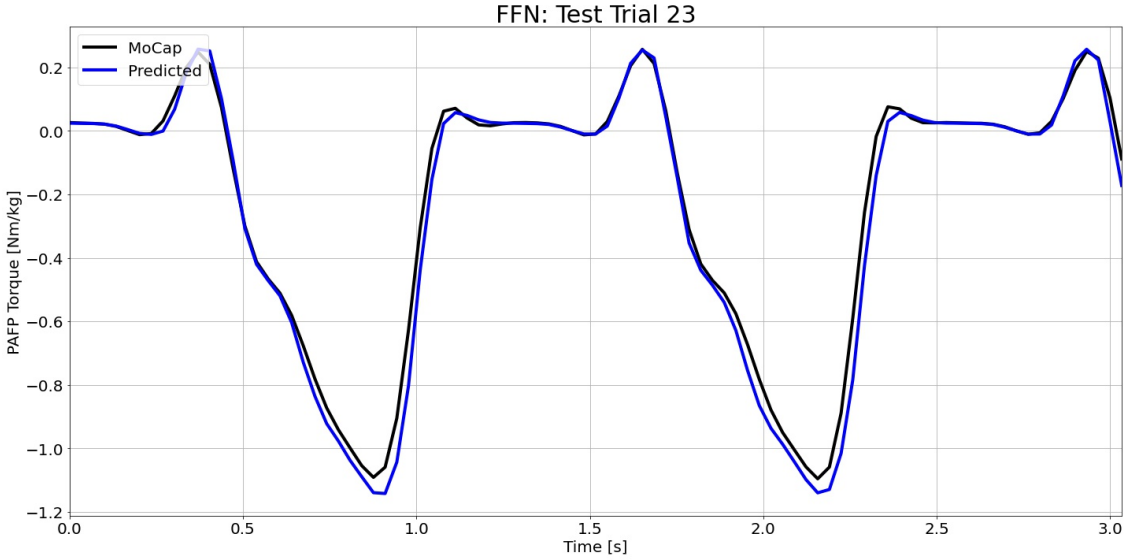


Figure I.48: FFN predictions compared to the MoCap inverse dynamics ground truth values for test time series 23.

I.2 Gated Recurrent Unit Neural Network

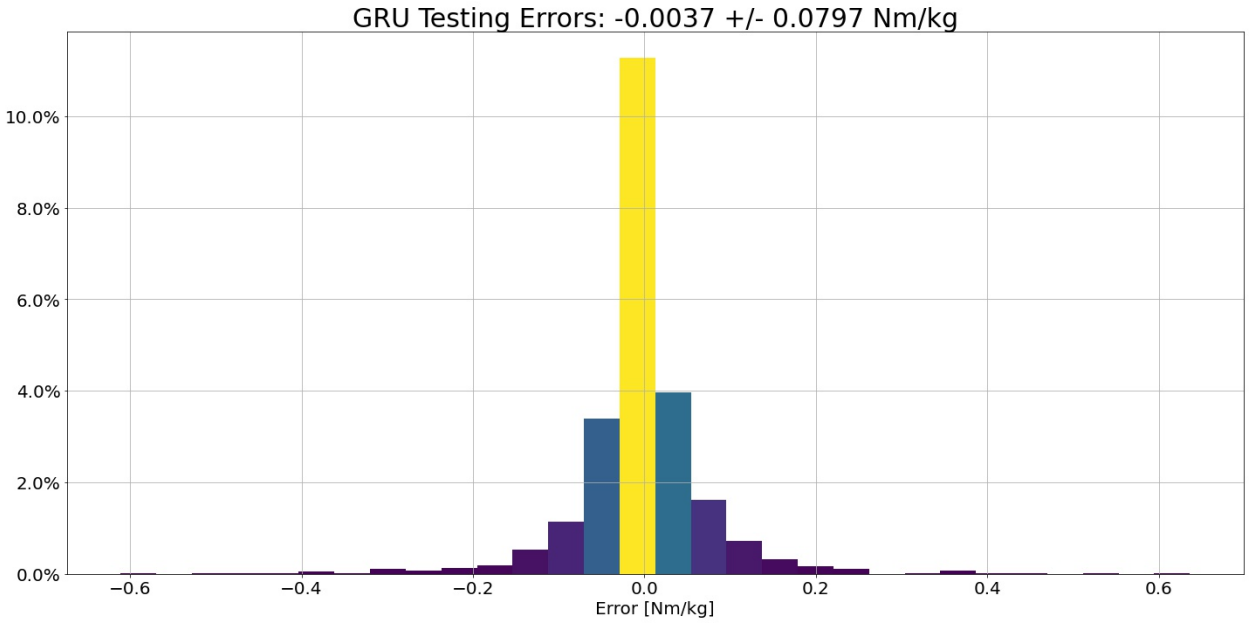


Figure I.49: GRU prediction error histogram for all test time series data.

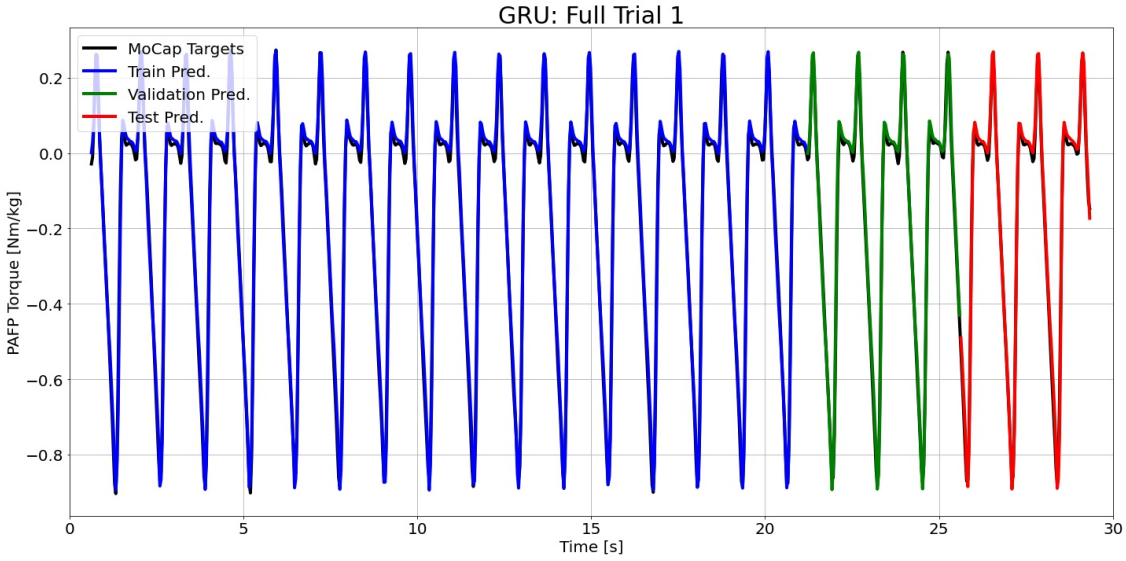


Figure I.50: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 1.

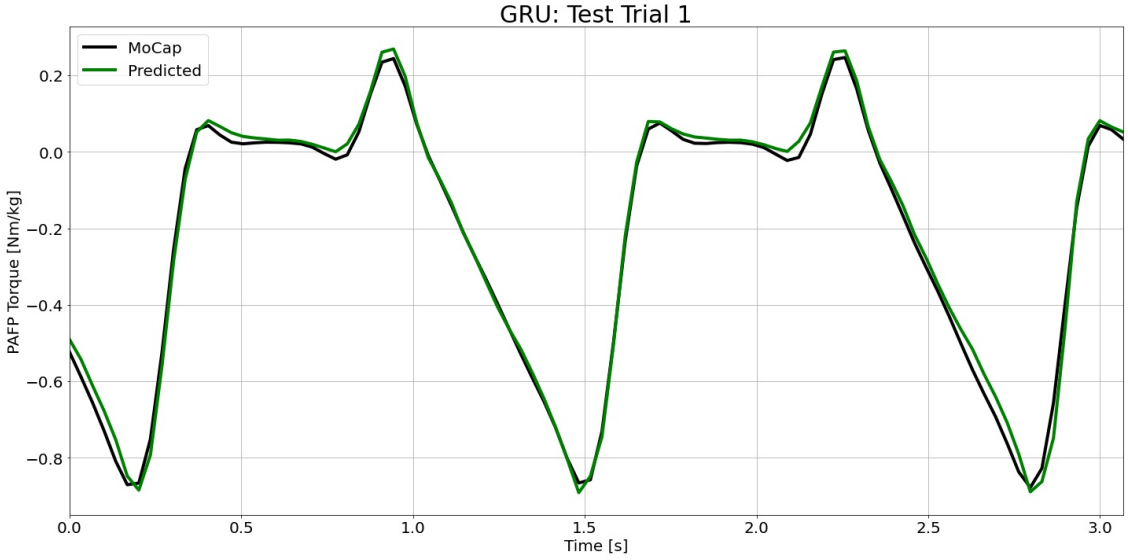


Figure I.51: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 1.

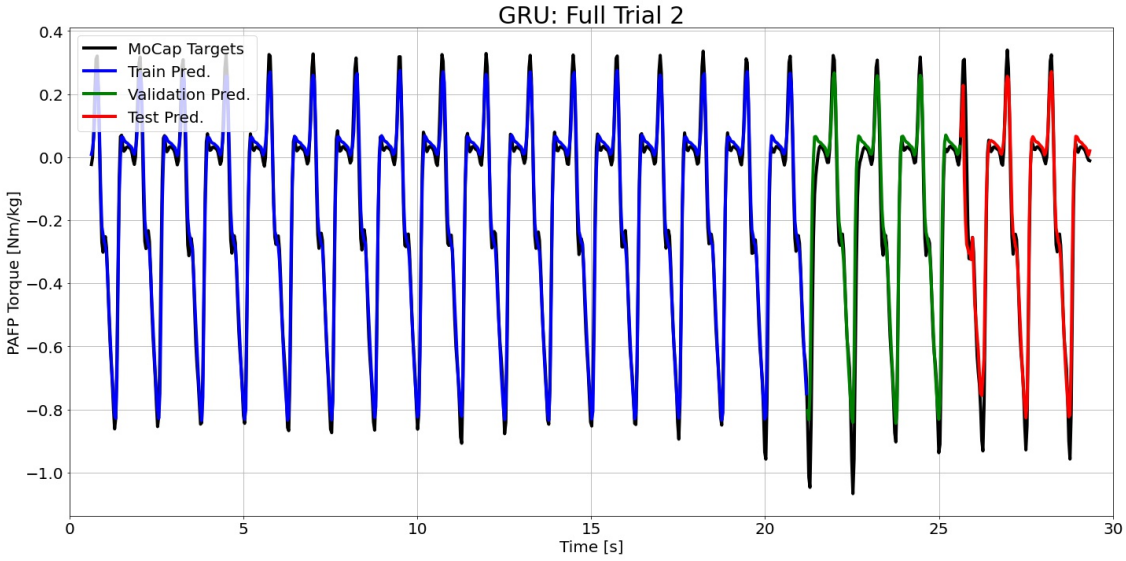


Figure I.52: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 2.

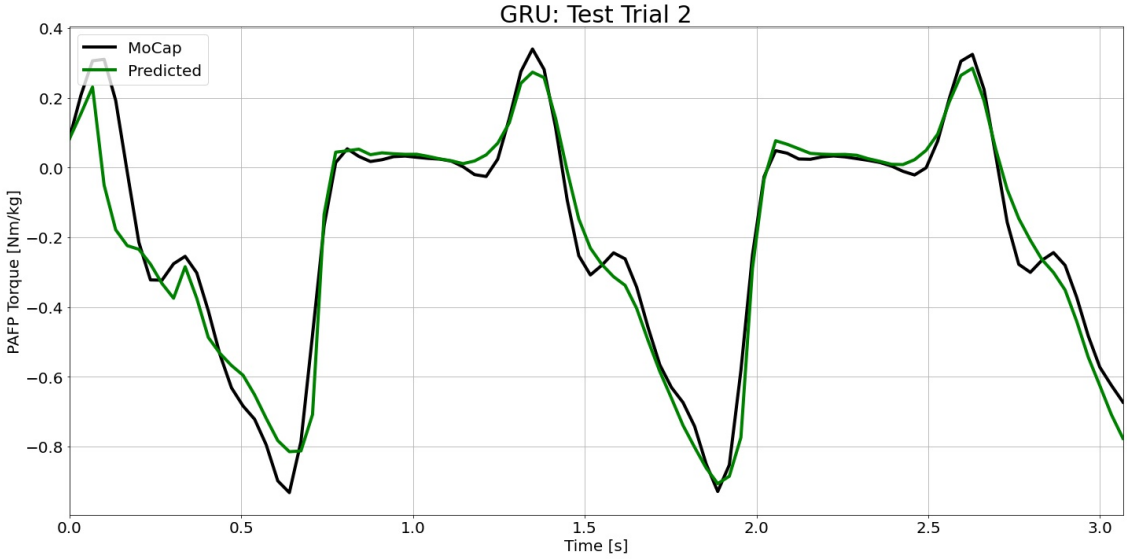


Figure I.53: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 2.

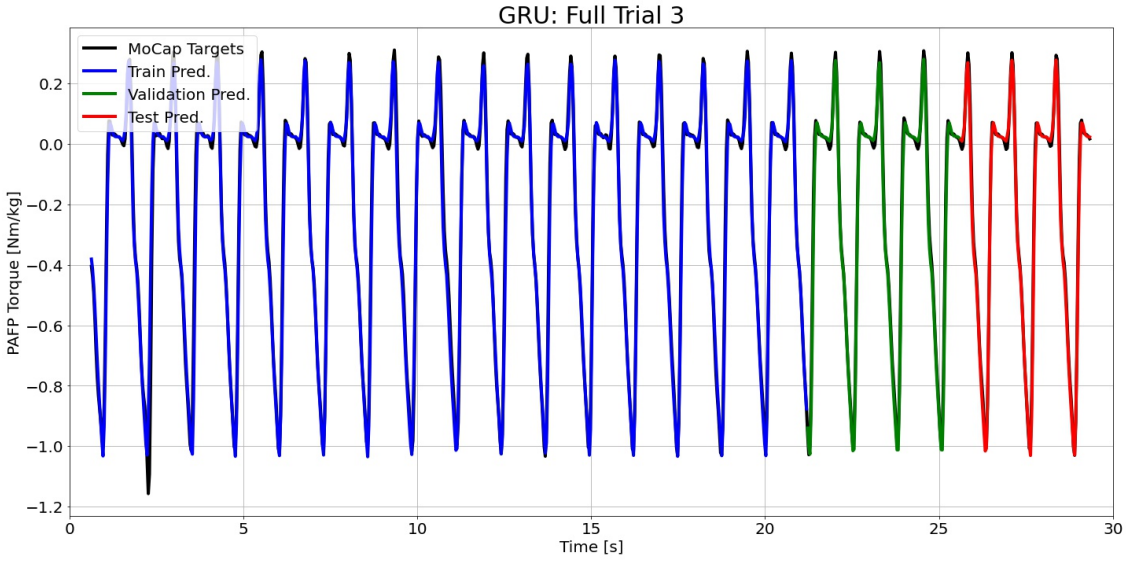


Figure I.54: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 3.

Figure I.55: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 3.

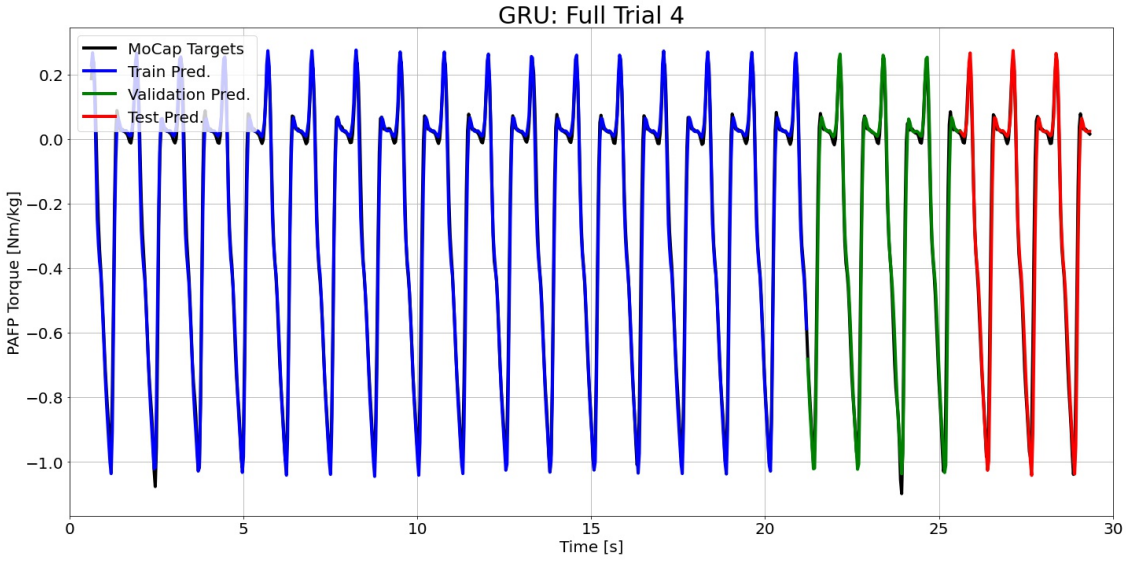


Figure I.56: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 4.

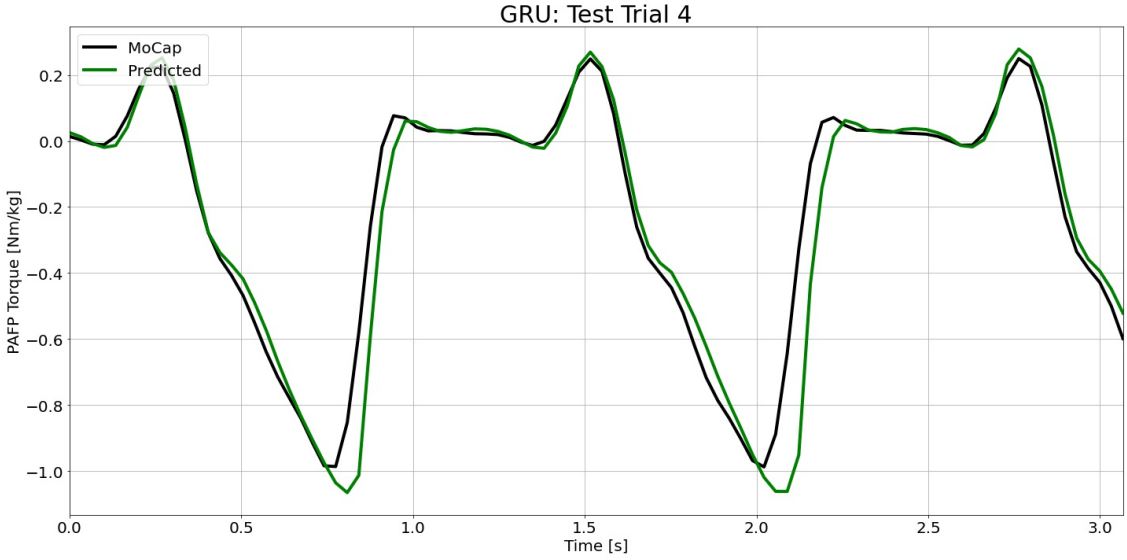


Figure I.57: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 4.

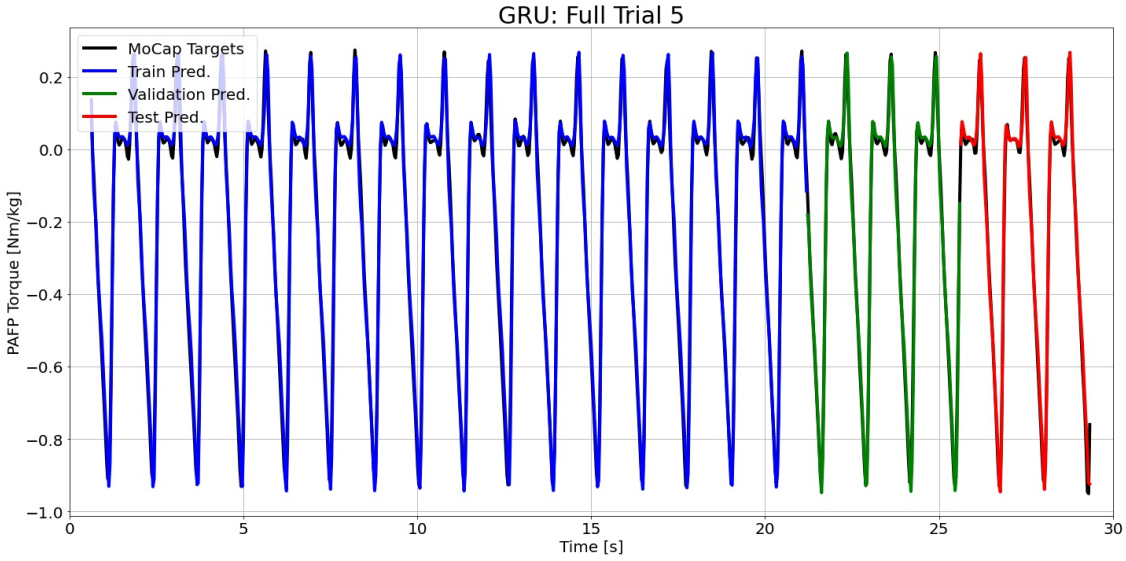


Figure I.58: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 5.

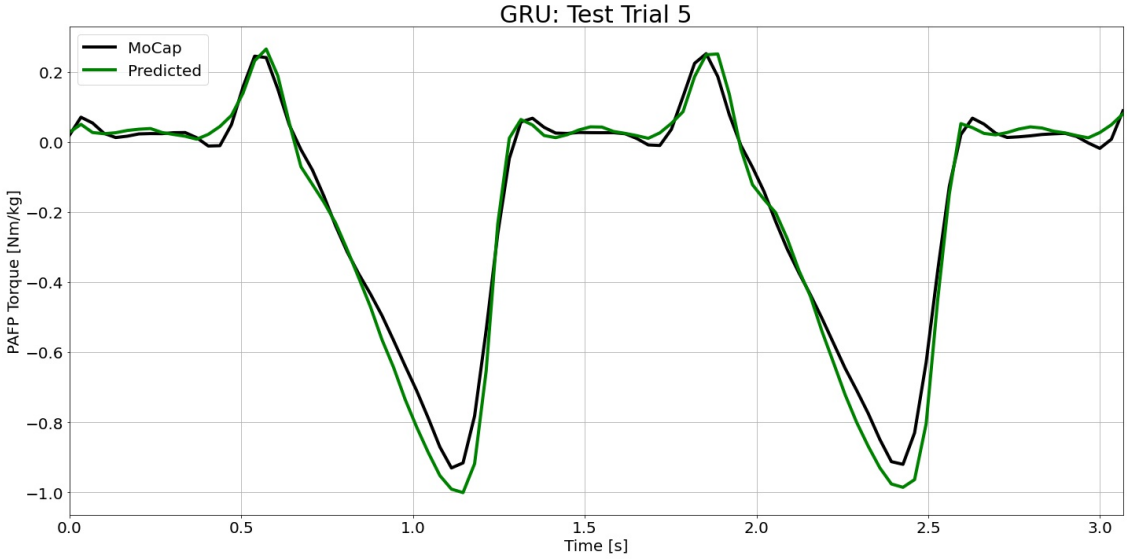


Figure I.59: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 5.

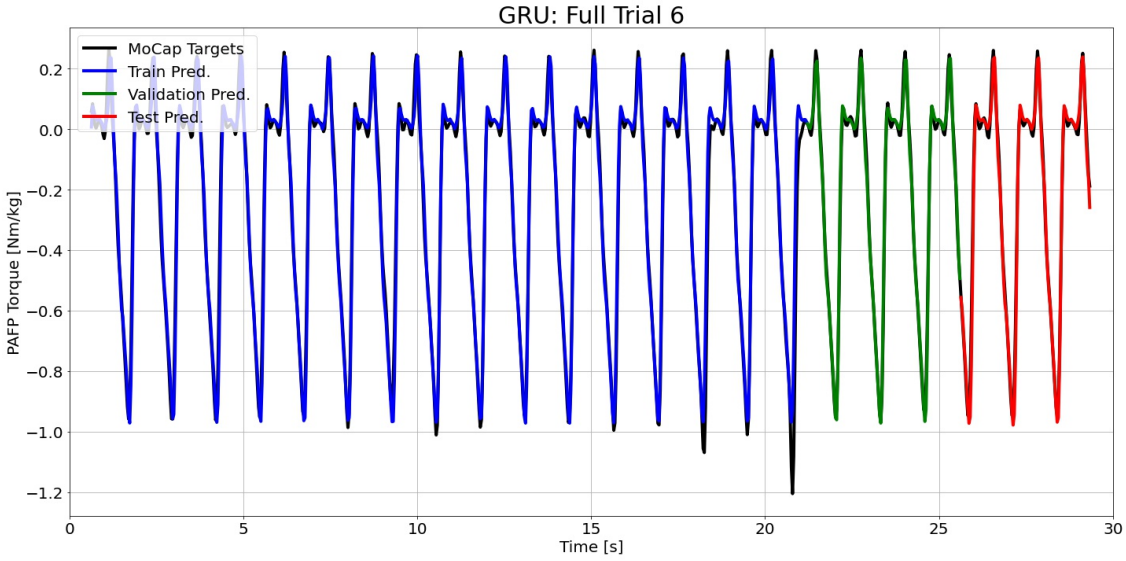


Figure I.60: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 6.

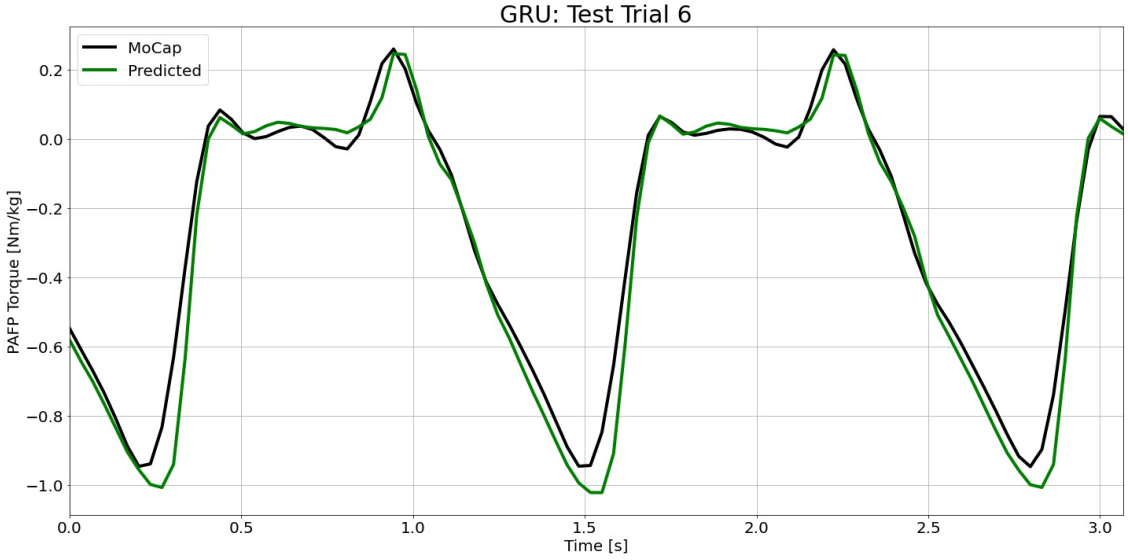


Figure I.61: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 6.

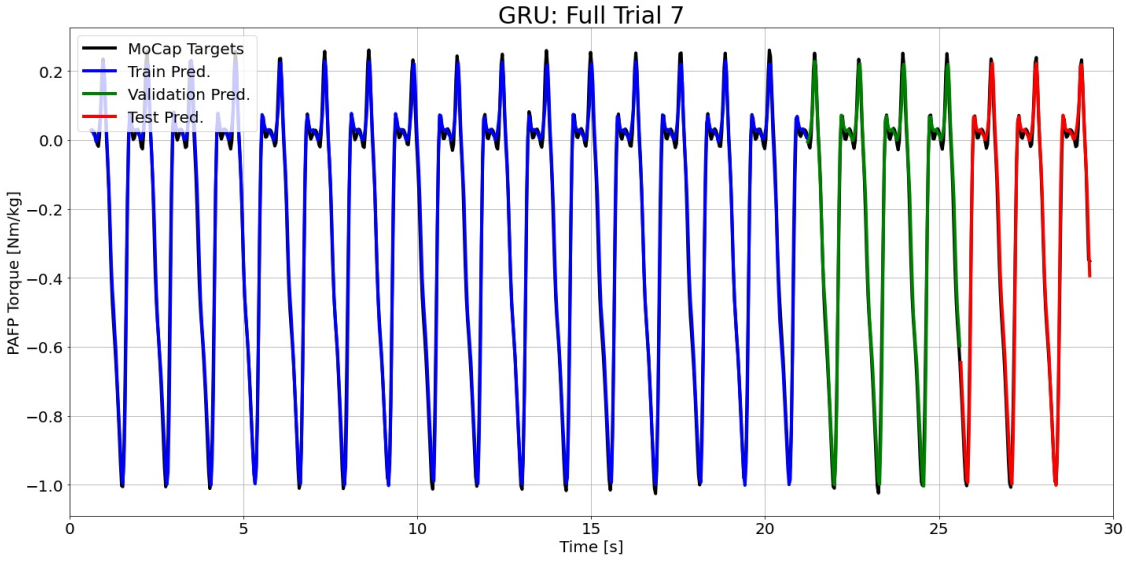


Figure I.62: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 7.

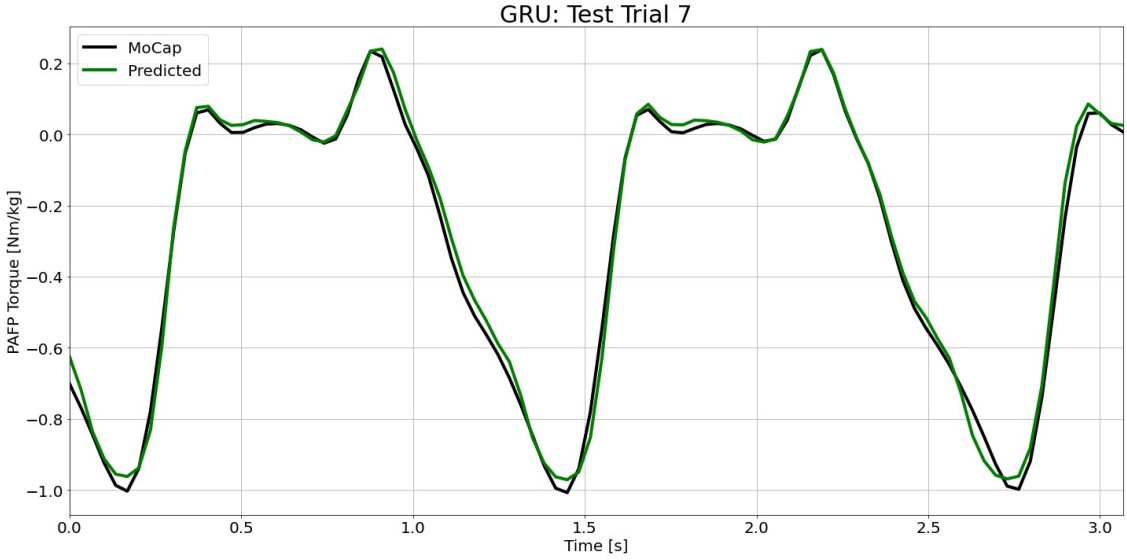


Figure I.63: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 7.

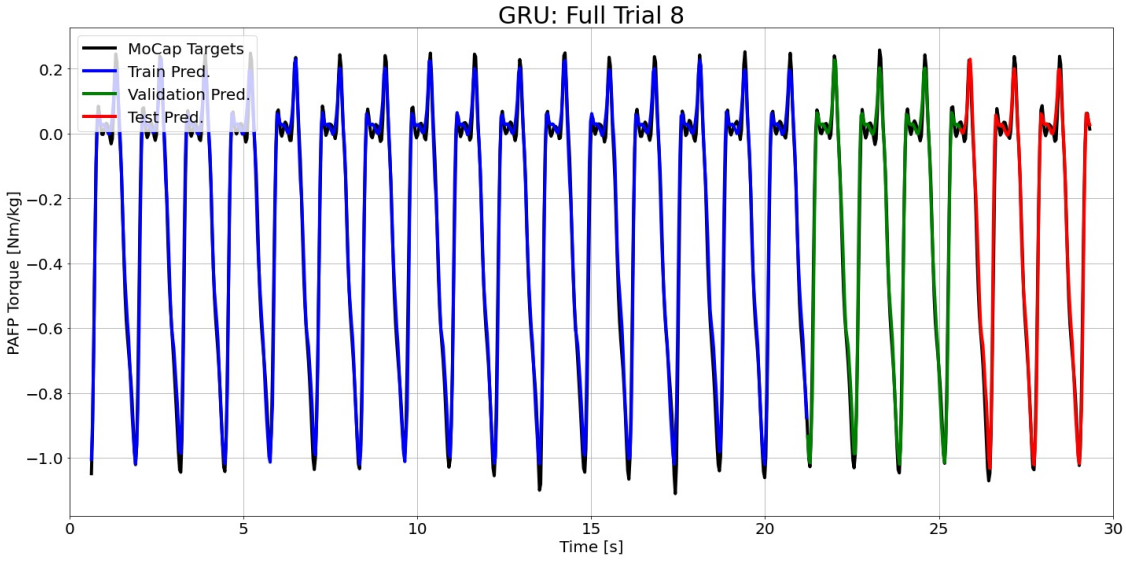


Figure I.64: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 8.

Figure I.65: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 8.

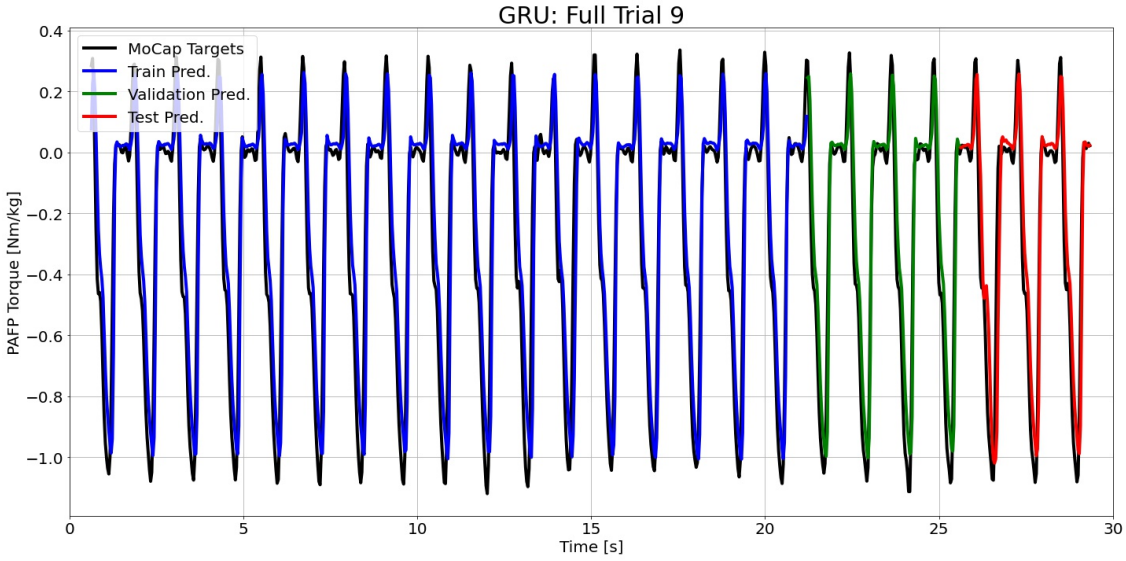


Figure I.66: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 9.

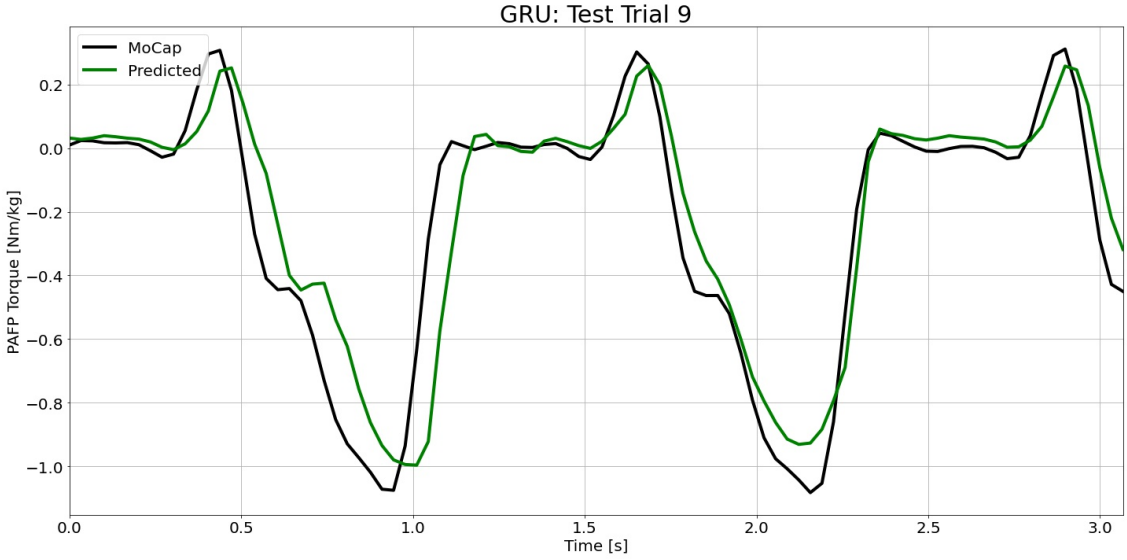


Figure I.67: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 9.

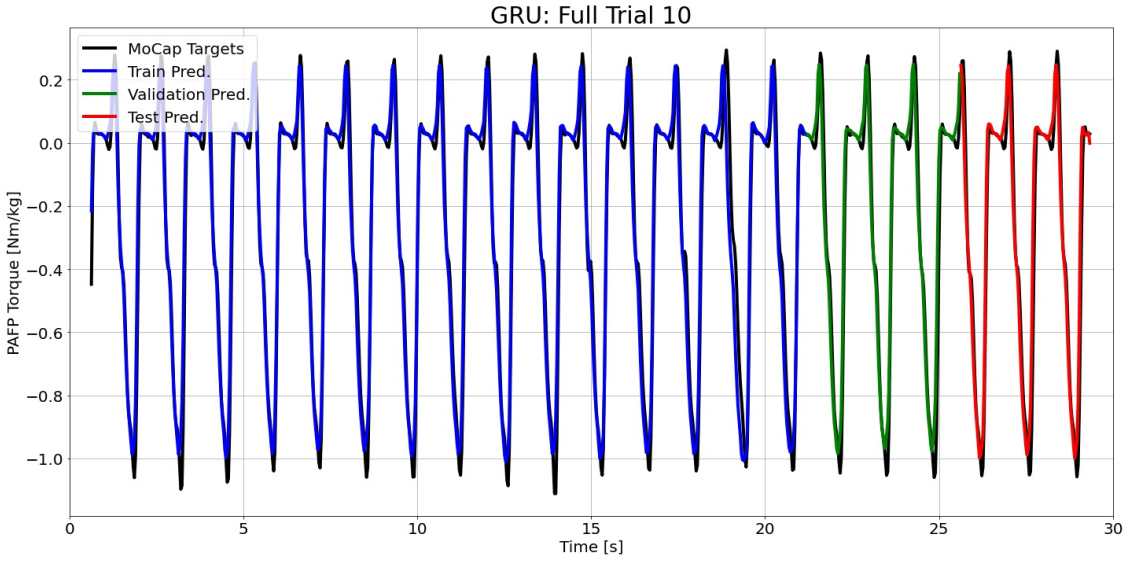


Figure I.68: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 10.

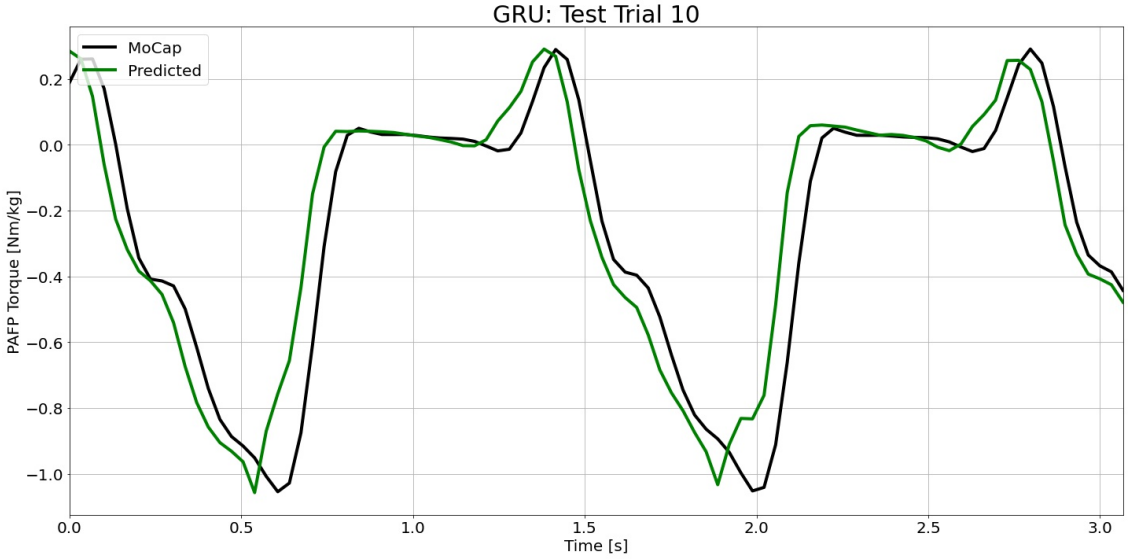


Figure I.69: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 10.

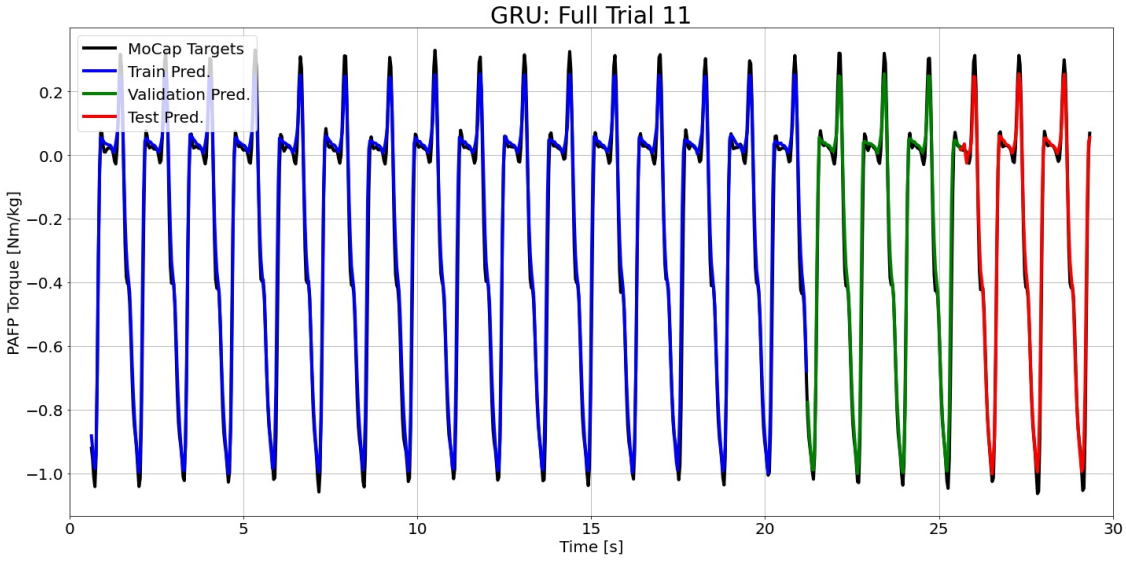


Figure I.70: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 11.

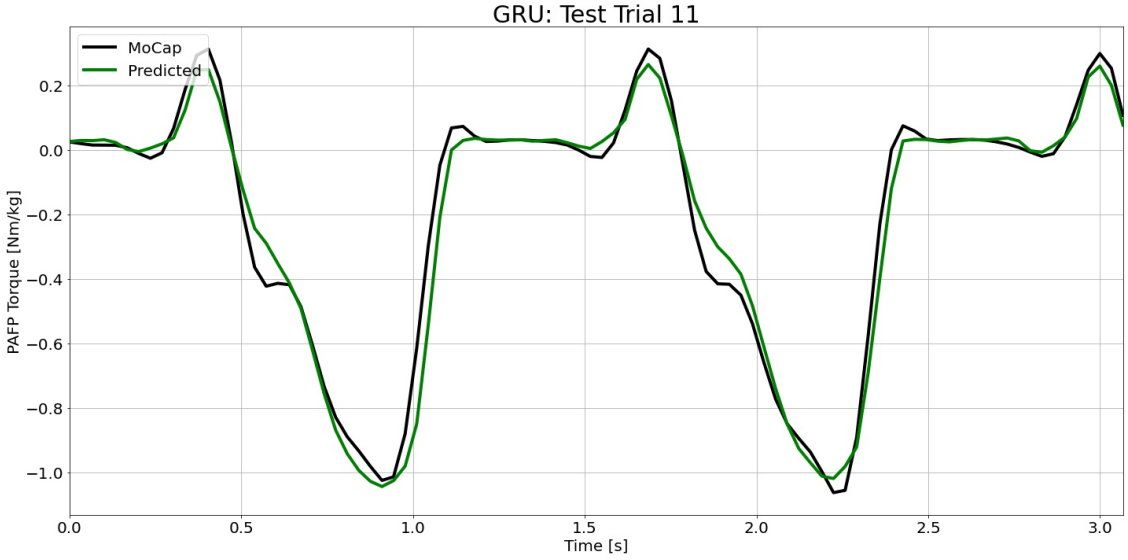


Figure I.71: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 11.

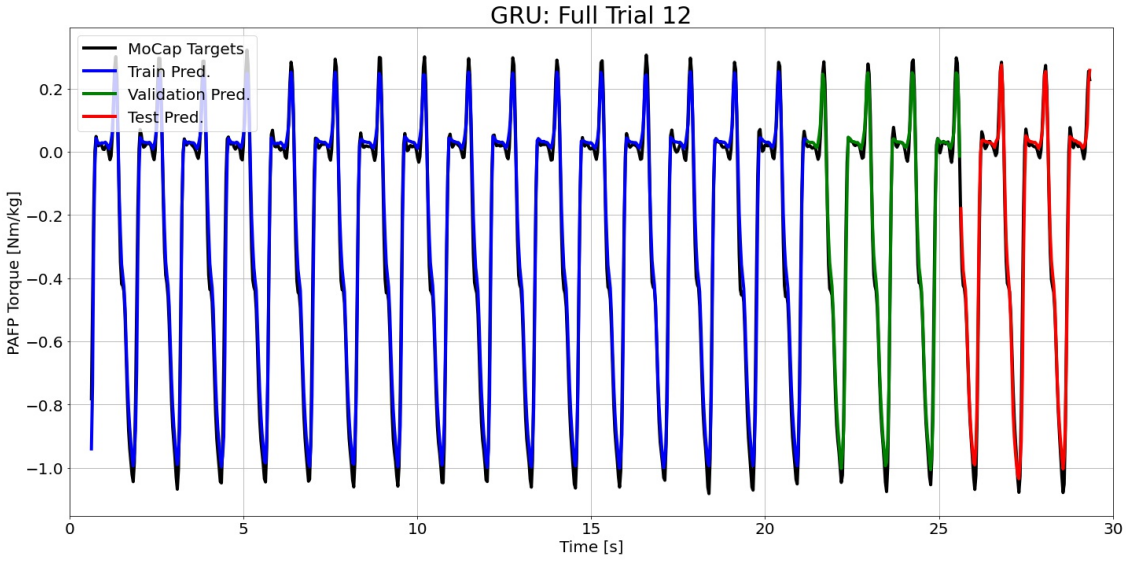


Figure I.72: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 12.

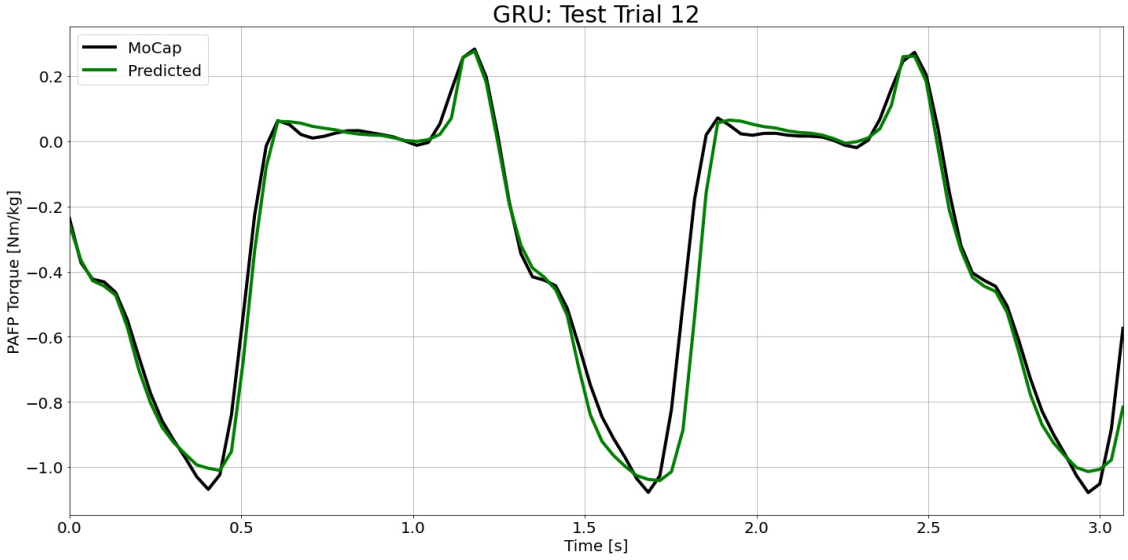


Figure I.73: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 12.

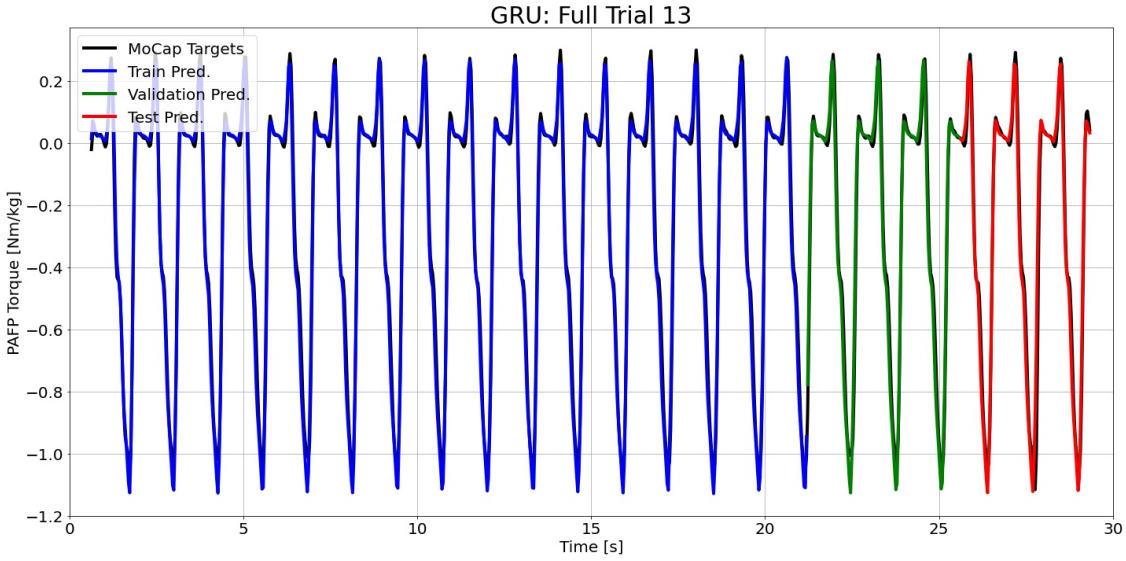


Figure I.74: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 13.

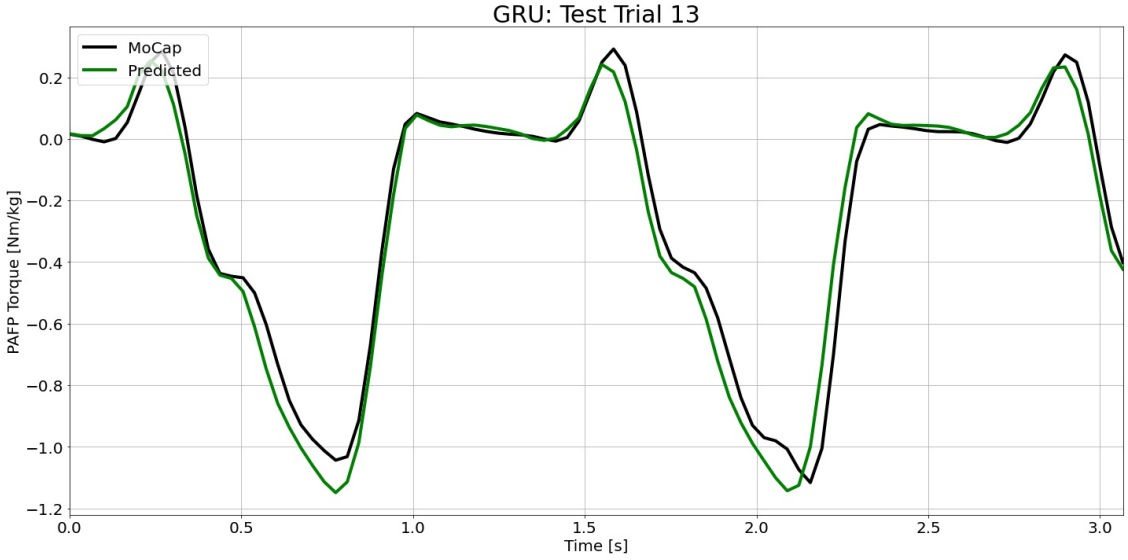


Figure I.75: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 13.

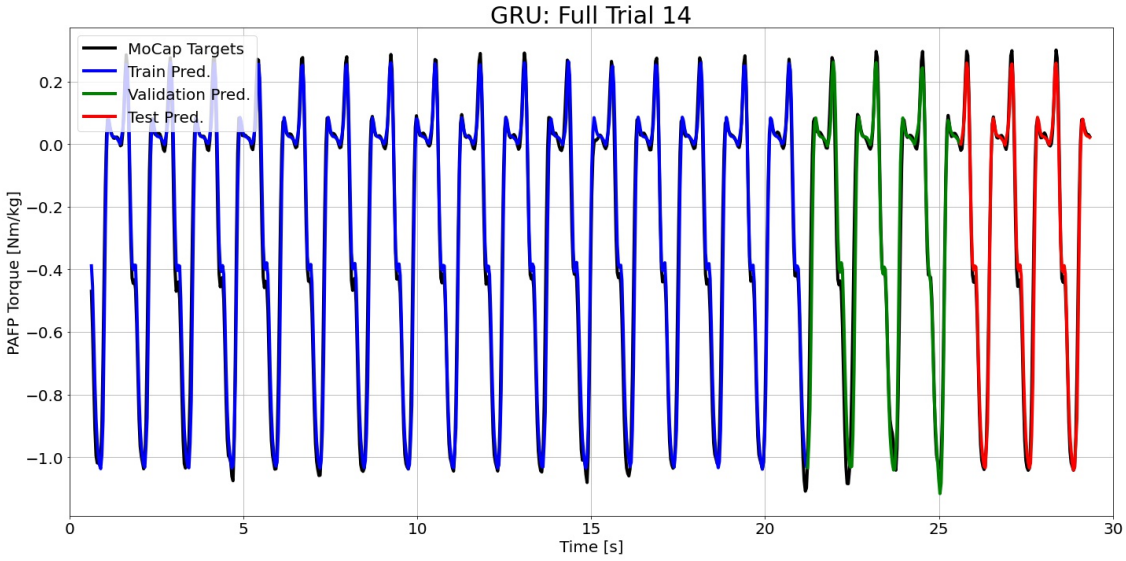


Figure I.76: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 14.

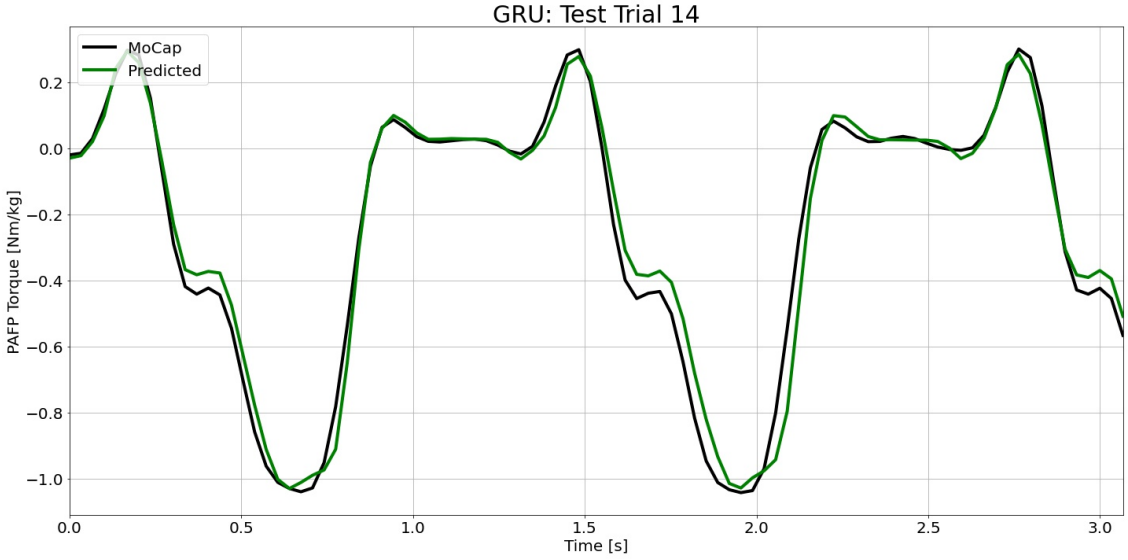


Figure I.77: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 14.

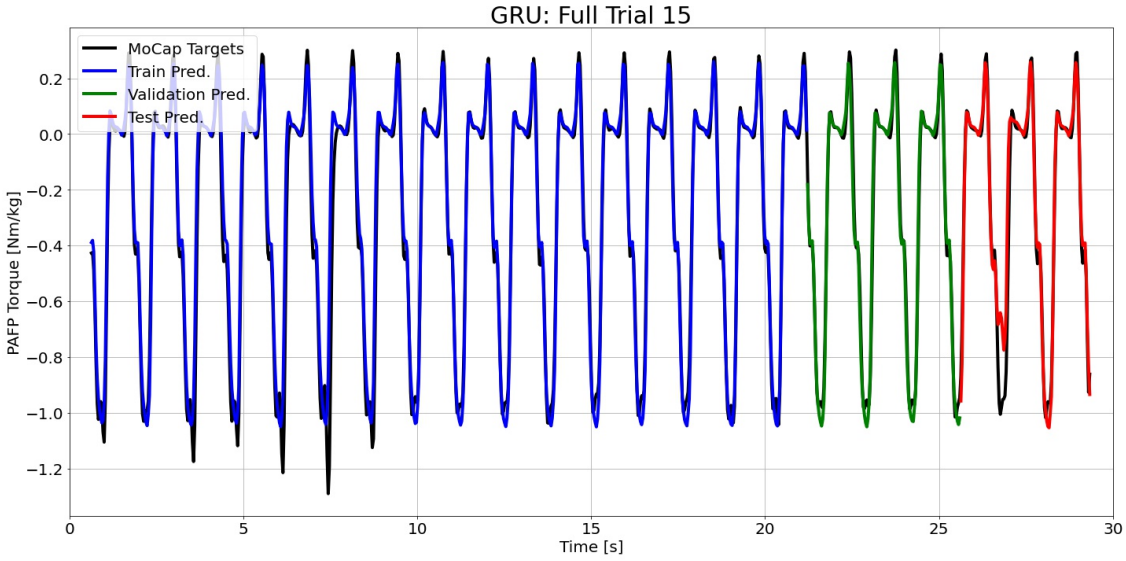


Figure I.78: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 15.

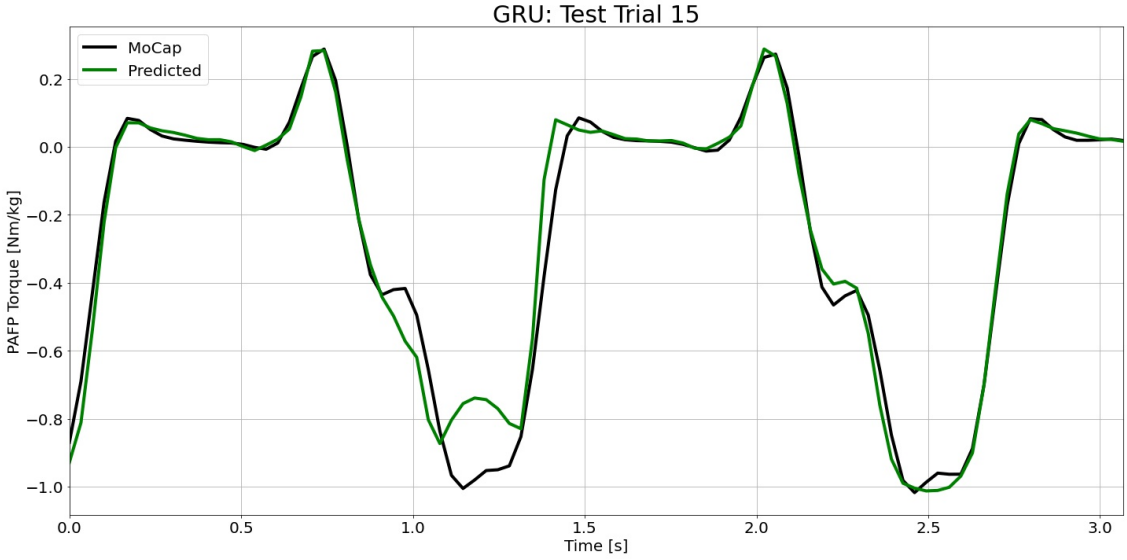


Figure I.79: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 15.

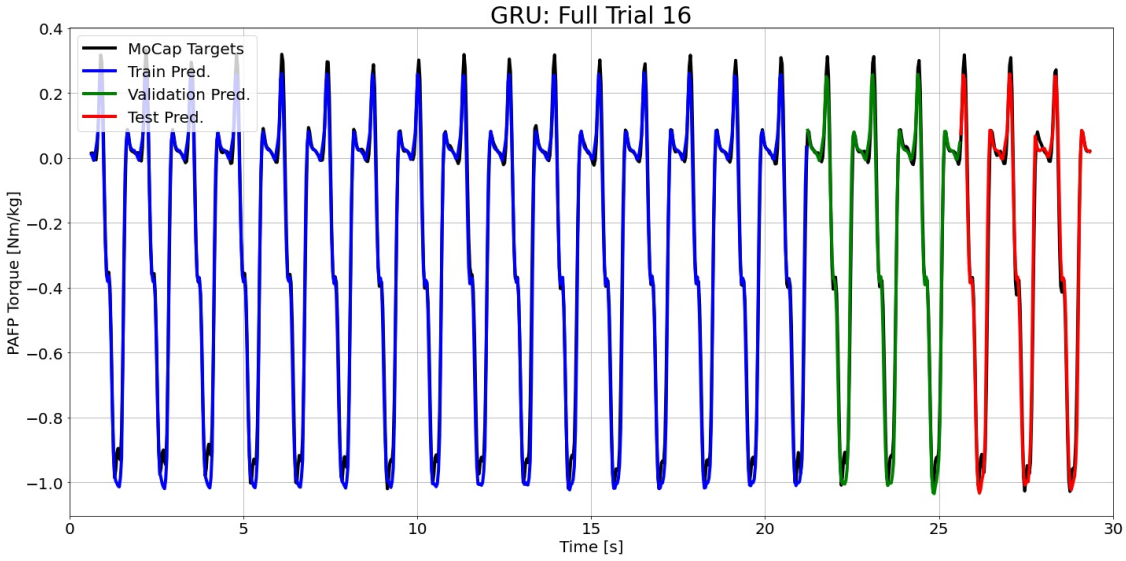


Figure I.80: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 16.

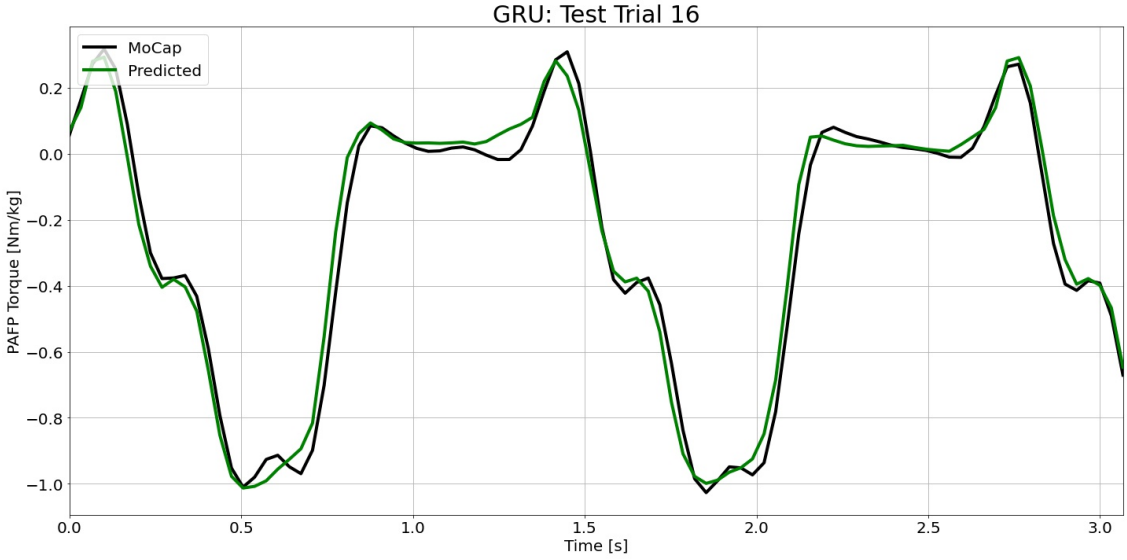


Figure I.81: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 16.

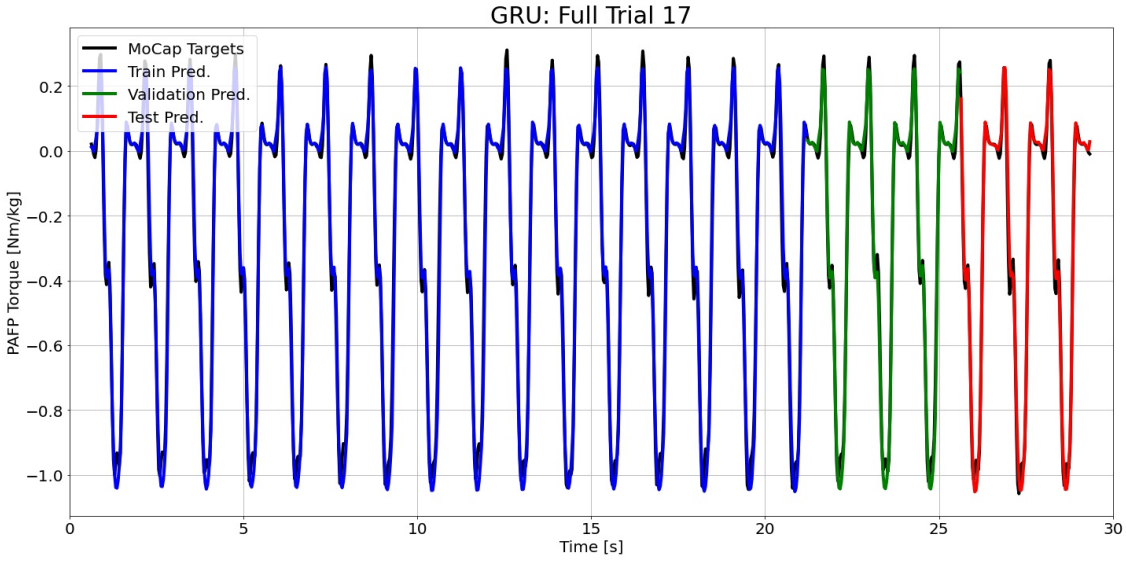


Figure I.82: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 17.

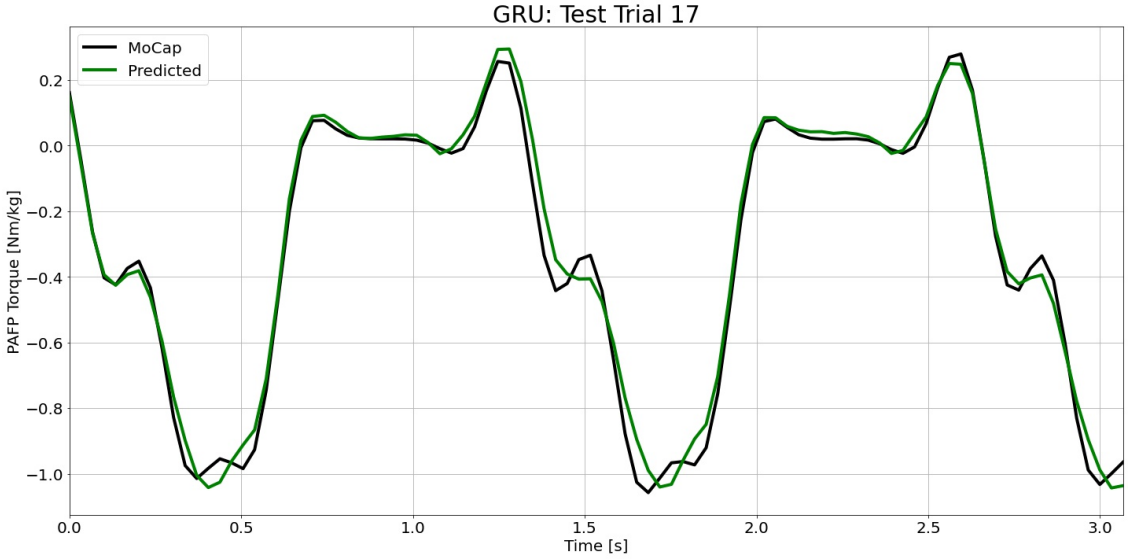


Figure I.83: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 17.

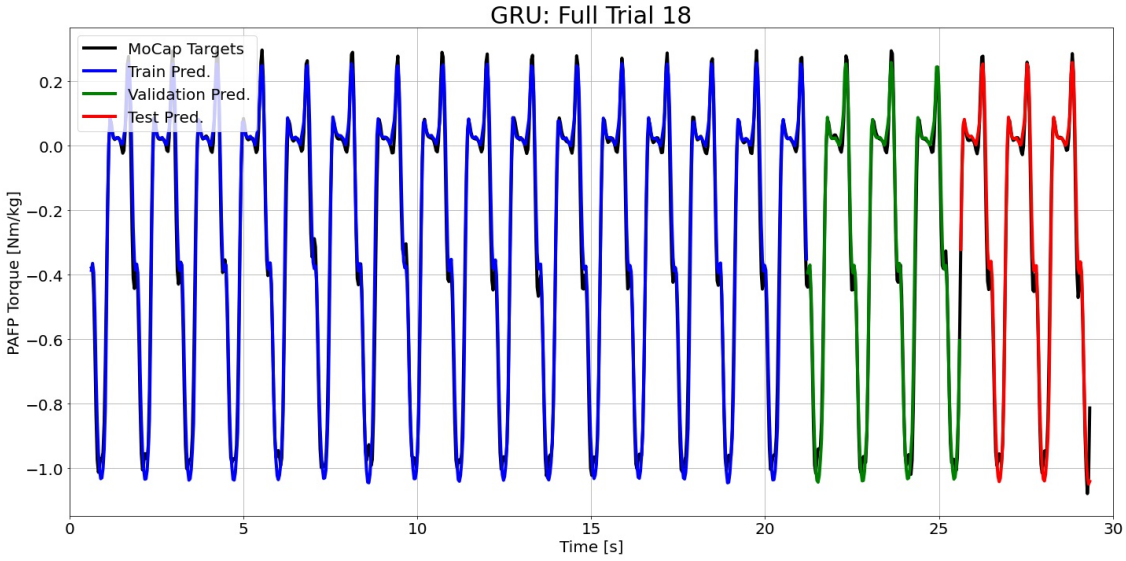


Figure I.84: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 18.

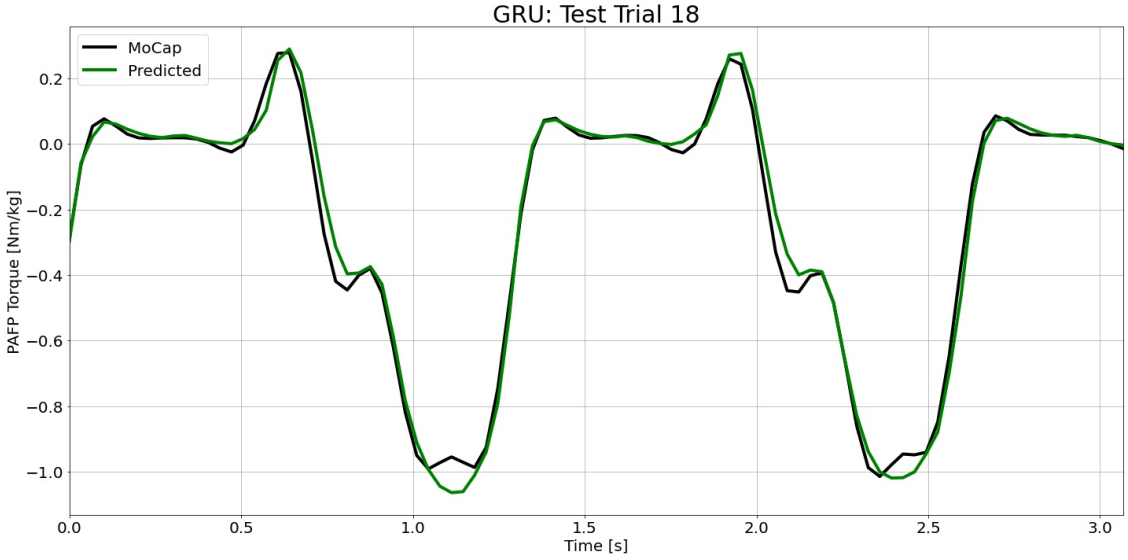


Figure I.85: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 18.

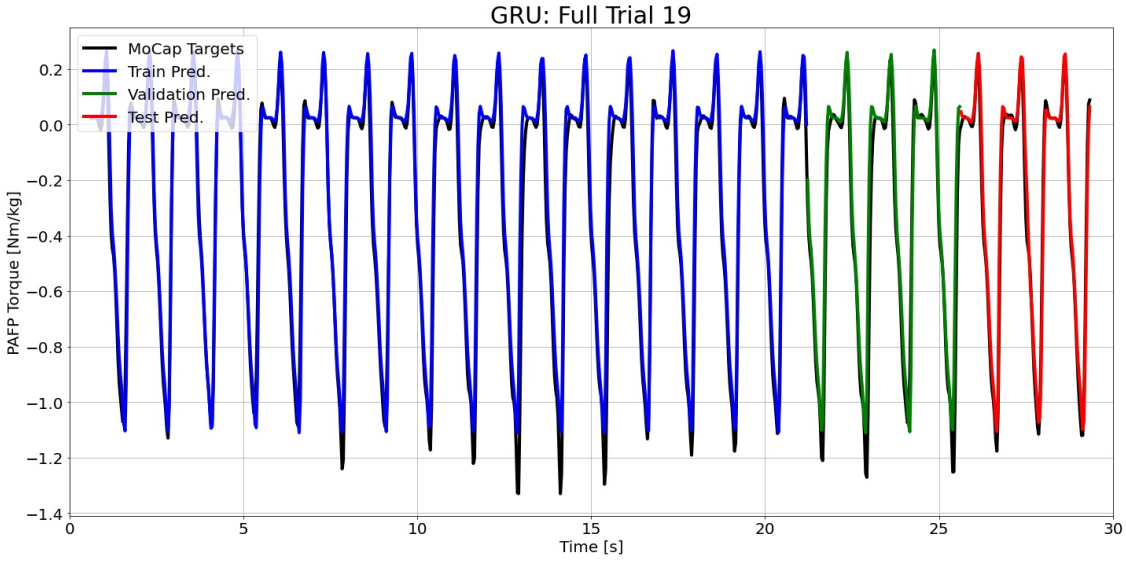


Figure I.86: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 19.

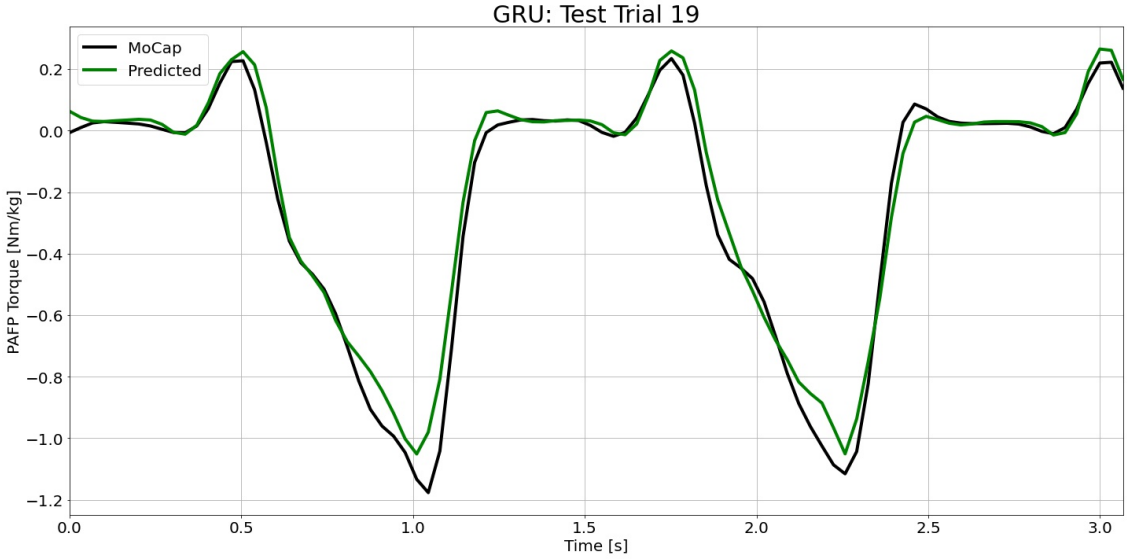


Figure I.87: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 19.

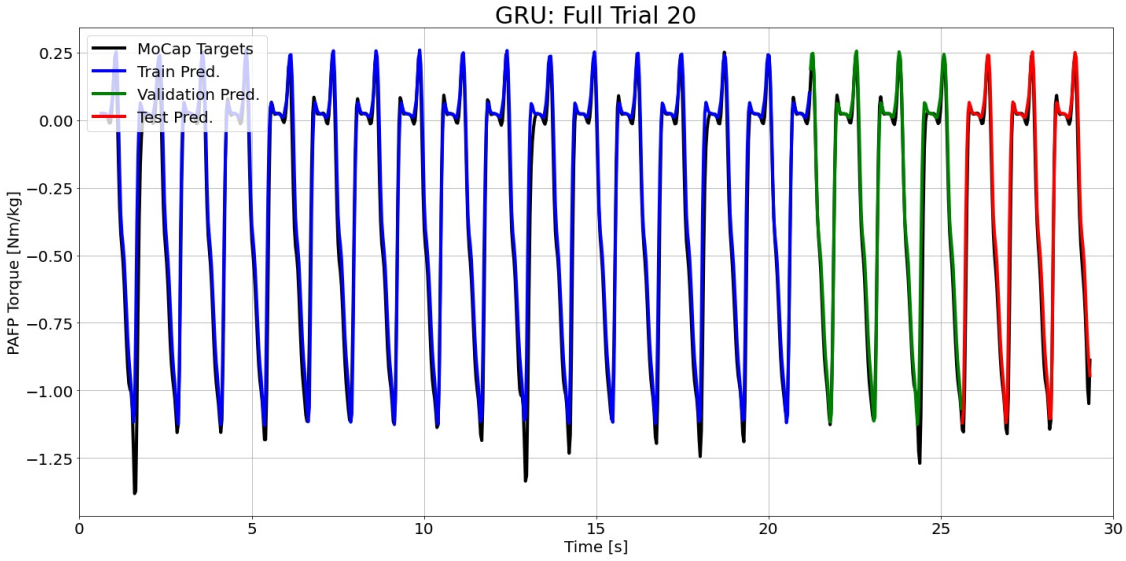


Figure I.88: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 20.

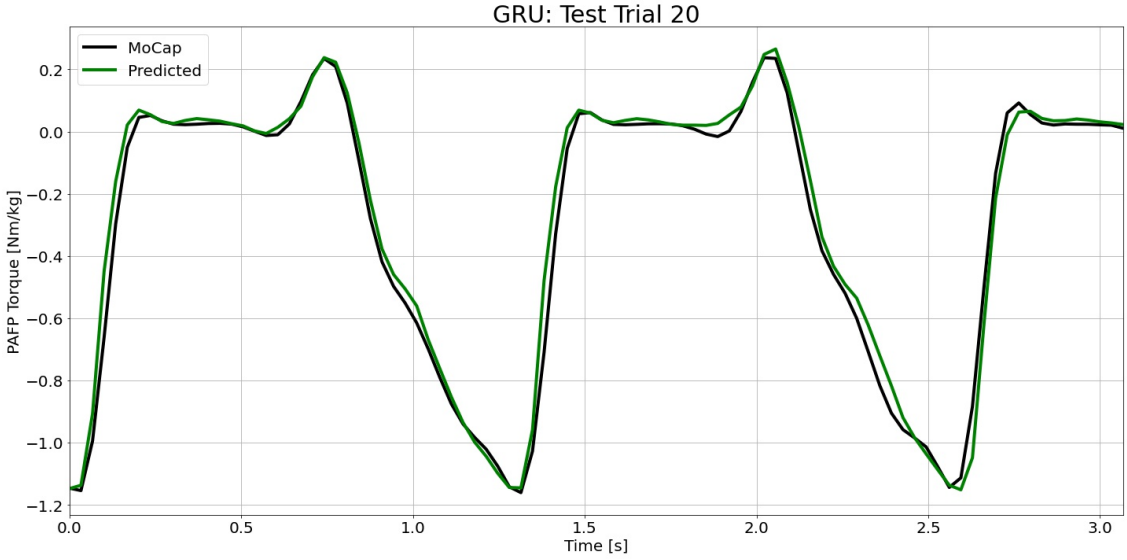


Figure I.89: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 20.

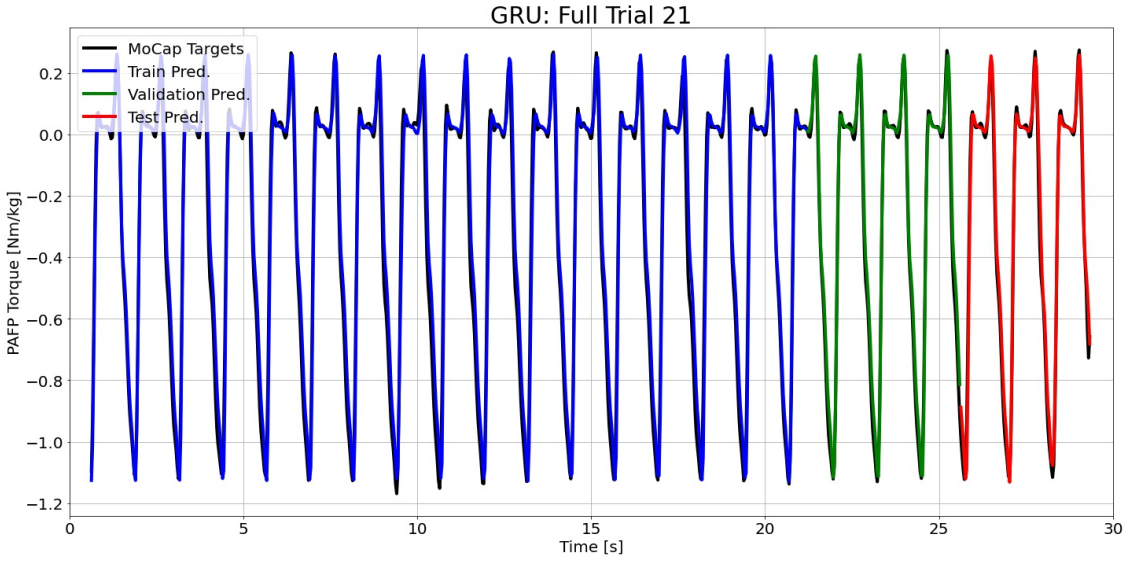


Figure I.90: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 21.

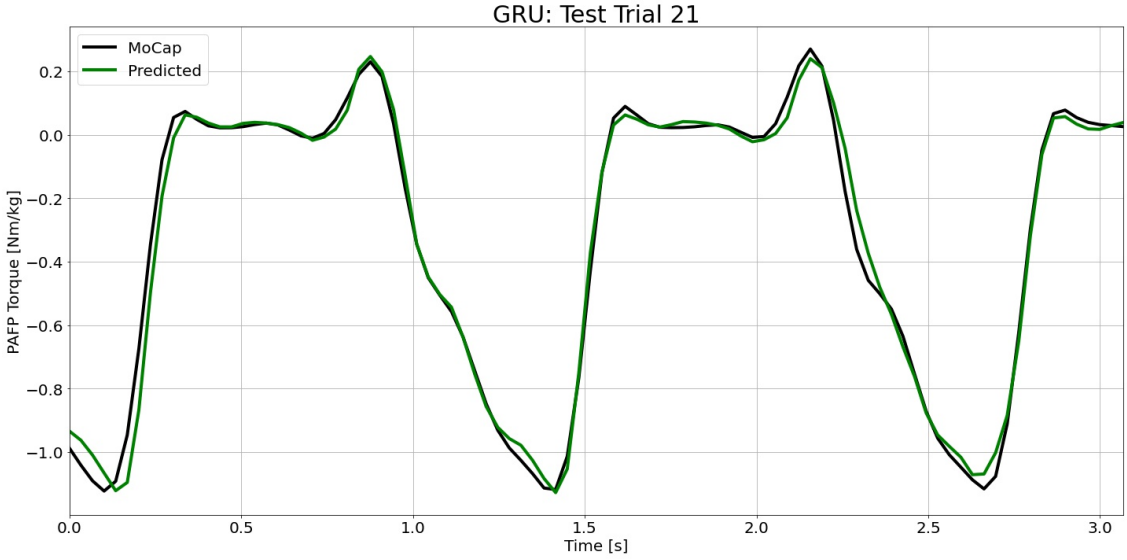


Figure I.91: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 21.

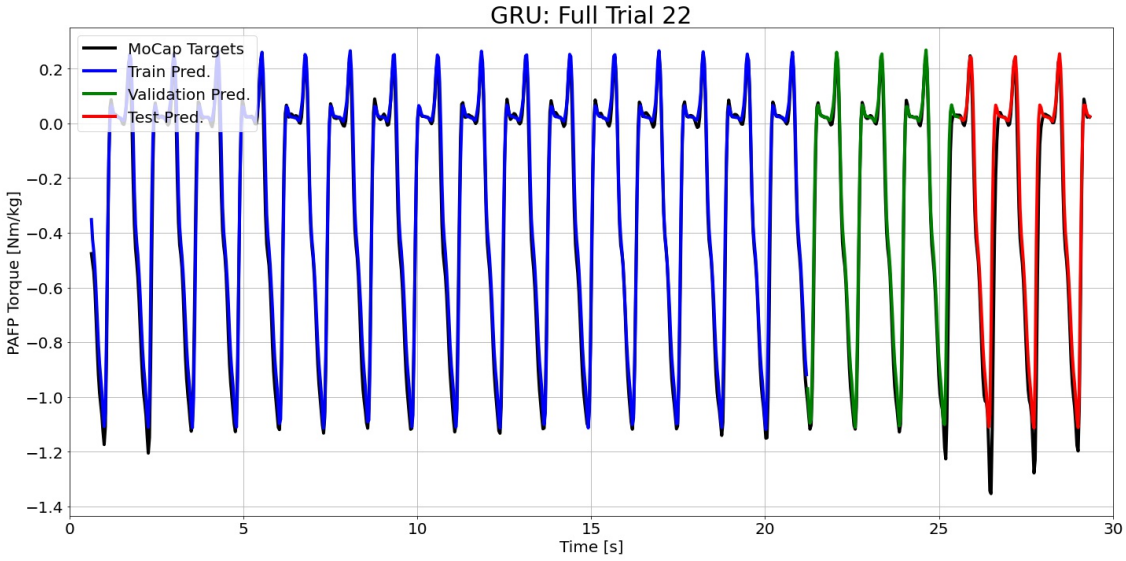


Figure I.92: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 22.

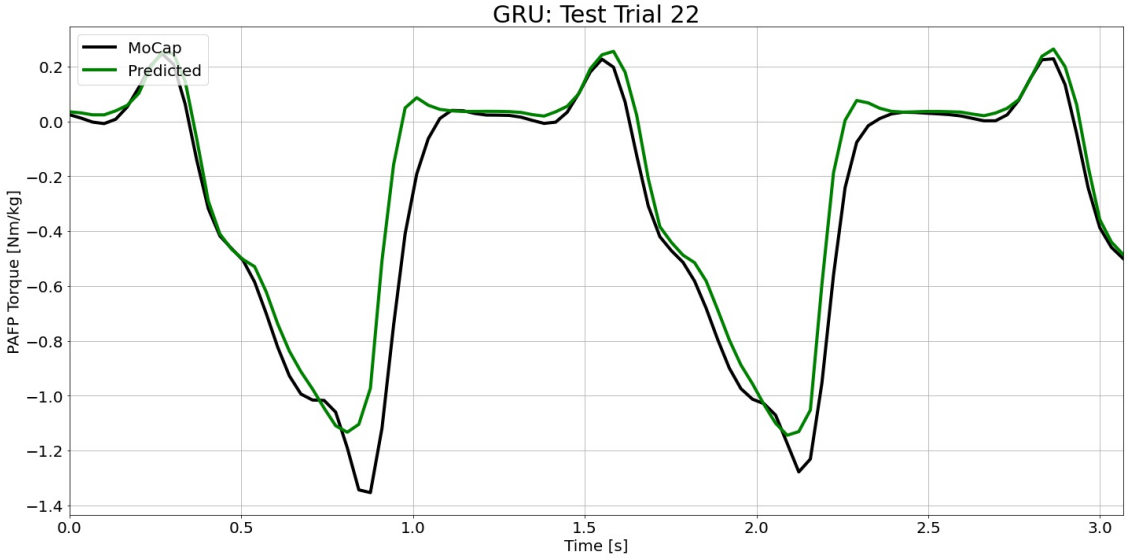


Figure I.93: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 22.

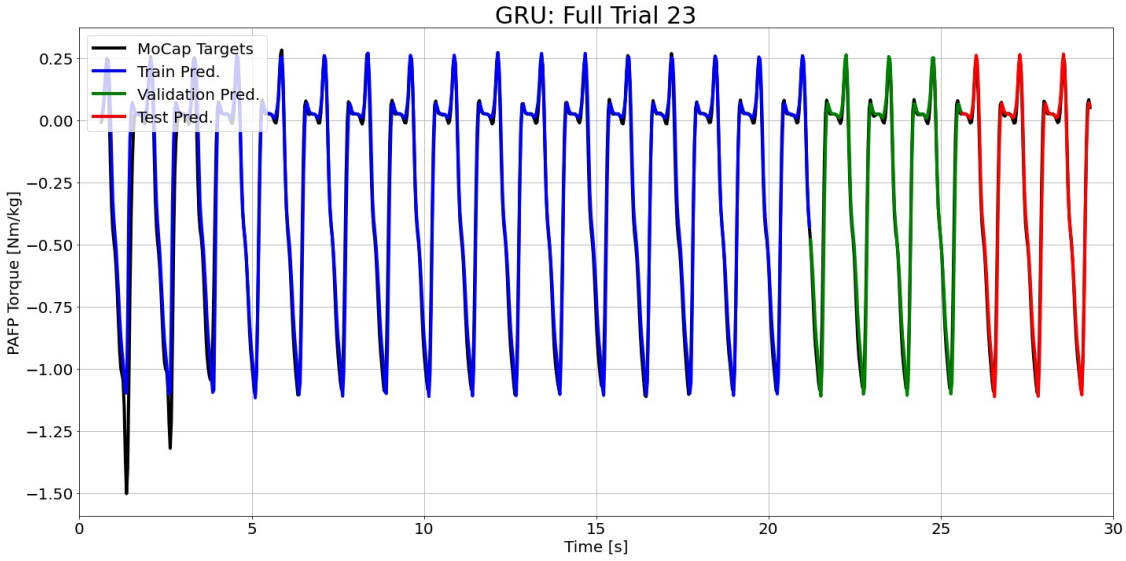


Figure I.94: GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 23.

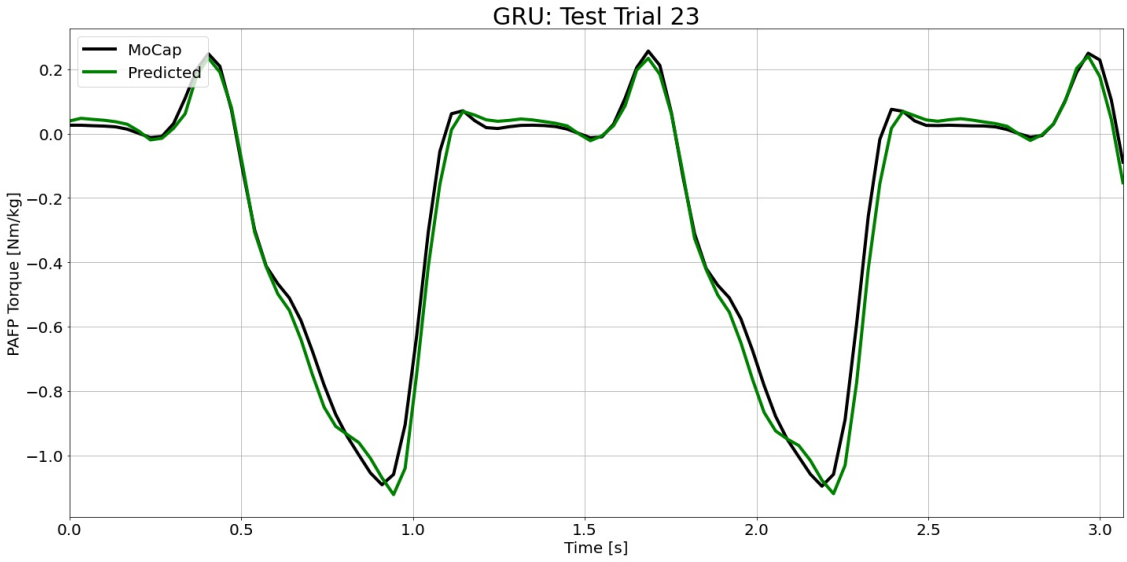


Figure I.95: GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 23.

I.3 Dual-Stage Attention-Based Gated Recurrent Unit Neural Network

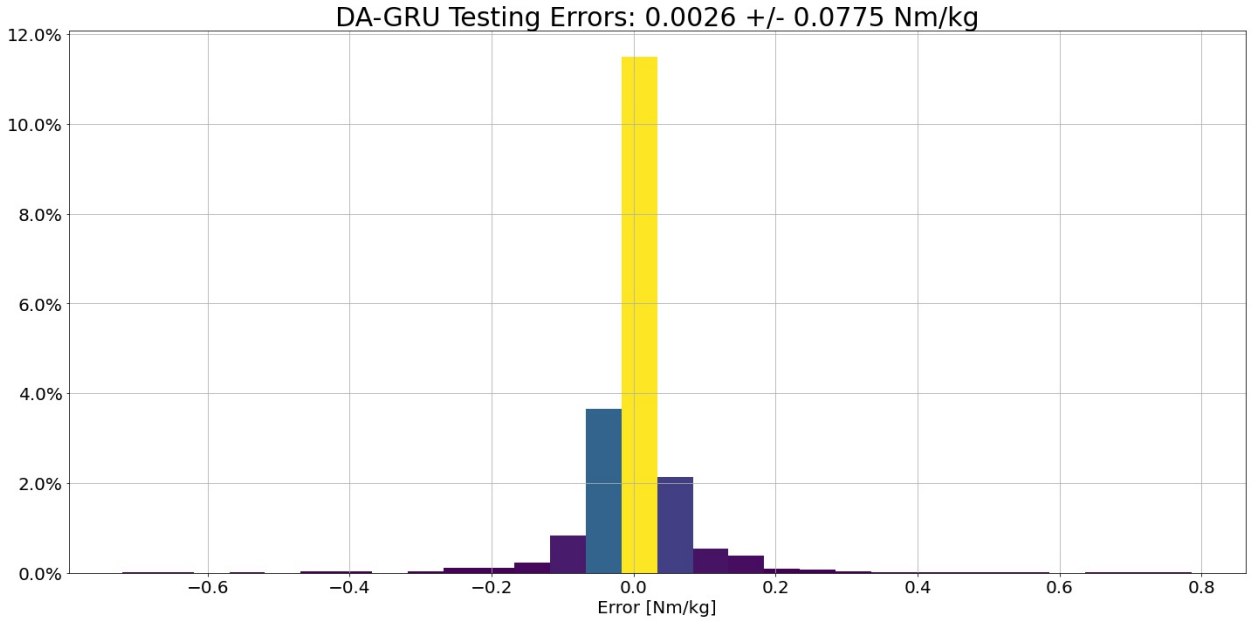


Figure I.96: DA-GRU prediction error histogram for all test time series data.

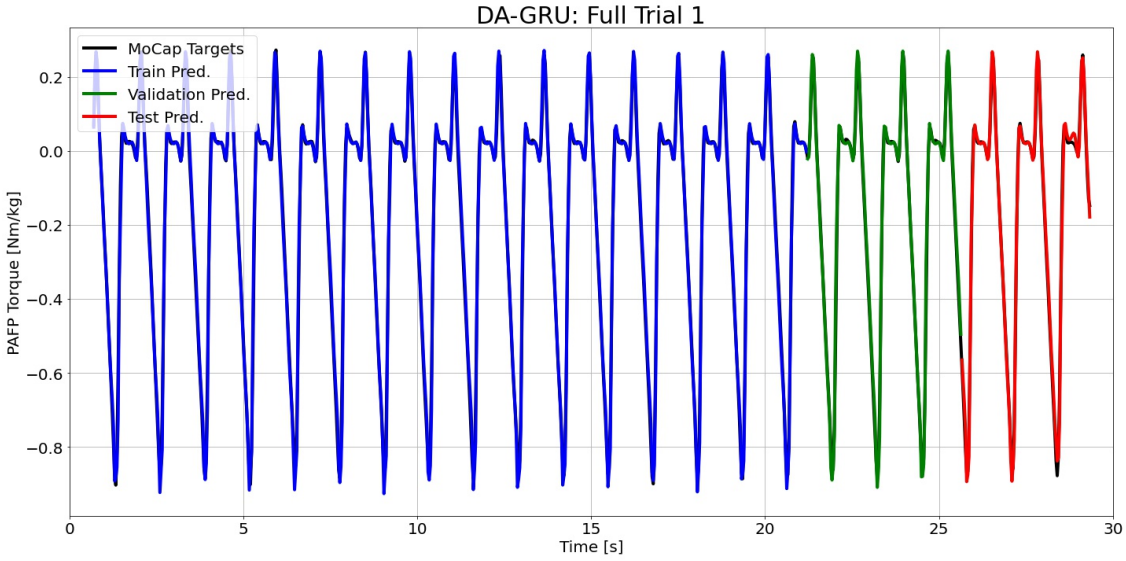


Figure I.97: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 1.

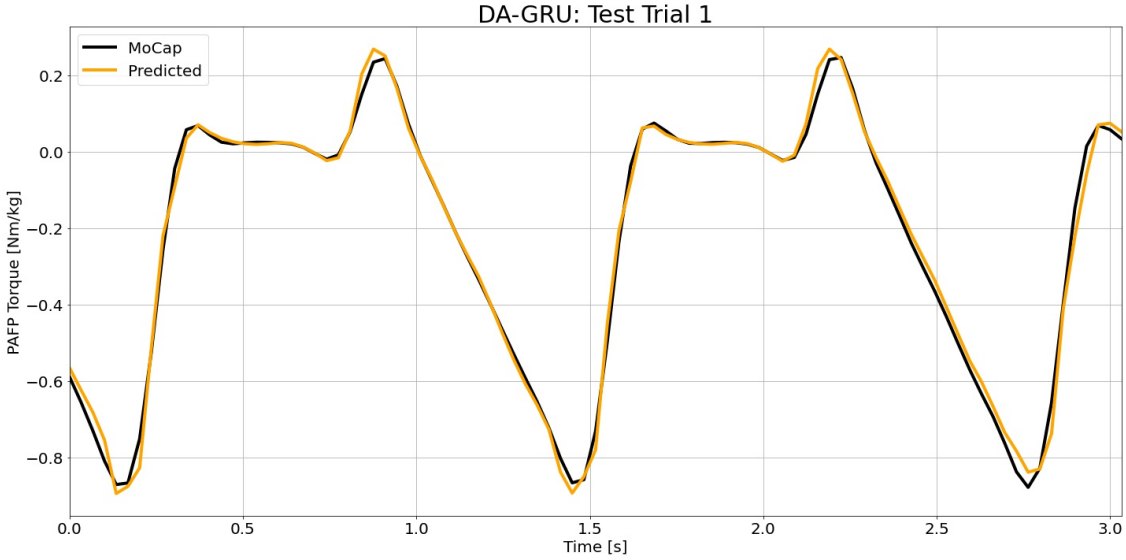


Figure I.98: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 1.

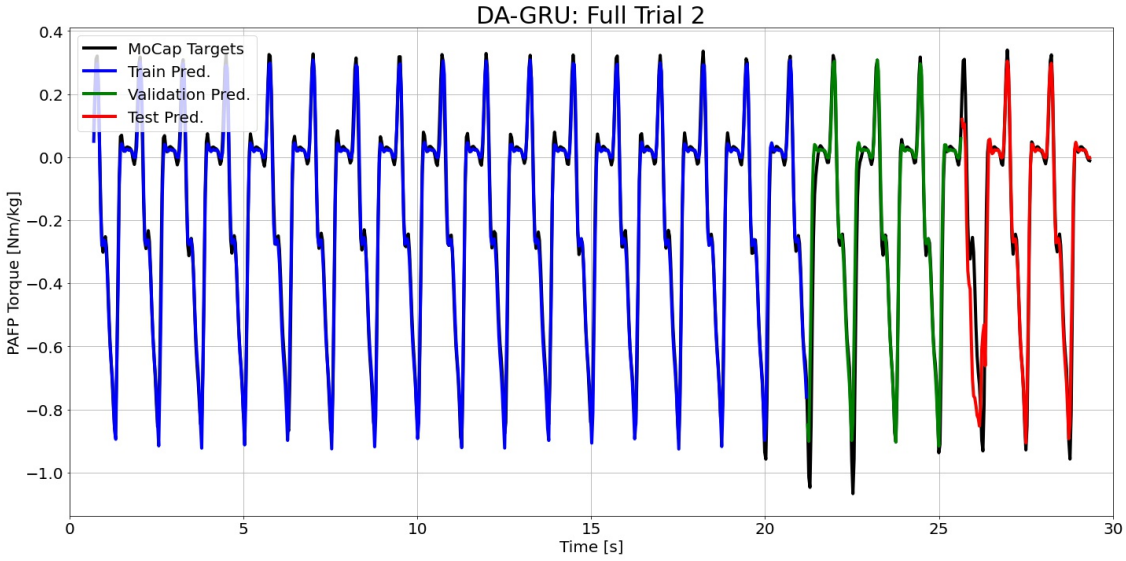


Figure I.99: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 2.

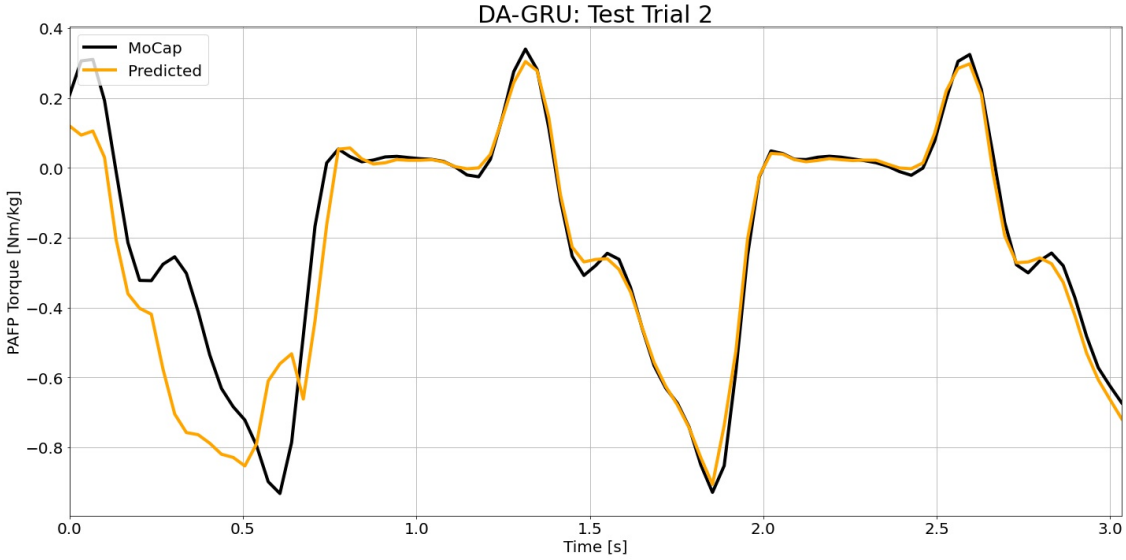


Figure I.100: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 2.

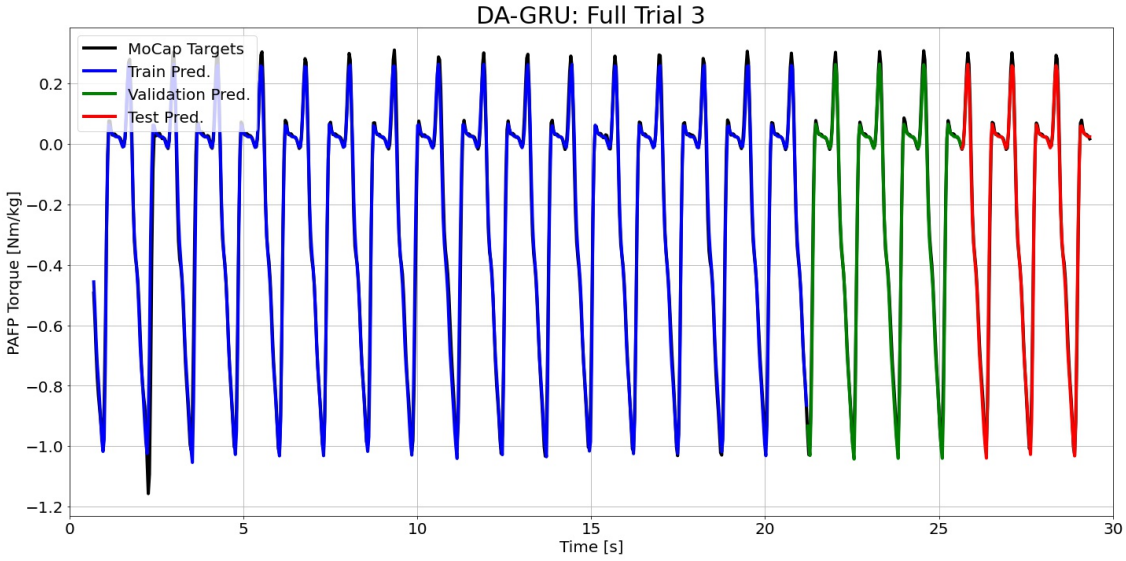


Figure I.101: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 3.

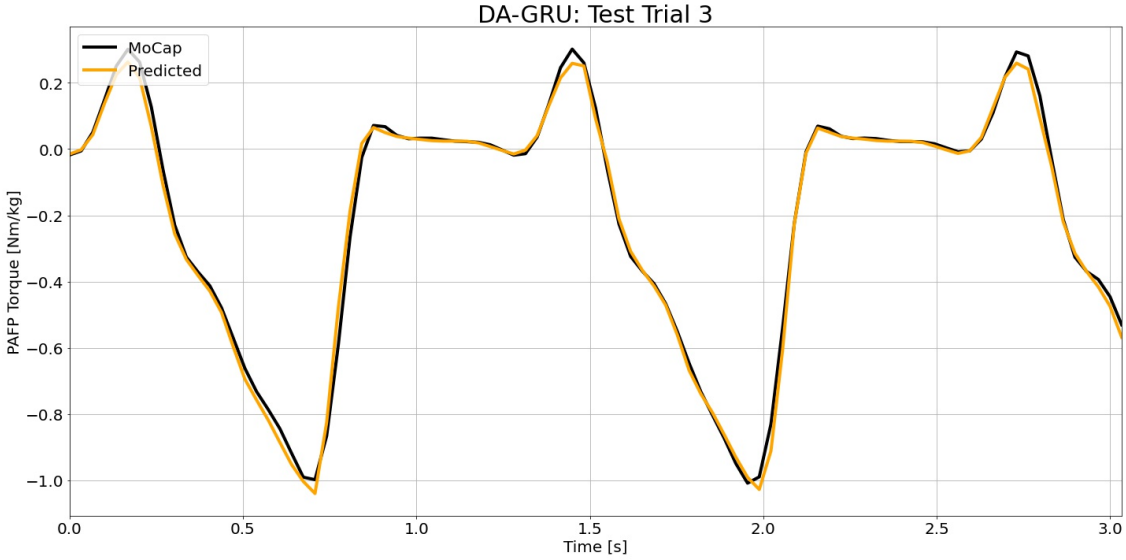


Figure I.102: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 3.

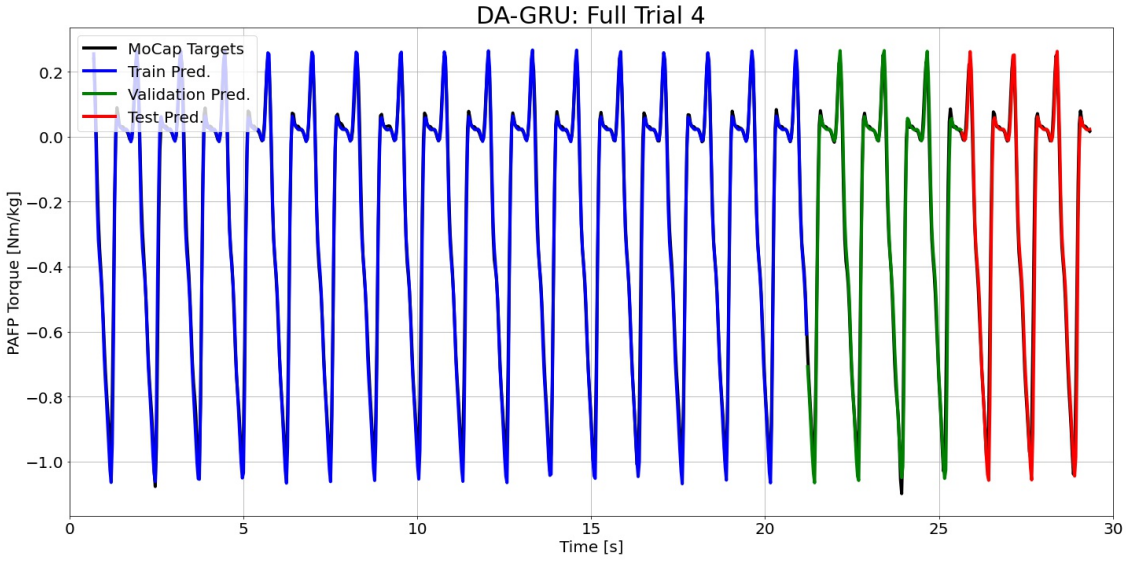


Figure I.103: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 4.

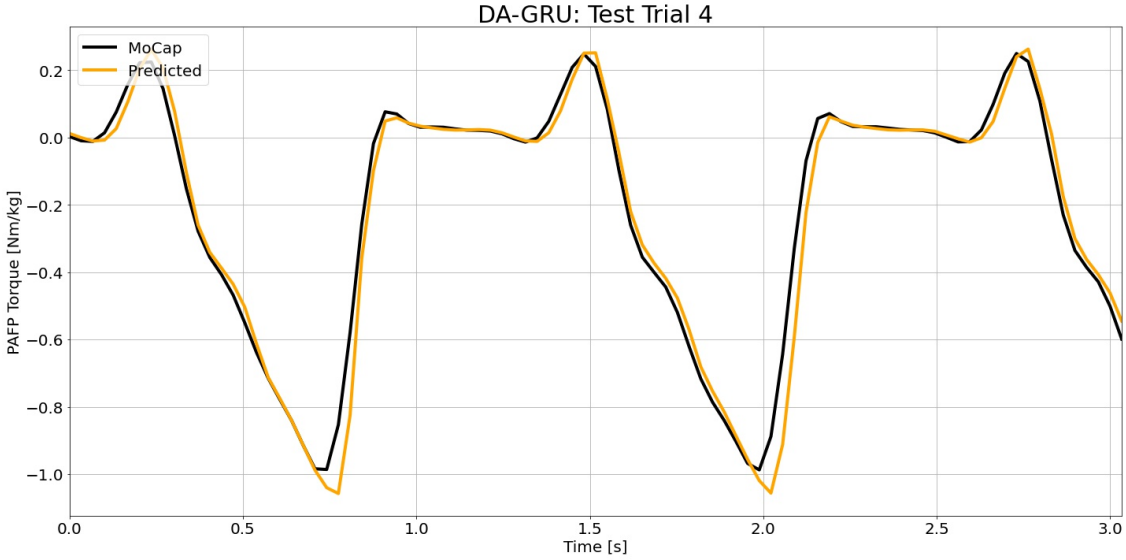


Figure I.104: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 4.

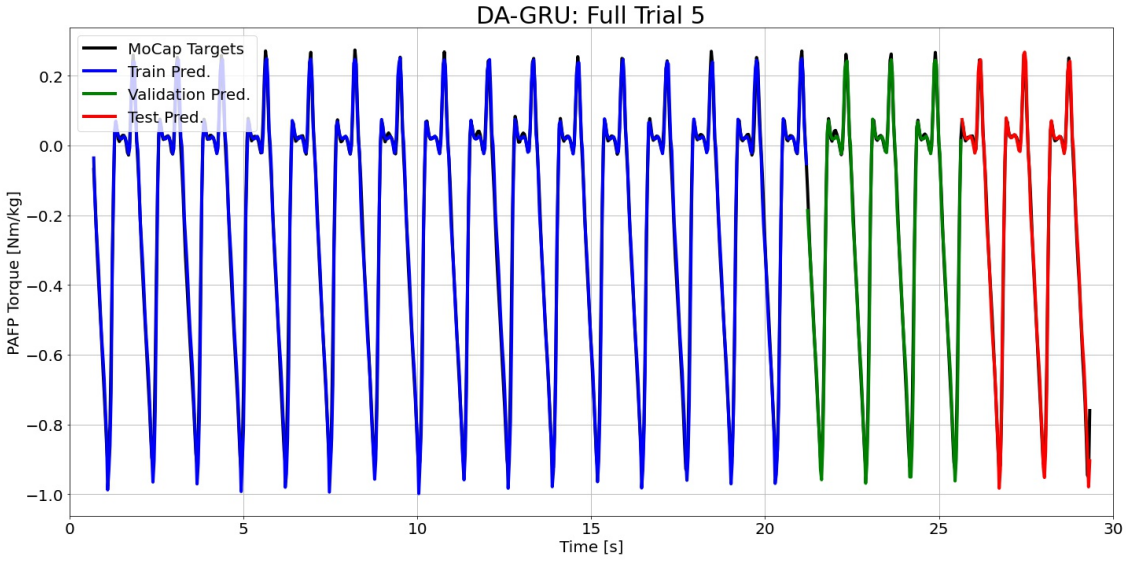


Figure I.105: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 5.

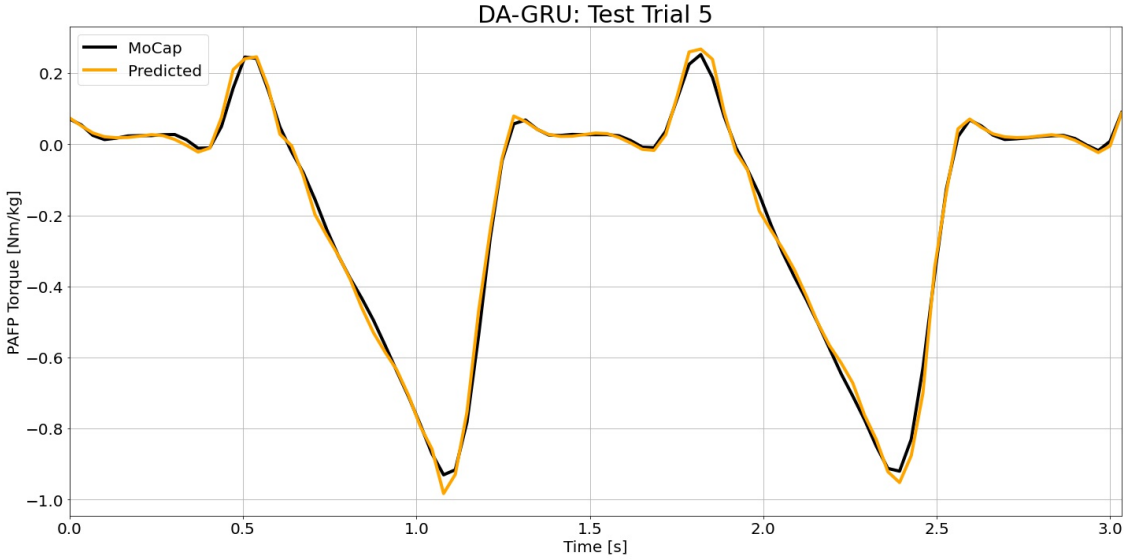


Figure I.106: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 5.

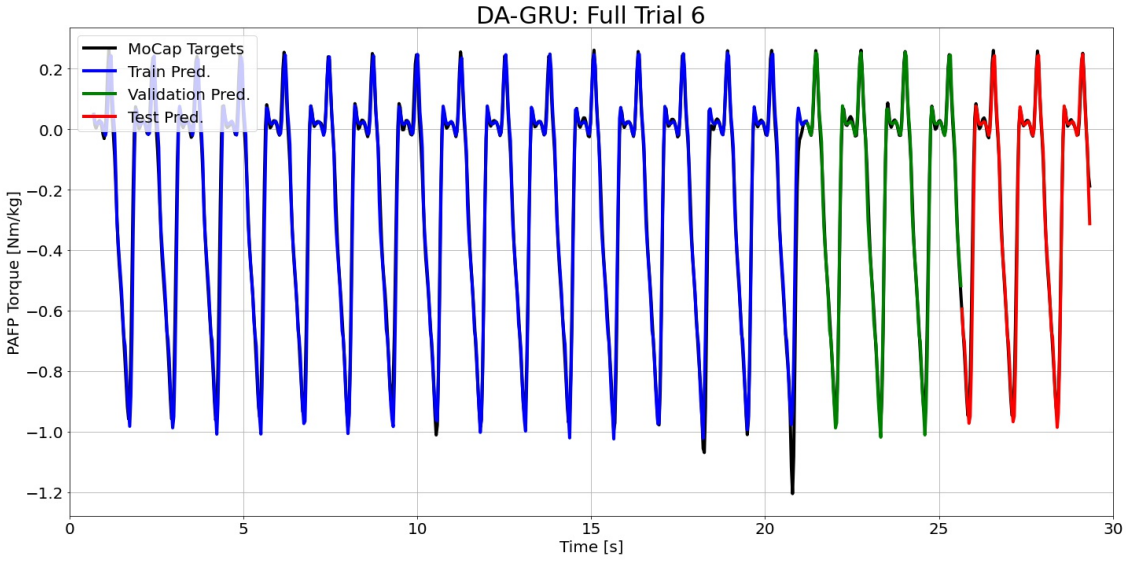


Figure I.107: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 6.

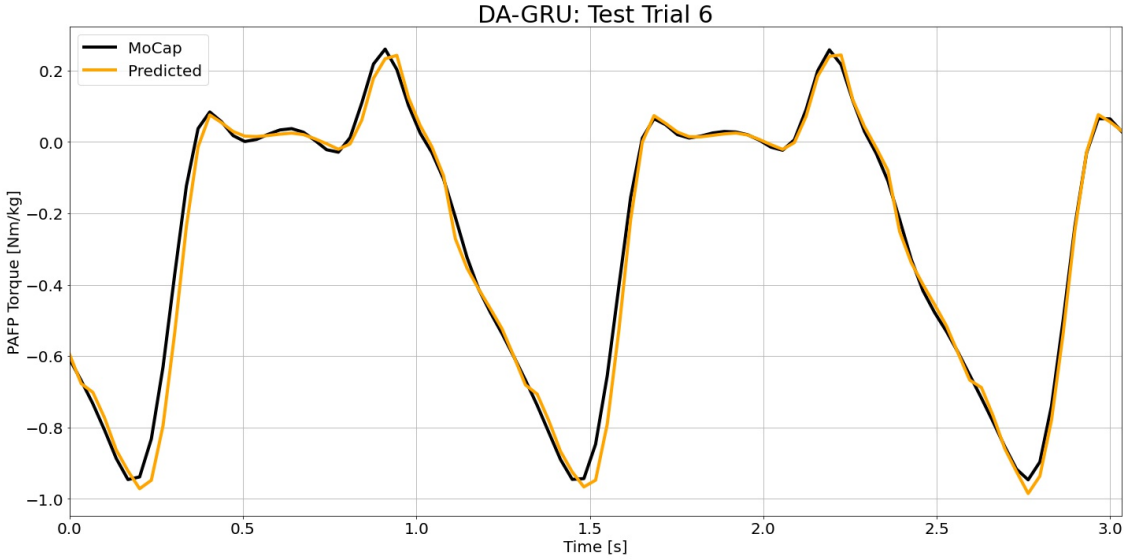


Figure I.108: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 6.

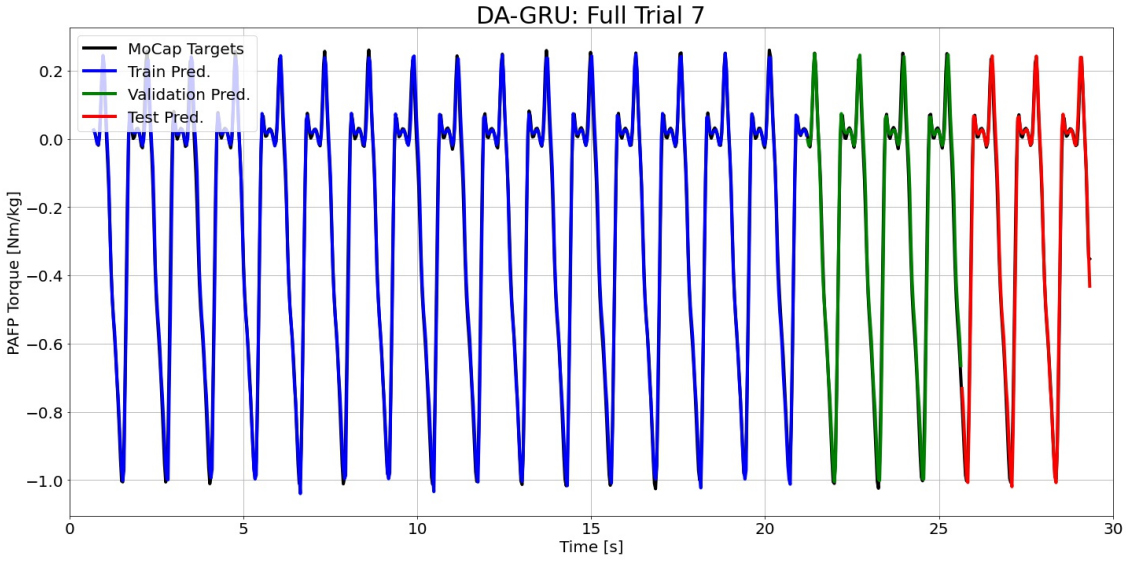


Figure I.109: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 7.

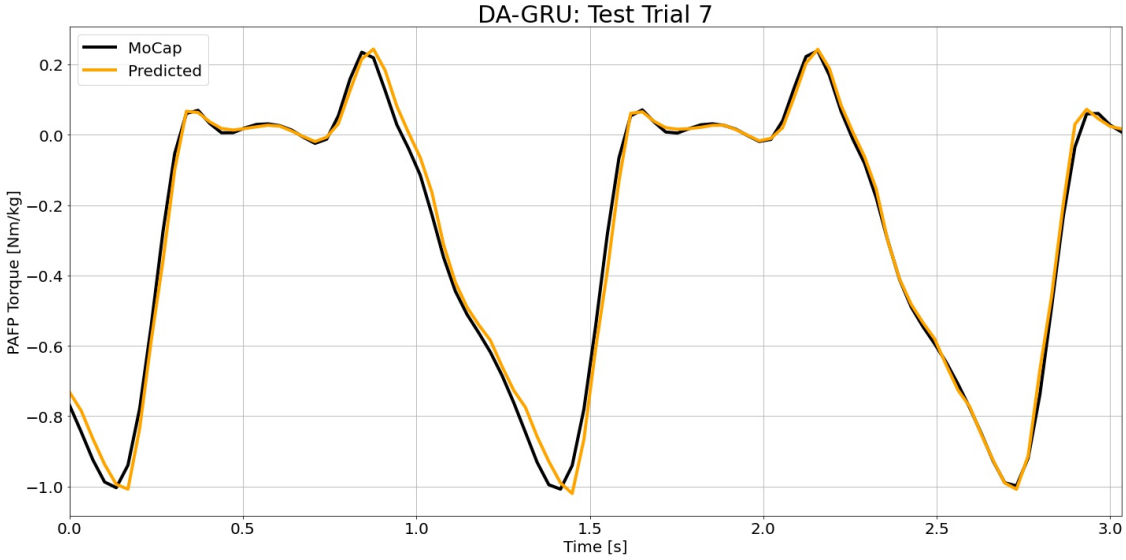


Figure I.110: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 7.

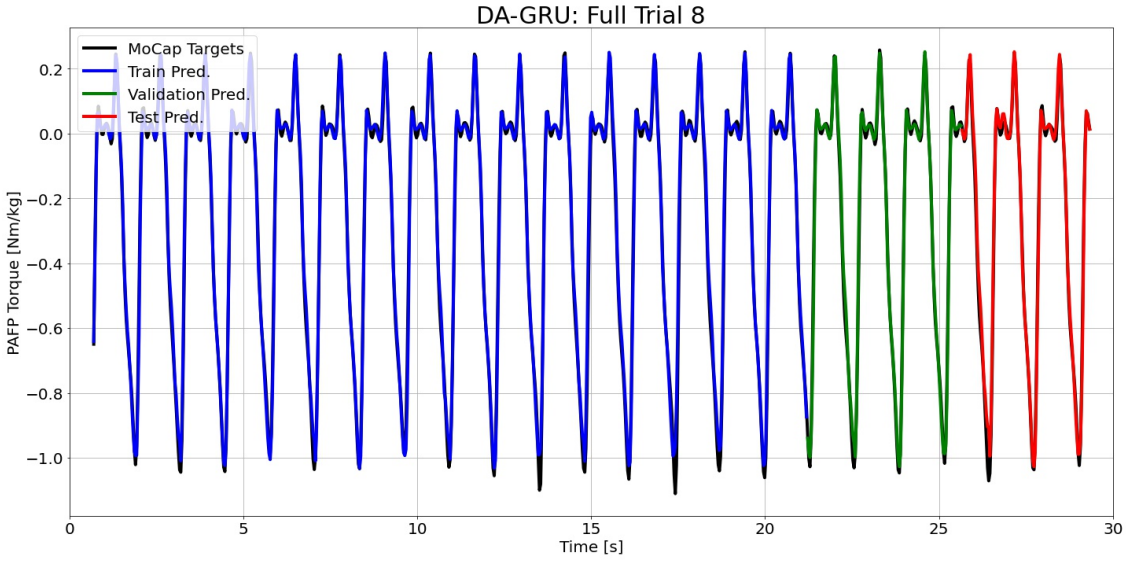


Figure I.111: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 8.

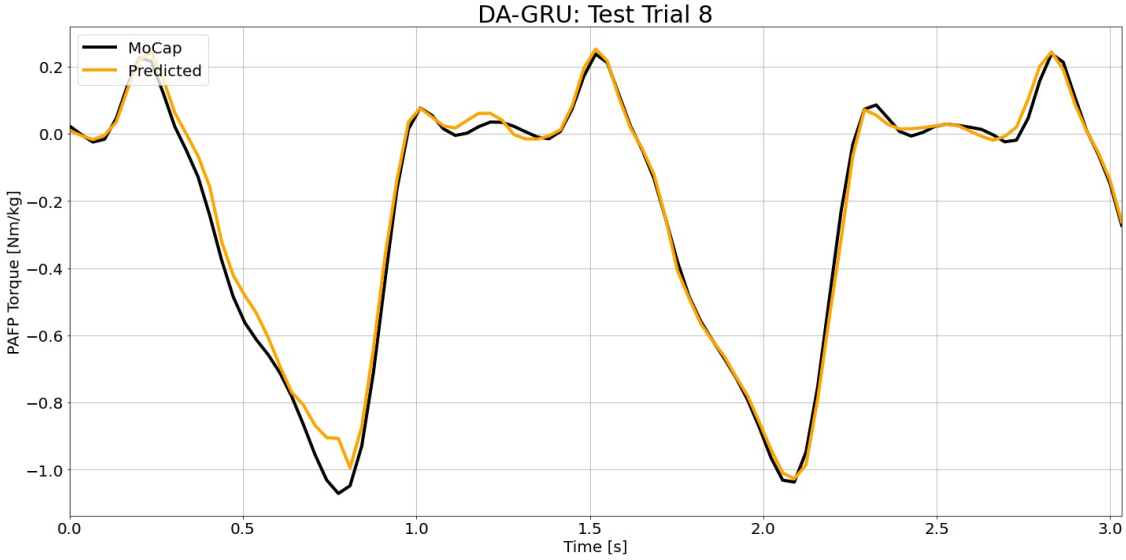


Figure I.112: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 8.

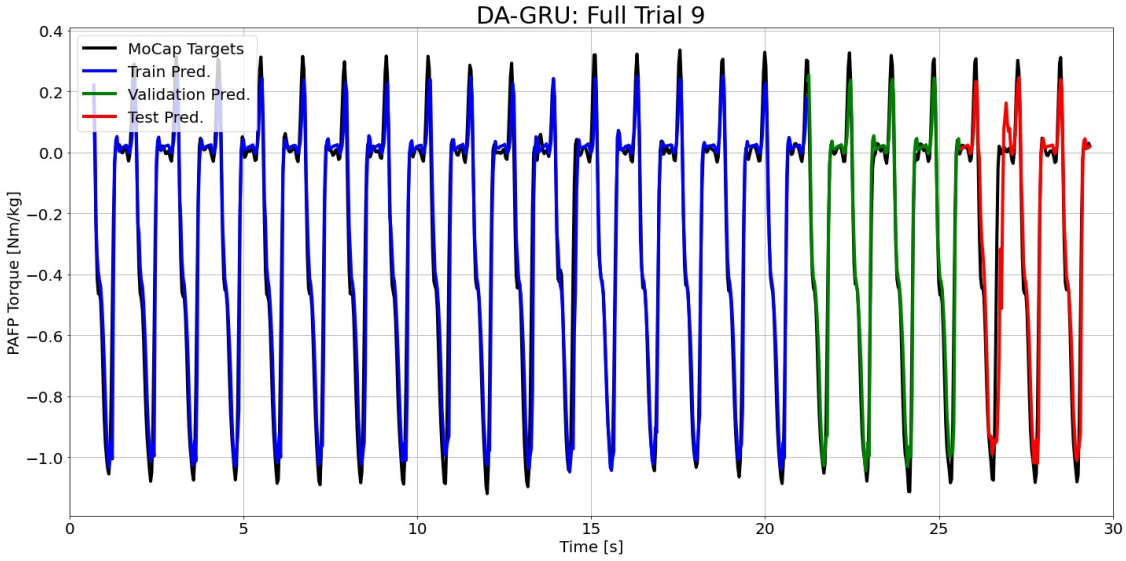


Figure I.113: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 9.

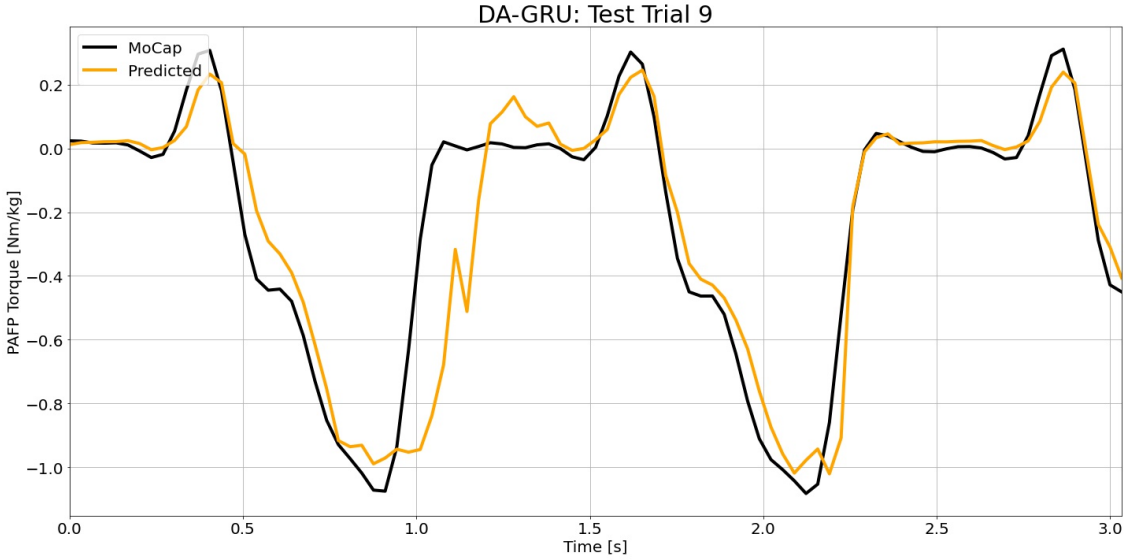


Figure I.114: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 9.

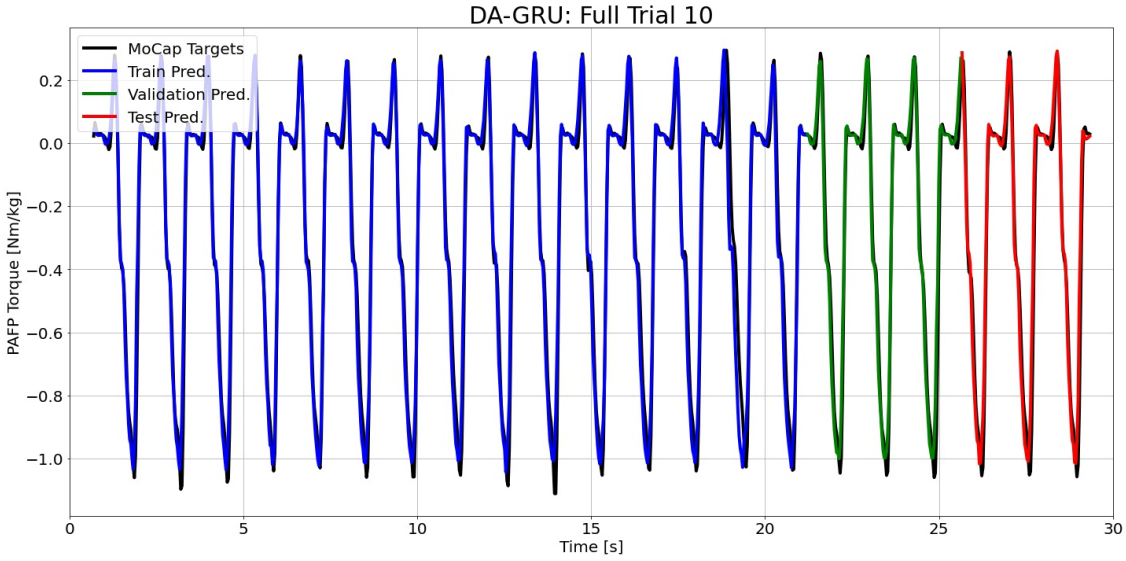


Figure I.115: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 10.

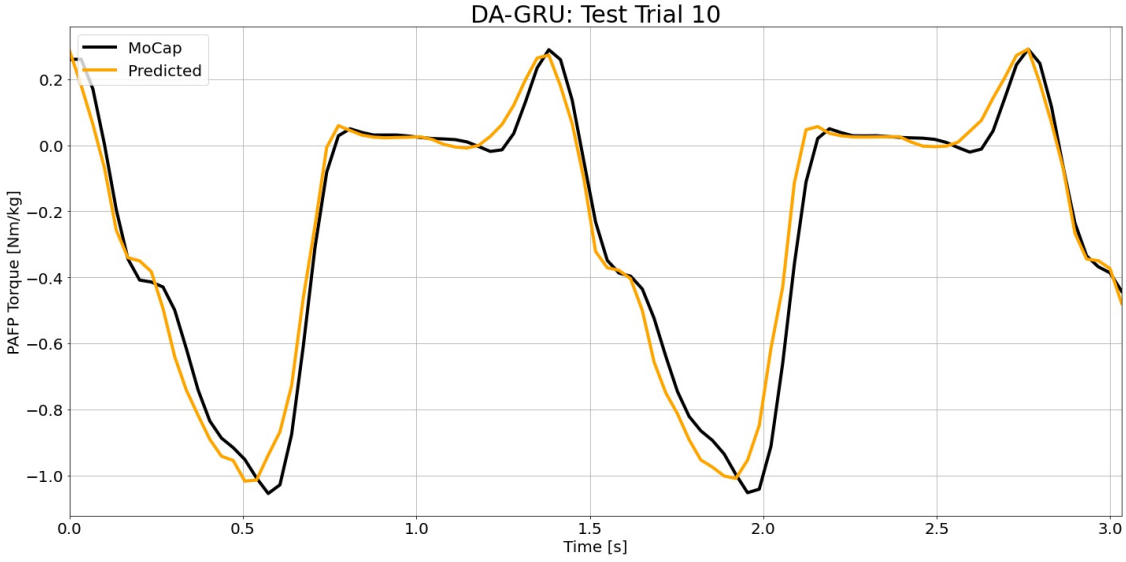


Figure I.116: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 10.

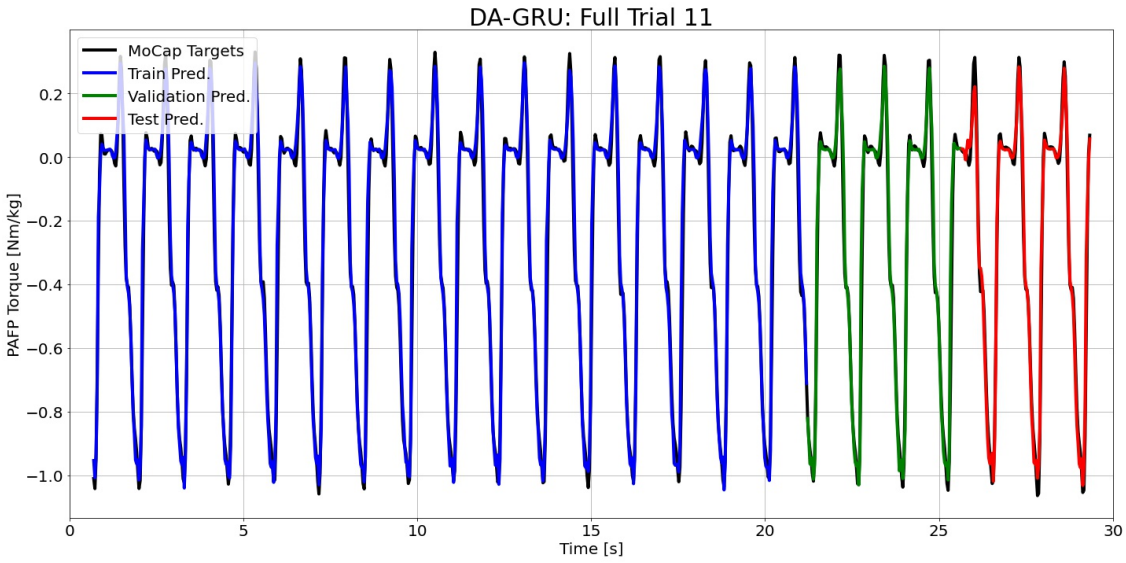


Figure I.117: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 11.

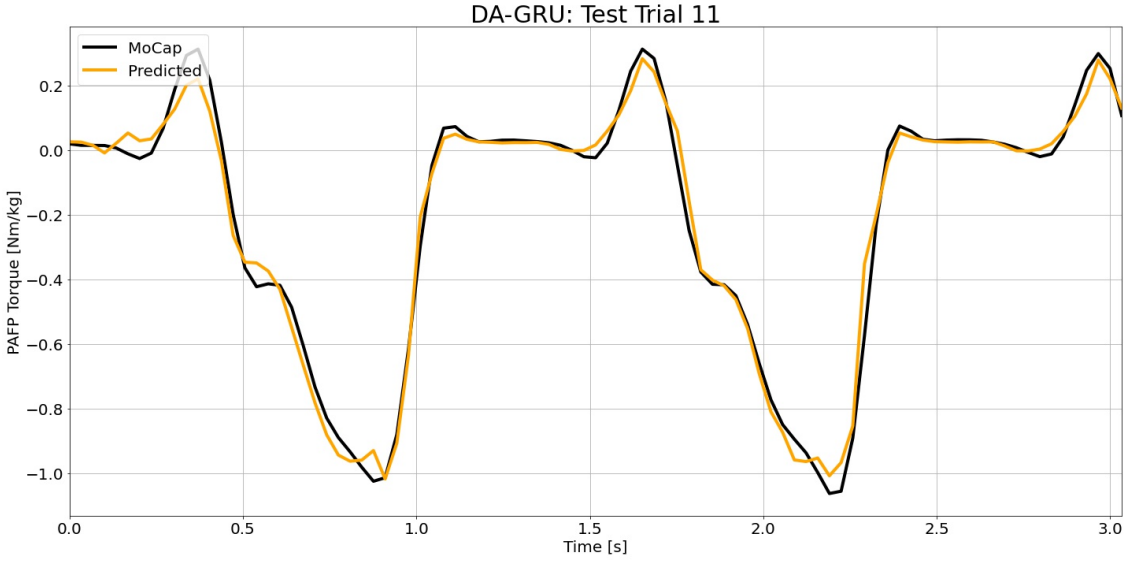


Figure I.118: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 11.

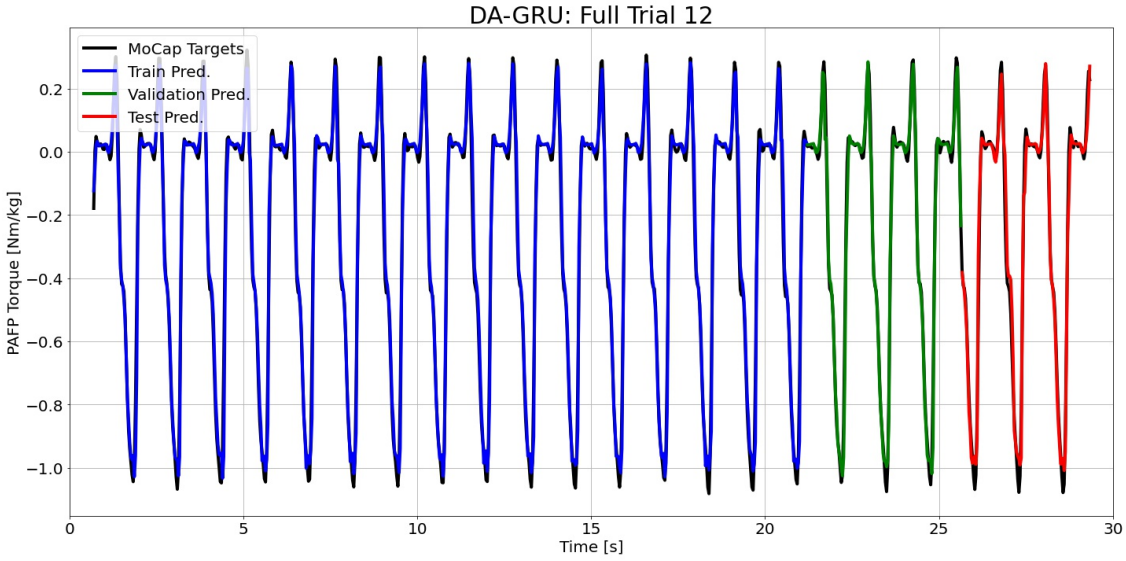


Figure I.119: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 12.

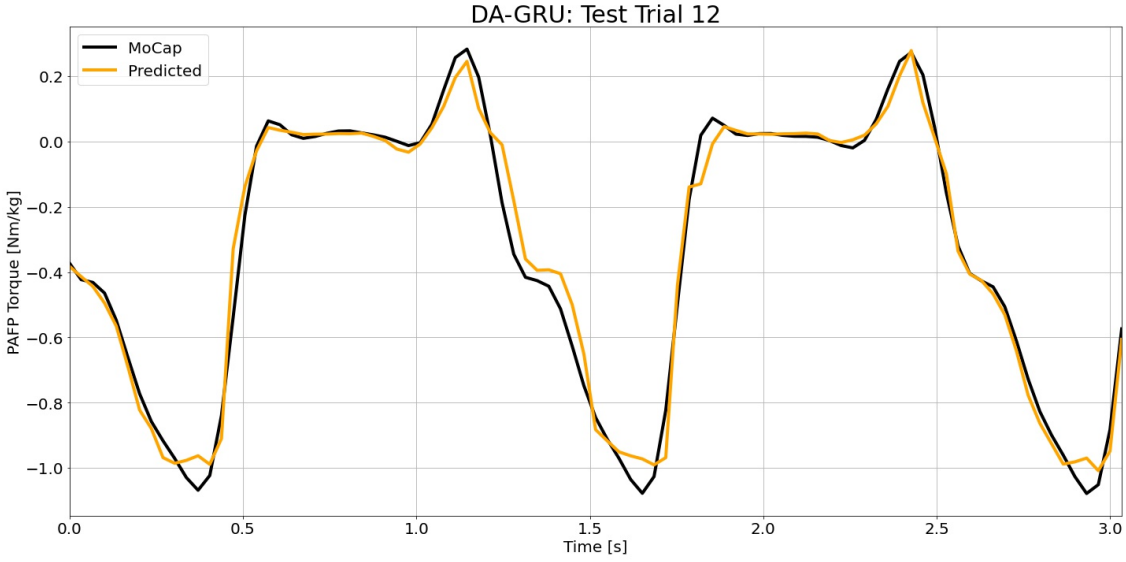


Figure I.120: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 12.

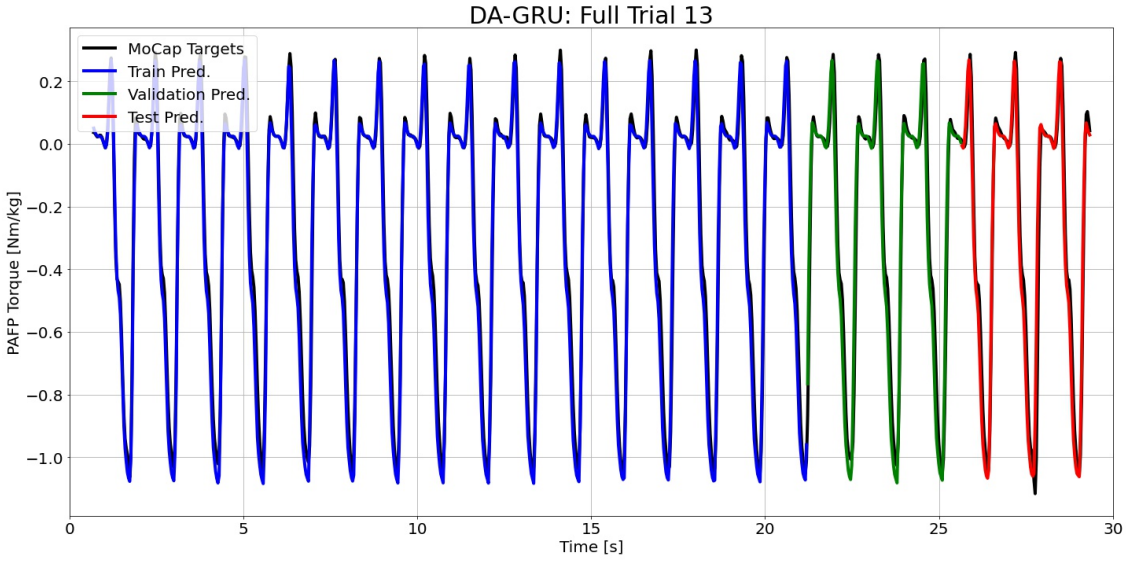


Figure I.121: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 13.

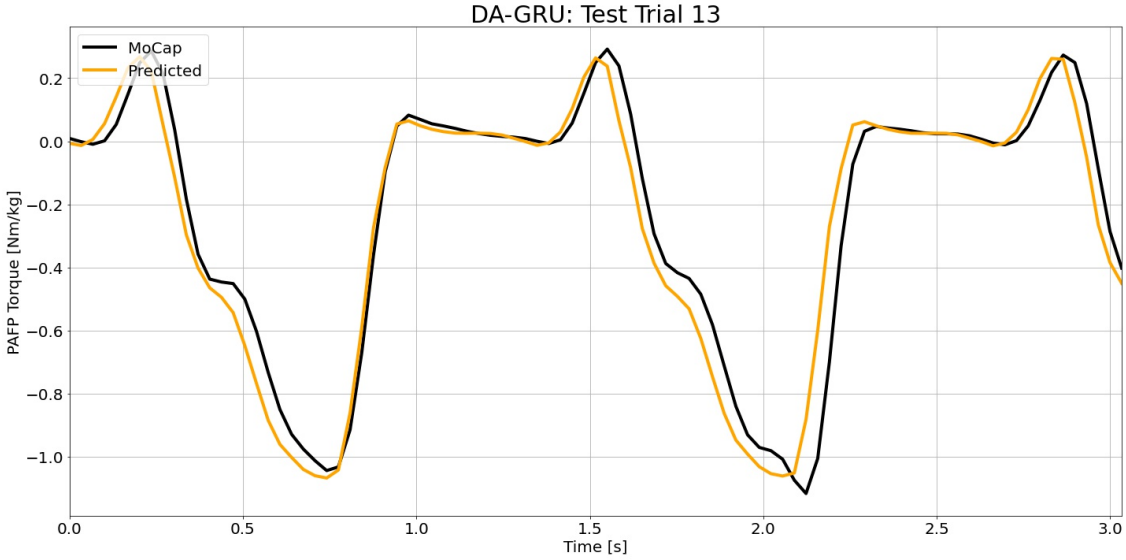


Figure I.122: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 13.

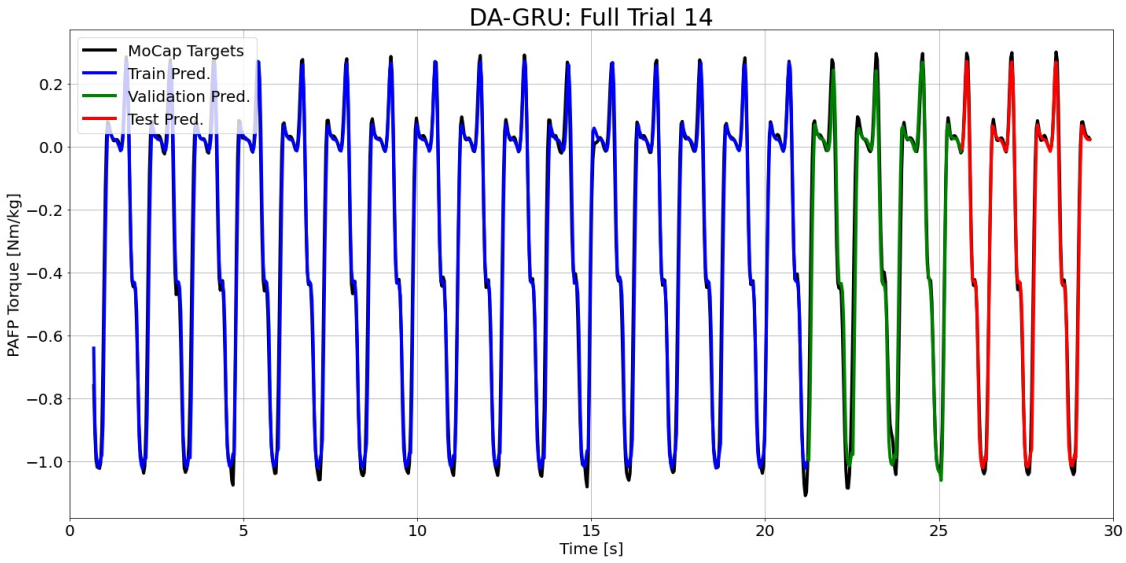


Figure I.123: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 14.

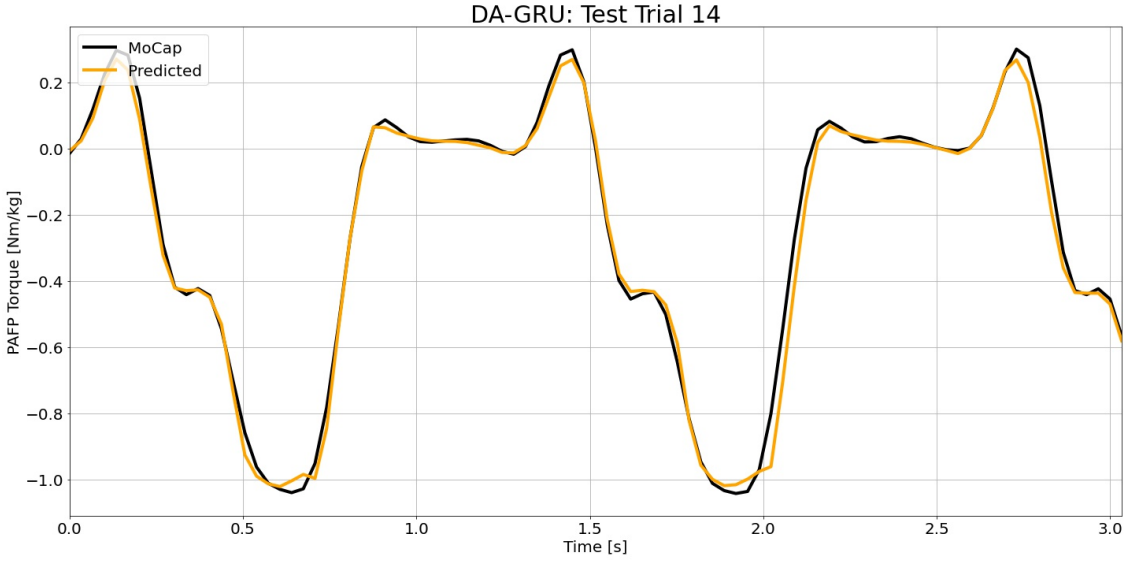


Figure I.124: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 14.

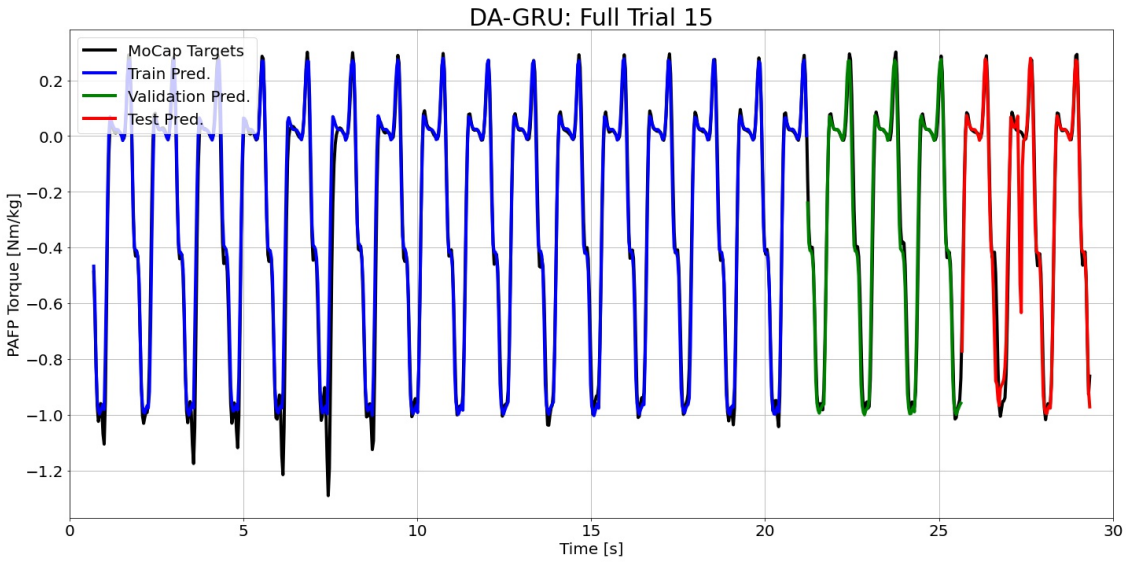


Figure I.125: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 15.

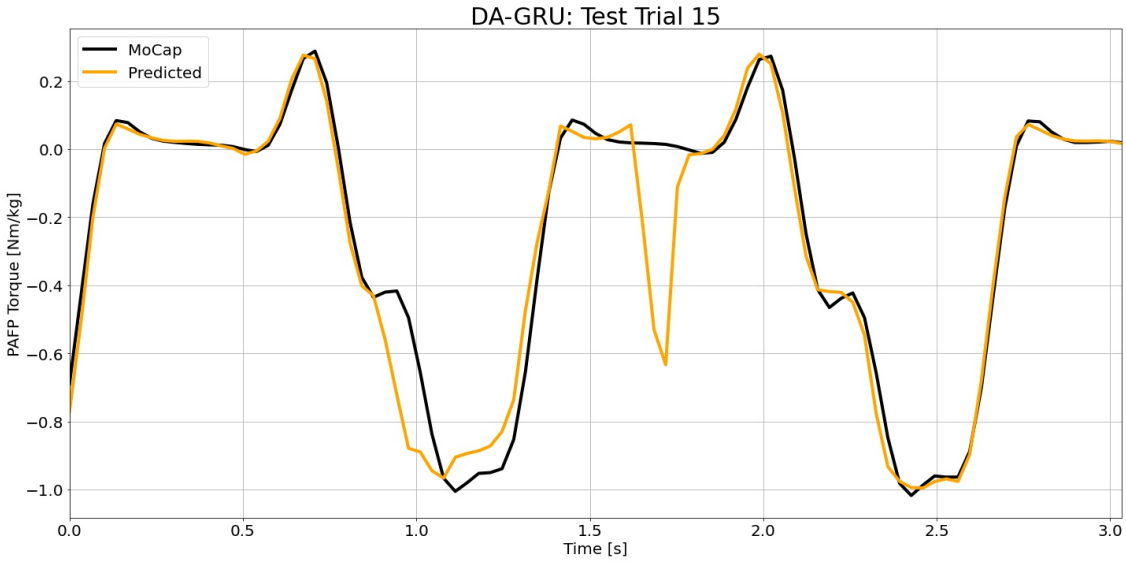


Figure I.126: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 15.

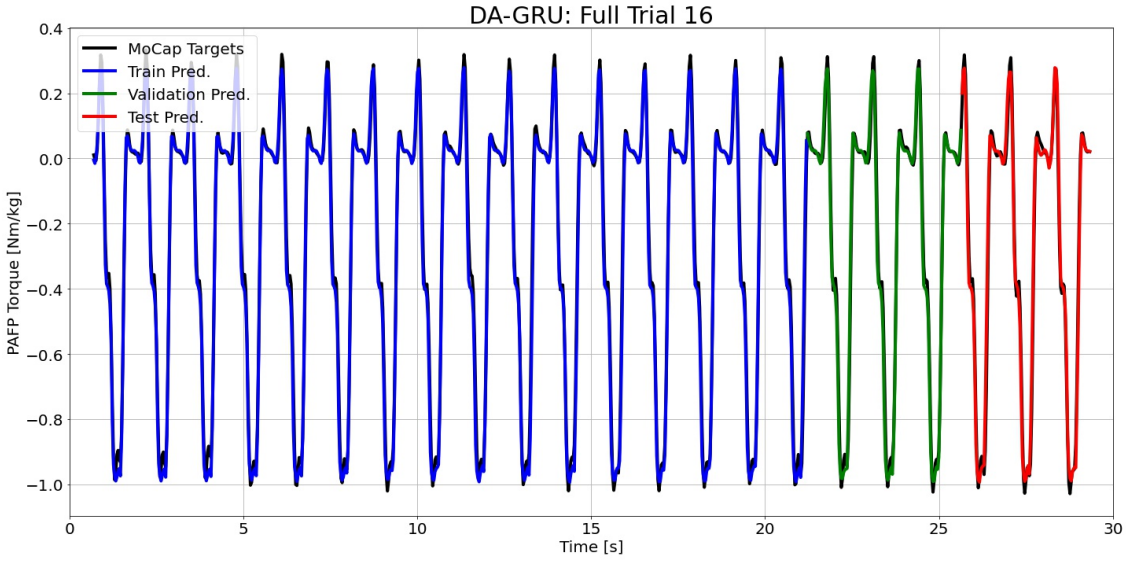


Figure I.127: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 16.

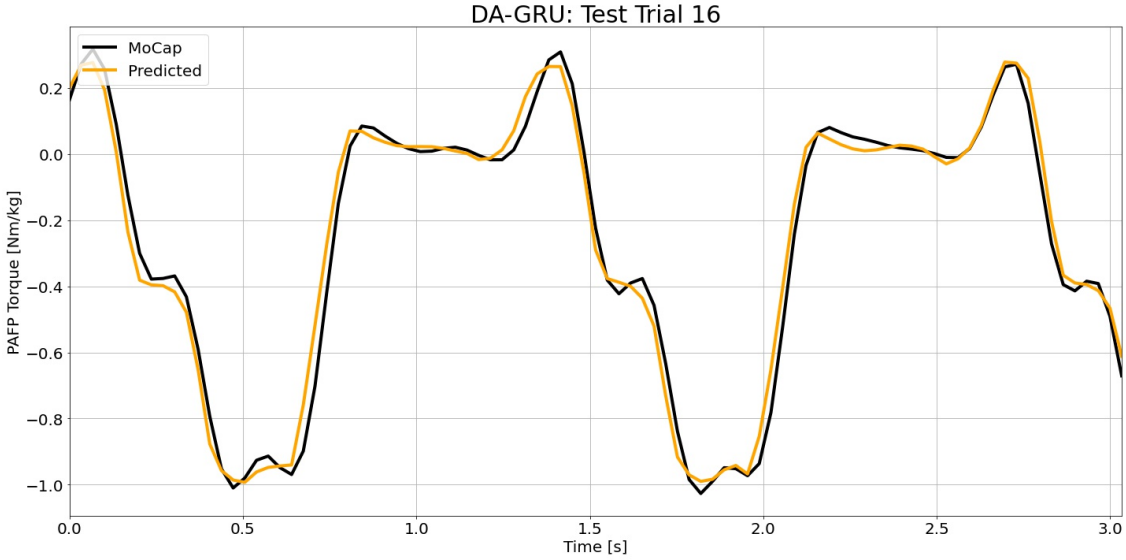


Figure I.128: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 16.

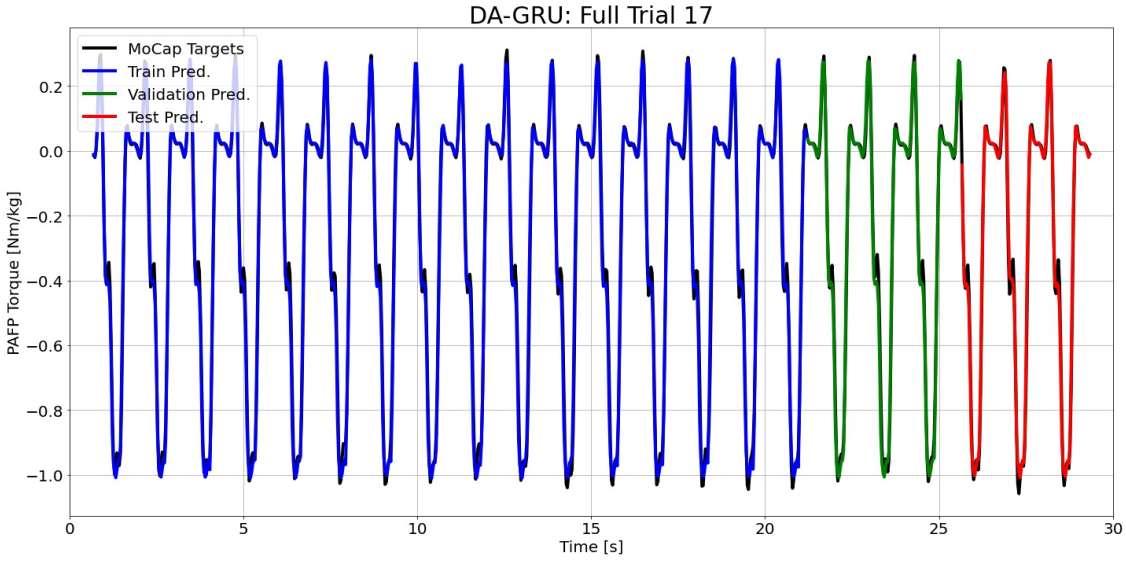


Figure I.129: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 17.

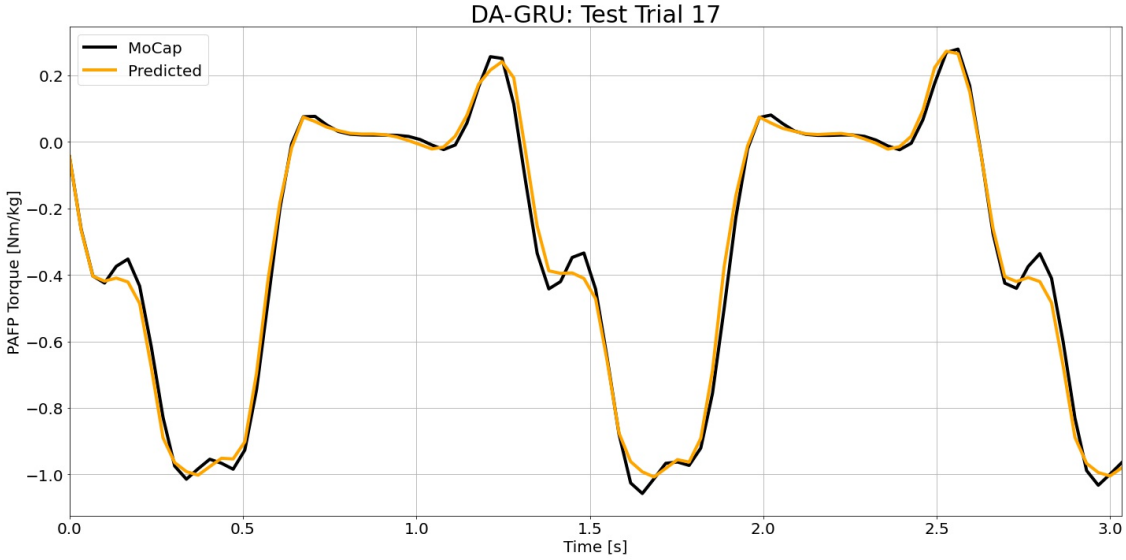


Figure I.130: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 17.

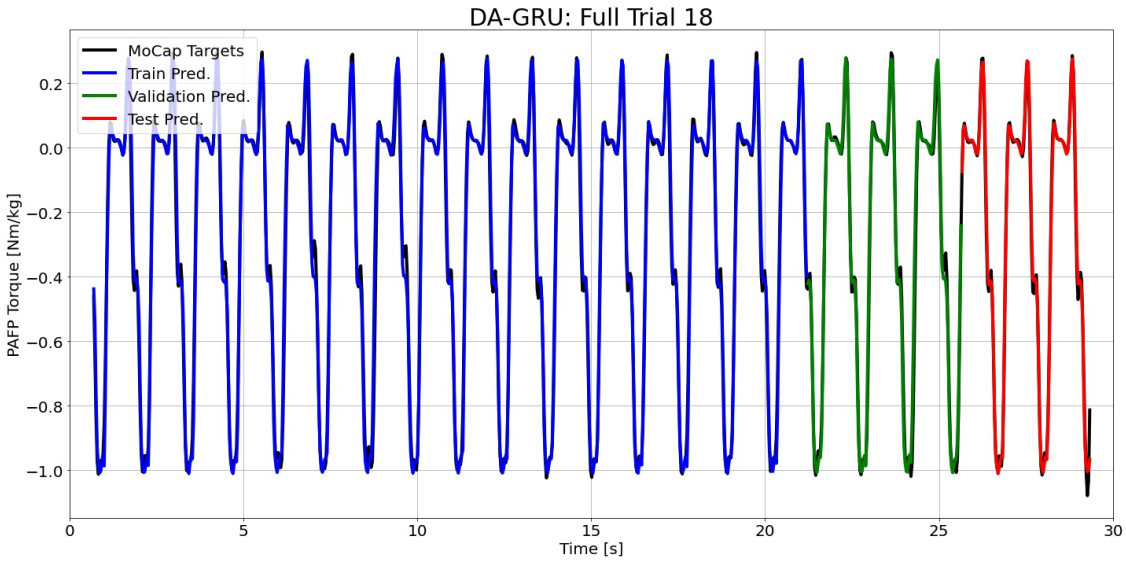


Figure I.131: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 18.

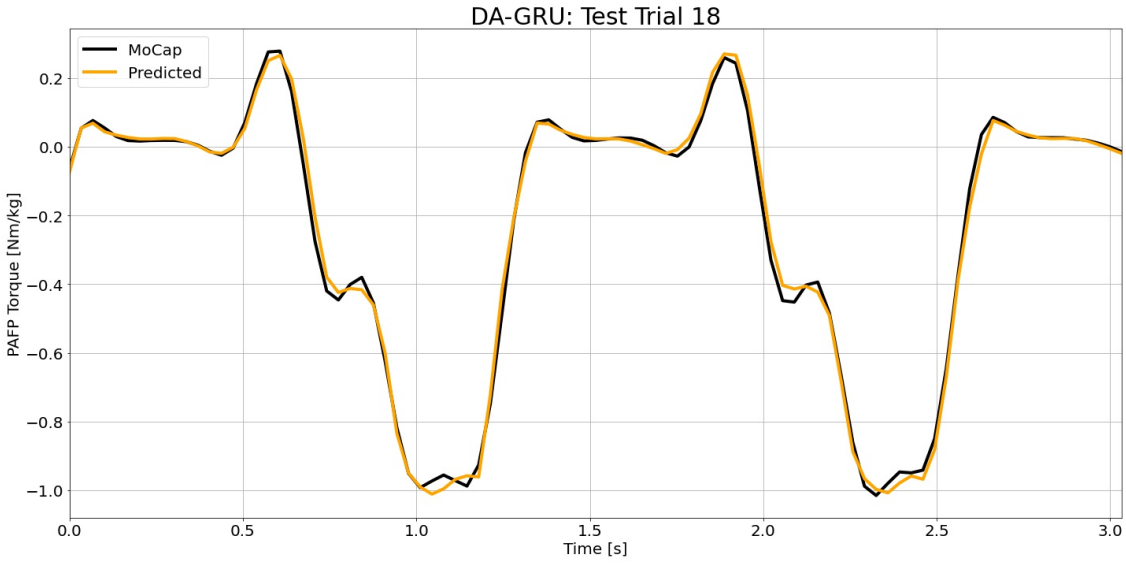


Figure I.132: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 18.

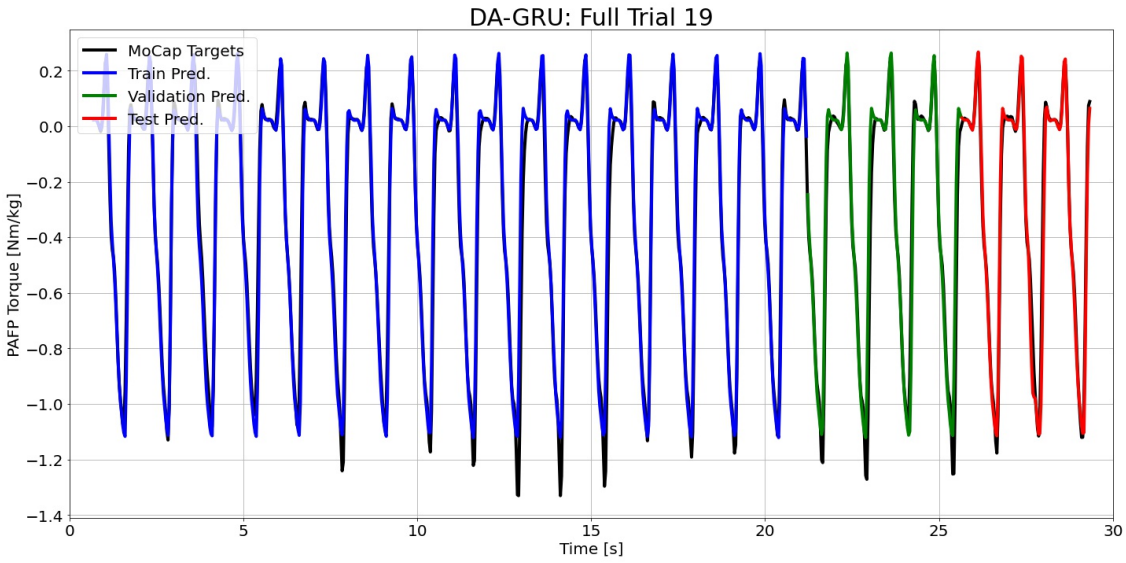


Figure I.133: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 19.

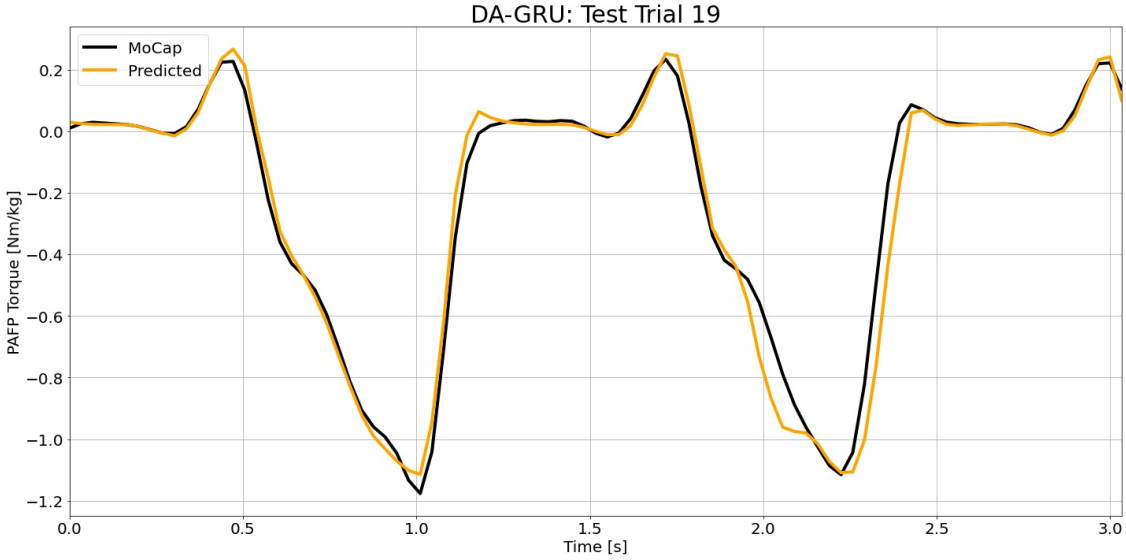


Figure I.134: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 19.

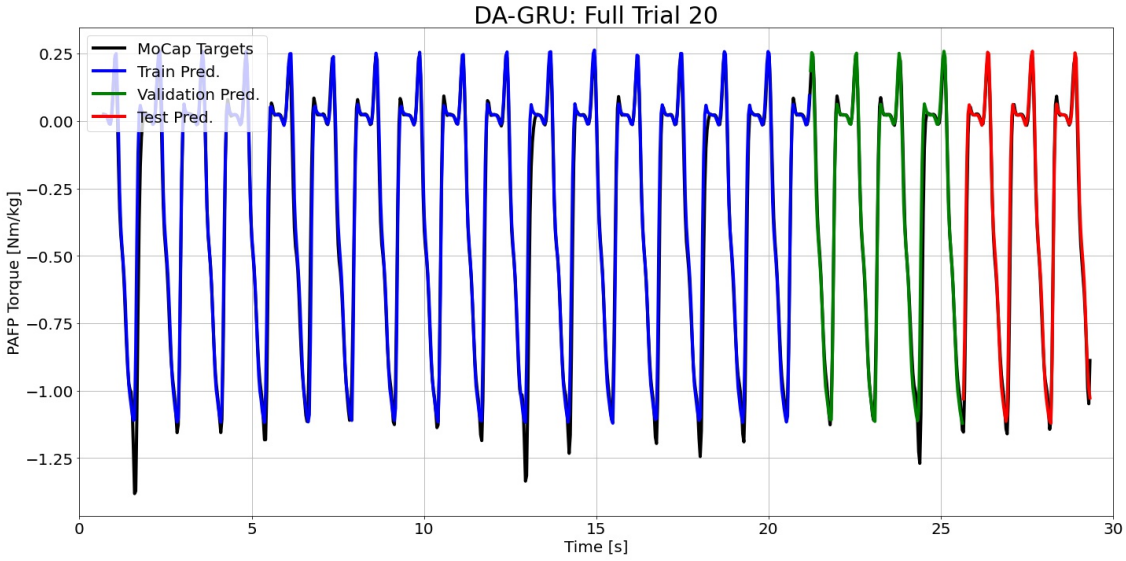


Figure I.135: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 20.

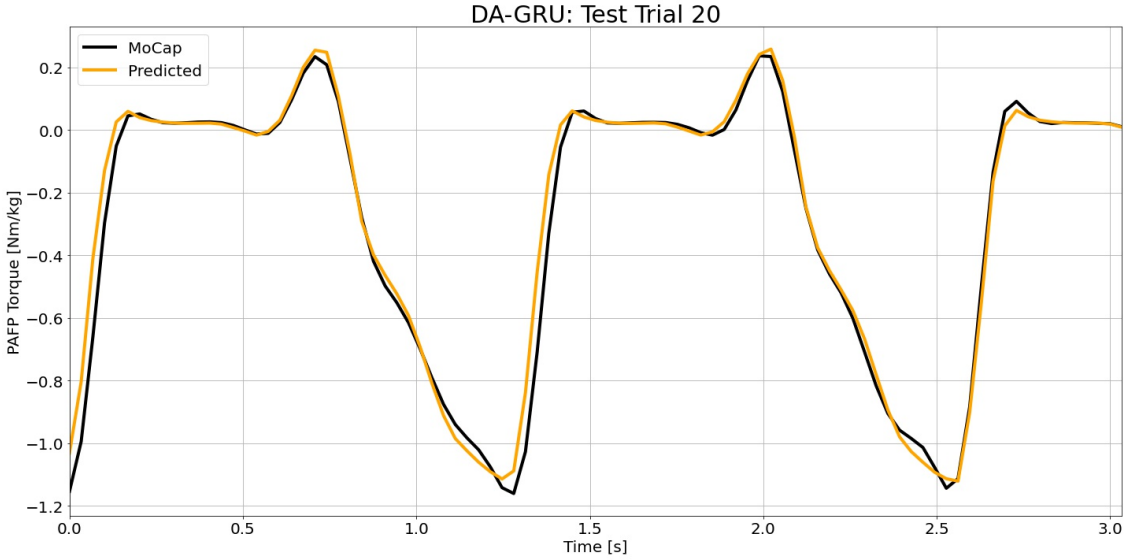


Figure I.136: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 20.

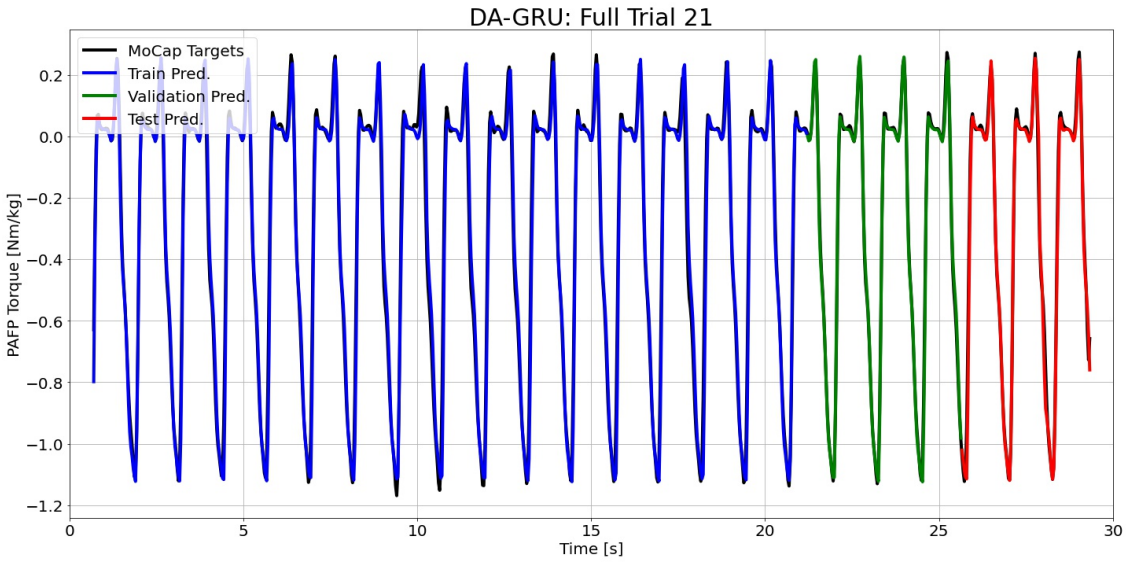


Figure I.137: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 21.

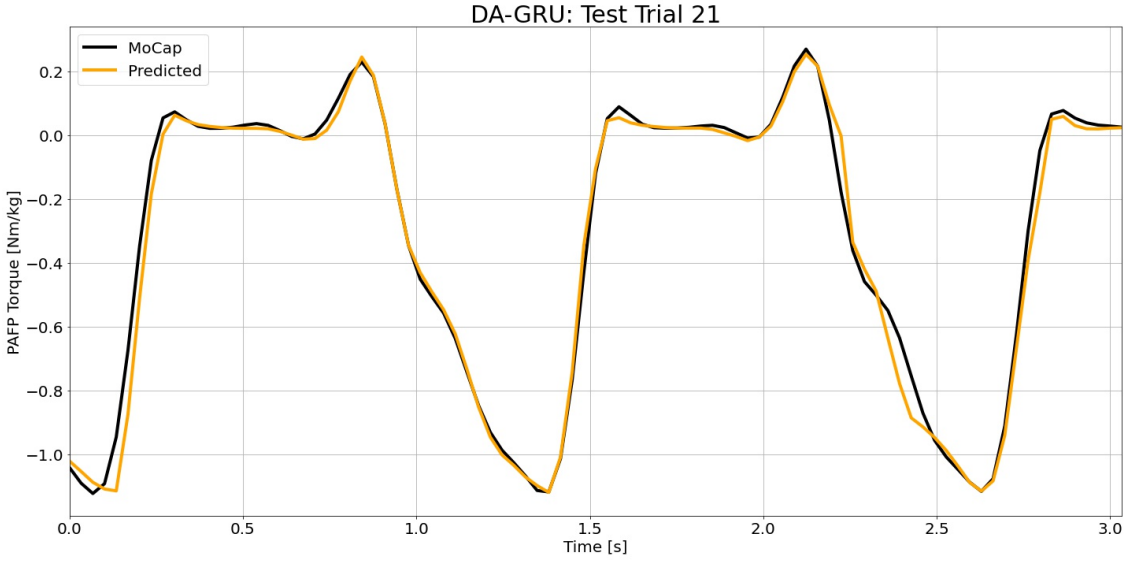


Figure I.138: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 21.

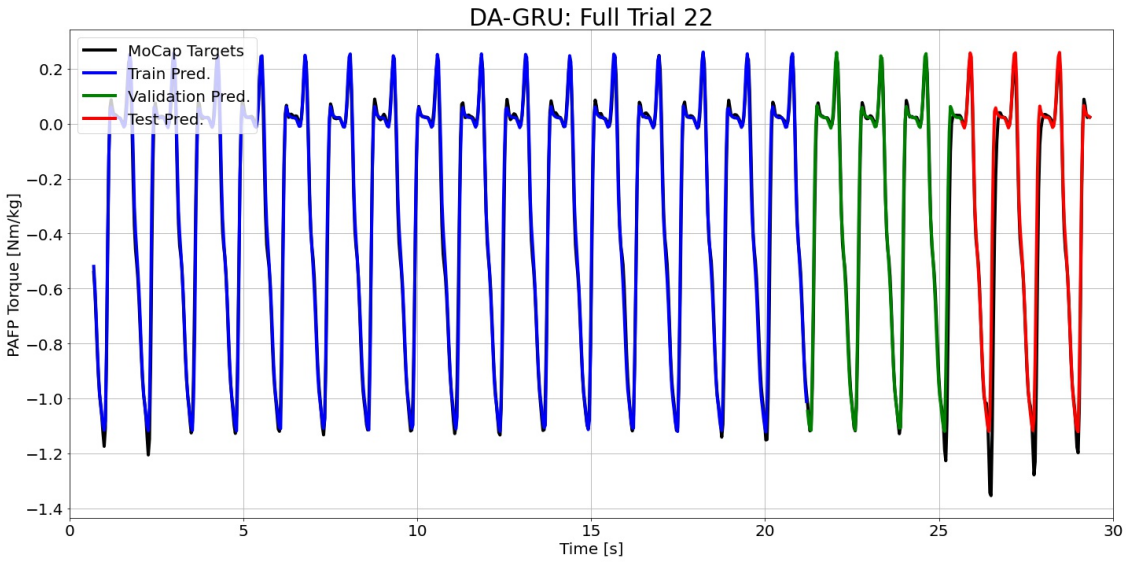


Figure I.139: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 22.

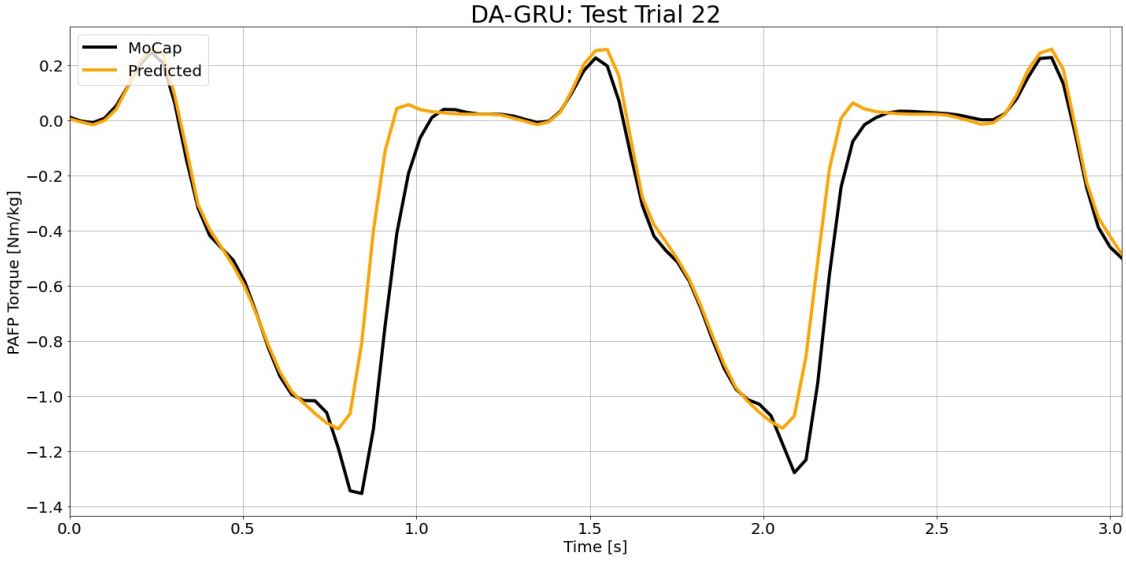


Figure I.140: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 22.

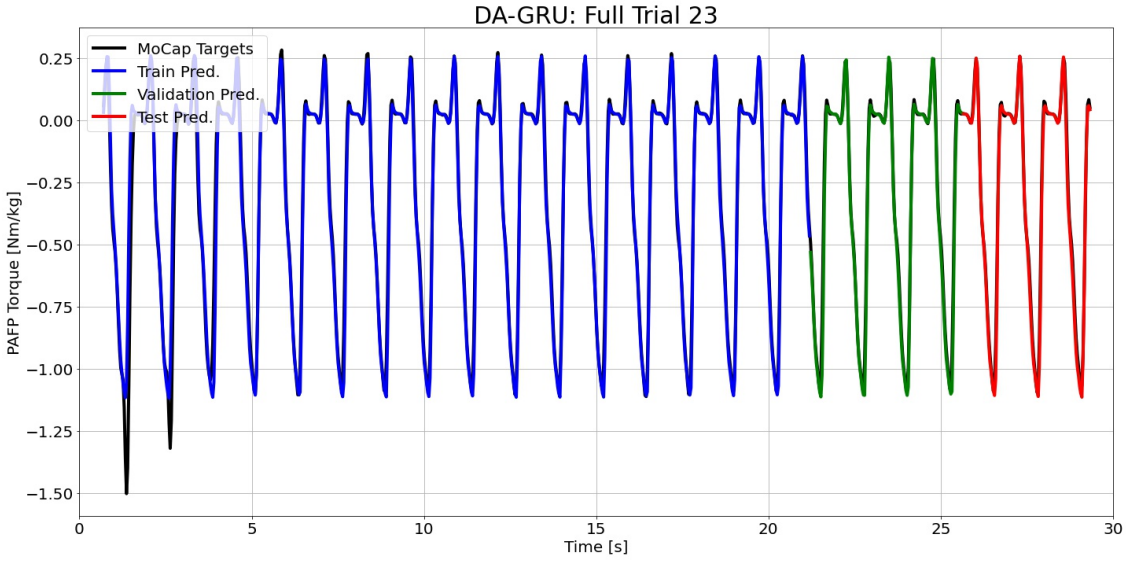


Figure I.141: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for full time series 23.

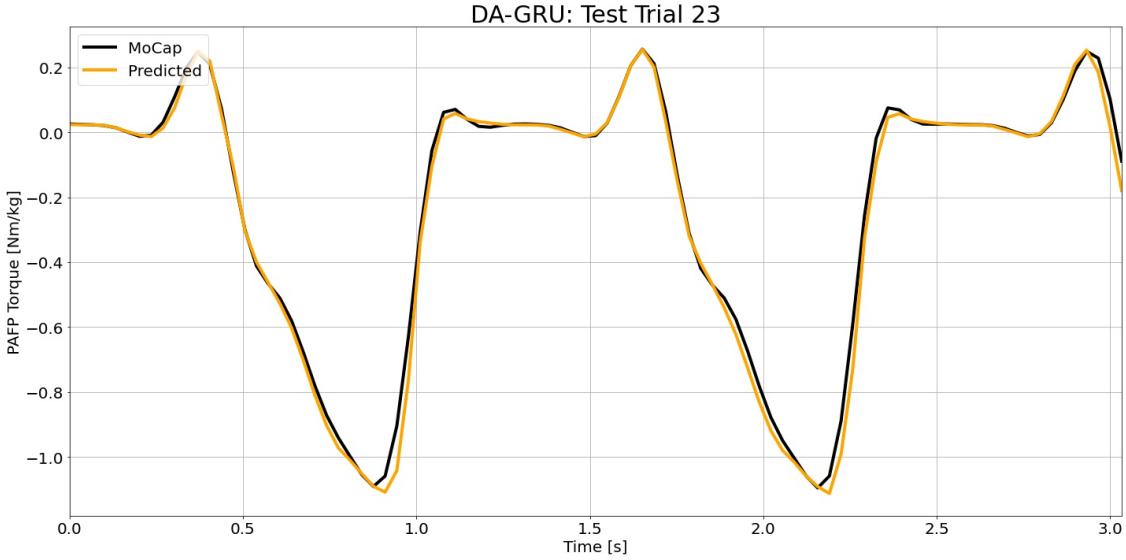


Figure I.142: DA-GRU predictions compared to the MoCap inverse dynamics ground truth values for test time series 23.