BOSTON UNIVERSITY

SARGENT COLLEGE OF HEALTH AND REHABILITATION SCIENCES

Dissertation

WHAT MOVES US: STUDIES OF PROPULSION MEASUREMENT IN POSTSTROKE WALKING

by

ANDRE M. ALVAREZ

B.S., University of Miami, 2014
M.S., University of Miami, 2017

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2025

Approved by

First Reader

Cara L. Lewis, PhD, PT
Professor of Physical Therapy
Professor of Health Sciences
Associate Professor of Medicine

Second Reader

Louis N. Awad, PhD, PT, DPT
Associate Professor of Physical Therapy

Third Reader

Elliot Lee Saltzman, PhD
Associate Professor of Physical Therapy

The wearisome grand passions and distasteful excitements of active lives, stressed to breaking point, are supplanted in the end by the implacable lassitude of walking: just walking. Serenity is the immense sweetness of no longer expecting anything, just walking, just moving on.

Frèdèric Gros, *A Philosophy of Walking*

WHAT MOVES US: STUDIES OF PROPULSION MEASUREMENT IN POSTSTROKE WALKING

ANDRE M. ALVAREZ

Boston University, Sargent College of Health and Rehabilitation Sciences, 2025

Major Professor: Cara L. Lewis, PhD, PT
Professor of Physical Therapy
Professor of Health Sciences
Associate Professor of Medicine

ABSTRACT

The efficiency of human walking gait is an important factor of what makes walking such a vital part of human daily living. However, this efficiency, particularly that of propulsion forces, is inhibited in populations with neuromotor walking deficits, including stroke. The ability to measure propulsion forces accurately involves the use of large and expensive equipment that is often not available for clinicians to use. With wearable sensors and machine learning, there presents an opportunity to make laboratory-based measurements of propulsion accessible to clinicians.

An initial step in this endeavor was to determine the most significant metrics of propulsion as it relates to distance walked during a 6MWT, a popular outcome measure linked to long distance walking function and increased quality of life. Thus, the first aim of this dissertation was to evaluate the effects of propulsion metrics on long distance walking function using statistical learning methods. The results showed that braking magnitude and impulse of both the paretic and nonparetic limbs were significant predictors of total distance walked, differing from the common focus on

propulsion.

A following step was to confirm that these measurements can be accessible to clinicians in a cost-effective manner. The second aim of this dissertation was to validate the accuracy of an IMU based machine learning algorithm in estimating propulsion metrics versus laboratory-based equipment. The results showed that both propulsion metrics and entire APGRF curves cannot accurately be estimated with the method used.

While the insight of braking metrics as a predictor of long distance walking function is useful, more work can be done to tailor accessible technologies to the needs of clinicians.

Contents

List of Tables

List of Figures

List of Abbreviations

6MWT	.....	6 Minute Walk Test
APGRF	.....	Anterior-Posterior Ground Reaction Forces
BW	.....	Body Weight
CoM	.....	Center of Mass
CWS	.....	Comfortable Walking Speed
EMG	.....	Electromyography
GRF	.....	Ground Reaction Forces
HMM	.....	Hidden Markov Model
IMU	.....	Inertial Measurement Unit
LOO	.....	Leave-One-Out
MAPE	.....	Mean Absolute Percent Error
NN	.....	Neural Network
RMSE	.....	Root Mean Squared Error
SGD	.....	Stochastic Gradient Descent
ST	.....	Stance Time
StD	.....	Standard Deviation
SVM	.....	Support Vector Machines
TLA	.....	Trailing Limb Angle

Chapter 1

Introduction

1.1 Human Walking Biomechanics

Walking is an important function in human daily living. Stretching back millions of years, humans have engaged in bipedal locomotion as a primary form of moving between two points (Zollikofer et al., 2005). Through evolution, humans have prioritized walking, shown by the significant increase in bipedal walking economy compared to their evolutionary cousins, chimpanzees (Sockol et al., 2007). This importance has stretched to the present day: a large and increasing portion of adults in the United States walk either for leisure or transportation (Kruger et al., 2008; Berrigan et al., 2012). Walking has an important impact on public health as an easily accessible form of moderate exercise than can reduce rates of chronic disease and medical expenditures (Lee and Buchner, 2008).

The efficiency of human walking gait is integral to its importance, and there have been multiple theories that look to explain its mechanics (Perry and Davids, 1992). Saunders et al. (1953) introduced the "six determinants of gait," wherein the goals of locomotion are to minimize the metabolic energy expenditure and the displacement of the body's center of mass (CoM) from a straight line of progression. For a time, this was accepted as true; however, more recent studies have shown that there is little evidence that reducing CoM displacement is important to human walking (Della Croce et al., 2001; Kerrigan et al., 2001). In fact, attempting to reduce CoM displacement leads to increased energy expenditure (Ortega and Farley, 2005;

Massaad et al., 2010; Gordon et al., 2009).

Another theory that attempts to explain the efficiency of walking using an inverted pendulum model. As explained by (Kuo and Donelan, 2010), the stance limb acts as an inverted pendulum, compensating changes in kinetic energy with changes in gravitational energy, minimizing the mechanical work done by the muscles. This is achieved in part by keeping the knee of the stance limb extended to reduce the moment force about the knee, reducing the amount of muscular force needed to support the body weight. Similar applications of conservation of energy are found in the swing limb, which follows a non-inverted pendulum like motion. The principal issue with this model, however, is that it does not explain why increasing walking speed increases energy expenditure; in a true pendulum, there is no force required. While this model helps explain how walking can be economical, it fails to describe why there is any energetic cost to walking.

Dynamic walking - locomotion generated primarily thought the passive dynamics of the legs, regardless of the application of power or control - takes from the inverted pendulum model to explain how the gait cycle can occur through passive dynamics through the study of passive walking machines (Kuo and Donelan, 2010; McGreer, 1990). In this model, the single limb support phase is similar to that of the inverted pendulum model, in that it requires the CoM velocity transitions from one inverted pendulum arc to the next. This occurs because the CoM velocity is directed forward and downward due to being somewhat perpendicular to the trailing limb (the previous stance leg). The new arc begins with a forward and upward velocity, following the leading limb (the next stance leg) (Kuo and Donelan, 2010). The dynamic walking model explains this redirection as a collision between the leading limb and the ground, dissipating energy; the resulting ground reaction force acts, in part, against the CoM velocity, requiring positive work to compensate (McGreer, 1990). For passive walking

machines, this is supplied by the gravitational potential energy of walking down a ramp (McGreer, 1990). When walking on level surfaces, dynamic walking robots have shown this energy can come from ankle pushoff or the hip (Kuo et al., 2005; Collins, 2005; Kuo, 2002).

The theory of dynamic walking can be applied to human walking as well (Kuo and Donelan, 2010). This is explained by (Kuo et al., 2005) through four events: collision, rebound, preload and pushoff. Collision, or initial contact, is defined as when the heel of the leading limb strikes the ground and performs negative work on the CoM with the ankle and knee joints. Quadriceps activation and knee extension of the leading limb mark rebound, which can be aided by the hips performing positive work accelerating the opposite leg through the swing phase. Preload occurs after the midpoint of the stance phase where the ankle joint performs negative CoM work. The Achilles tendon likely aids in elastic energy storage from collision, allowing for an increased pushoff duration over both rebound and preload; for the inverted pendulum motion to contribute energy to ankle muscles for pushoff; and assisting in slowing the inverted pendulum motion resulting in reduced CoM velocity and less energy loss during collision. The cycle of reduction and restoration of energy requires active muscles with associated metabolic costs; this is called the step-to-step transition cost of human walking (Kuo and Donelan, 2010). A large part of gait is focused on reducing the energy cost of step-to-step transitions (Donelan et al., 2002b).

During this step-to-step transition of each gait cycle, a braking force is generated by the leading limb as it makes contact with the ground in front of the body during the rebound event. In order to move the CoM into the next step, a pushoff force produced by the trailing limb is required (Donelan et al., 2002b; Kuo and Donelan, 2010; Zelik and Adamczyk, 2016). These pushoff, or propulsion forces have shown to be important as they are the main sources of positive power during step-to-step

transitions (Cappozzo et al., 1976; Winter, 1983; Hof et al., 1992). This burst of power comes largely from the plantarflexor muscles (the soleus, medial and lateral gastrocnemius) and tendons that work about the ankle joint; the hip muscles (iliopsoas and others) contribute less so. Coordination of the timing and magnitude play a vital role in step-to-step transitions. Through increases in speed, propulsion magnitude and onset are adjusted; propulsion magnitude increases, while propulsion onset - often occurring just before contralateral heel strike - begins earlier (Kuhman and Hurt, 2019). Mistimed and improper propulsion may contribute to inefficiencies and increased metabolic costs (Kuo and Donelan, 2010; Mian et al., 2006; Kramer et al., 2016). The knee and foot perform net negative work during this phase of the gait cycle (Zelik et al., 2015; Zelik and Adamczyk, 2016).

As shown, propulsion is a key factor in human walking. This is in part due to its effect on both the individual limb and the body's CoM (Zelik and Adamczyk, 2016). The first of these is explained by the large majority of energy produced by the ankle plantarflexors; it is stored in the trailing limb as it enters swing phase, with a minimal amount moving across the hip joint to the head, arms and trunk (Winter and Robertson, 1978). However, it has been shown that the trailing limb works on the CoM during the step-to-step transition, increasing its speed and kinetic energy (Donelan et al., 2002a). Both of these occurrences can be explained when it is considered that, while the limb is a small portion of the total body mass, it contributes substantially to the dynamics of the body through its localized acceleration (Zelik and Adamczyk, 2016).

Considering the body as a system of segments, (Zelik and Adamczyk, 2016) proposed an equation (Equation 1.2) that helps to relay this point. In it, the body is divided into two segments: the pushoff limb (mass m_{limb}) and the remainder of the body (mass m_{ROB}). The position of the full body CoM (mass $M = m_{limb} + m_{ROB}$,

position $\vec{r}_{ROB}$) is defined by the position of the CoM of the two segment groups:

$$\vec{r}_{CoM} = \vec{r}_{limb}(m_{limb}/M) + \vec{r}_{ROB}(m_{ROB}/M) \tag{1.1}$$

Taking consecutive time derivatives of CoM position produces CoM acceleration:

$$\vec{a}_{CoM} = \vec{a}_{limb}(m_{limb}/M) + \vec{a}_{ROB}(m_{ROB}/M) \tag{1.2}$$

As shown in Equation 1.2, the acceleration of the CoM ($\vec{a}_{CoM}$) is affected by both the accelerations of the trailing limb and the remainder of the body, in relation to their contribution to the total body mass ($\vec{a}_{limb}(m_{limb}/M)+\vec{a}_{ROB}(m_{ROB}/M)$). As the trailing limb accounts for relatively little of the total body mass, it may be determined that the remainder of the body would be the primary contributor to $\vec{a}_{CoM}$; however, this would not be the case if $\vec{a}_{limb}$ is sufficiently large when compared to $\vec{a}_{ROB}$. This so happens to be true in the case of human walking: $\vec{a}_{limb}$ is large, directed forward and upward; $\vec{a}_{ROB}$ is small and works in the opposing direction (Lipfert et al., 2014). This equation shows that, with a large enough change in velocity or position, the smaller mass of the trailing limb can be a vital contributor to whole-body energy change. This allows propulsion to be a major factor in the flow of the gait cycle, creating walking patterns that are fast, stable and efficient (Browne and Franz, 2017; Kuo and Donelan, 2010).

However, the inability to efficiently transition from step to step is commonly seen in populations with neuromotor walking deficits - including stroke (Bethoux and Bennett, 2011; Mahon et al., 2015; Hass et al., 2005). Stroke is one of the leading causes of long-term disability in the United States. It is estimated that 7 million Americans have had a stroke, with about 795,000 new or recurrent occurrences each year; it is projected that an additional 3.4 million people will have a stroke by 2030 (Virani et al., 2020). A majority of poststroke individuals have a walking gait that

is labor intensive and inefficient; walk at speeds that are not safe for walking in the community; and have much reduced physical activity, leading to severe effects on their physical healthy and quality of life (Reisman et al., 2009; Farris et al., 2015; Awad et al., 2016; Duncan et al., 2011; English et al., 2014; Michael et al., 2005; Rand et al., 2010). While the primary goal of stroke rehabilitation is the return to efficient walking patterns, current therapies are insufficient in reducing this disability (Dickstein, 2008; Danks et al., 2014; Dean et al., 2012; Sullivan et al., 2014).

A common characteristic of post-stroke walking is the inability of the paretic limb to provide pushoff work, due in part to impaired propulsion function resulting from aging or neurological injury (Chen et al., 2005; Jonkers et al., 2009; Bowden et al., 2006; Franz, 2016). As ankle plantarflexors are the primary generators of propulsive power during walking, impaired plantarflexion function can lead to inefficient walking patterns. This impairment results in the redistribution of positive work from the ankle joint to the knee and hip joints, due to mistimed and insufficient propulsion; this distal-to-proximal redistribution of positive work does not fully offset the reduction in plantarflexor work (Bowden et al., 2006; Chen and Patten, 2008; Farris et al., 2015). This leads to a lack of kinetic energy in the paretic limb at toe-off (the beginning of swing phase) reducing knee flexion both at toe-off and during swing, resulting in increased functional leg length during swing (Chen et al., 2005; Balaban and Tok, 2014). To compensate, subjects often raise their trunk during pre-swing and swing - in what is referred to as "hip hiking" - and increase lateral movement of the foot during swing - known as circumduction. Both of these strategies help to provide floor clearance of the limb to avoid tripping (Chen et al., 2005; Balaban and Tok, 2014). During the stance phase, the impaired single limb support of the paretic limb leads to an increase in the non-paretic pre-swing kinetic energy and decrease in swing time due to weakness or poor balance; this may also be compensated for with larger step

widths (Chen et al., 2005).

Trailing limb angle also plays an important role in propulsion function, translating ankle plantarflexion torque into propulsion. Healthy individuals modulate both trailing limb angle (TLA) and ankle moment to increase propulsive force; individuals post-stroke rely more heavily on changes in TLA to modulate speed, perhaps due to impaired plantarflexor strength (Hsiao et al., 2015a,b). This leads to compensatory reliance on the non-paretic limb for propulsion that is characteristic of post-stroke hemiparesis; paretic limb propulsion can be up to 68% less than that of the non-paretic limb, and is linked to impaired walking function and hemiparetic severity (Awad et al., 2015a; Bowden et al., 2008; Hsiao et al., 2016a; Bowden et al., 2006; Farris et al., 2015; Jonkers et al., 2009; Turns et al., 2007). Within the post-stroke population, levels of paretic propulsion are influential: those who are considered unlimited community ambulators based on their walking speed and distance have relatively high levels of paretic propulsion, while the paretic propulsion of home ambulators is substantially lower (Bowden et al., 2013; Hsiao et al., 2016a,b; Fulk et al., 2017). Deficits in paretic propulsion have also been related to long distance walking function - a crucial determinant of community participation and perceived quality of life (Awad et al., 2015b; Browne and Franz, 2017; Neptune et al., 2001; Combs et al., 2013).

Differences in kinematic, kinetic and metabolic features of walking gait between healthy and stroke subjects is heavily reported (Chen et al., 2005; Balaban and Tok, 2014; Reisman et al., 2009; Ellis et al., 2013; Detrembleur et al., 2003). The priority of current rehabilitation protocols is to obtain walking independence as quickly as possible, at the cost of the functional restoration of the paretic limb (Kitago and Krakauer, 2013). This is often measured through walking speed, as it is associated with many temporospatial parameters of gait (Balaban and Tok, 2014). While increases in walking speed are linked to walking independence, short term improvements

are often the result of compensatory strategies (Chen et al., 2005). Common solutions such as ankle-foot orthoses, while they aid the paretic limb in ground clearance, have negative effects on propulsion and gait adaptability, which increase the energy cost of walking (van Swigchem et al., 2014; Vistamehr et al., 2014; Wutzke et al., 2012).

Higher energy costs of walking in both older adults and populations with neuromotor walking deficits have been found to be a primary contributor to physical inactivity, as shown by the energy cost of walking being up to twice as high for those post-stroke compared to healthy walkers (Franceschini et al., 2013; Lapointe et al., 2001; Moore et al., 2010; Wert et al., 2013; Detrembleur et al., 2003). Even the energy cost of walking of community-dwelling poststroke individuals is double of what is predictive of future mobility decline (Awad et al., 2016). When made to walk at more normal walking speeds, poststroke individuals consume less energy than at their normal speeds; however, this level is still highly inefficient due to reliance on gait compensations (Hsiao et al., 2016a; Reisman et al., 2009; Detrembleur et al., 2003; Bowden et al., 2013). As gait symmetry has been shown to be an important part in minimizing the energy cost of walking in healthy subjects, reducing asymmetries in stroke gait has shown to produce faster, less costly walking (Ellis et al., 2013; Awad et al., 2015b).

Propulsion is an important function of human walking gait; despite this, conventional rehabilitation efforts have been unable to restore patient's ability to generate propulsion symmetrically from each limb. The development and study of interventions targeting propulsion function is an active area of research (Awad et al., 2017b; Browne and Franz, 2019). However, there remains a need for measurement instruments, accessible to clinicians, that allow for the clinical management of individual limb propulsion deficits. Wearable sensors provide an opportunity to bridge this gap.

1.2 Wearable Sensors

For decades, sensors have been used to study the biomechanics of gait (Cutting and Kozlowski, 1977). Paralleled with the boom in sensing technology, many different technologies have been used in gait detection, such as video-based (Wang et al., 2003); floor-based (Vera-Rodriguez et al., 2012); and wearable sensor based technology (Gafurov, 2007). Of these, the development of wearable sensors for the study and advancement of gait interventions is a highly active area of research (Stanton et al., 2017; Sprager and Juric, 2015; Taborri et al., 2016).

Some sensors - such as footswitches, foot pressure insoles and electromyography (EMG) - have applications in the analysis of gait; however, they also have major limitations. Footswitches are considered the gold standard for gait phase detection among wearable sensors, as they directly detect foot contact with the ground; as such, they are often used to validate other types of sensors (Abaid et al., 2013; Mannini and Sabatini, 2012; Taborri et al., 2014). As they can only detect foot contact with the ground, footswitches are unable to detect anything during the swing phase and cannot measure forces during the gait cycle (Pappas et al., 2004). Patients with impaired gait may also produce issues in accuracy and reliability as their gait patterns can be irregular (Aminian et al., 2002).

Foot pressure insoles improve on footswitches, as they can record contact of the full foot with the ground along with ground reaction forces. Combined with machine learning, this allows for more time sensitive gait detection and the discrimination of different parts of the swing phase (Crea et al., 2012; Jacobs and Ferris, 2015; Catalfamo et al., 2008). However, foot pressure insoles have limitations of their own, such as compromising foot-ground interaction; repeated mechanical stress leading to sensor wear out; and participant discomfort (Ancillao et al., 2018). EMG signals are used in gait phase detection due to repeatable muscle activity in the lower body (Hof

et al., 2002), and in studying muscle coordination and synergies (Clark et al., 2010; Steele et al., 2015). Drawbacks of EMGs include higher complexity in acquiring and processing data, as well as the price of equipment.

Inertial sensors allow for the study of walking biomechanics beyond gait phases, such as spatiotemporal (*e.g.* step length, cadence) and kinematic parameters (*e.g.* joint angles). Using kinematic parameters, ground reaction forces can be estimated; these are of particular importance, as they allow for the study of kinetic forces during walking and the calculation of internal joint forces and moments through inverse dynamics (Ancillao et al., 2018). Examples of inertial sensors include accelerometers, gyroscopes and inertial measurement units (IMUs).

Accelerometers are sensors that measure linear acceleration of a body in its own instantaneous rest frame. As they are small, inexpensive and readily available (Kavanagh and Menz, 2008), accelerometers are a common solution in gait analysis, using a variety placements and number of sensors (Taborri et al., 2016). Multiple studies have investigated various positioning of sensors on the body with the goal of studying segmentation of the gait cycle and walking incline (Mijailoviü et al., 2009; Rueterbories et al., 2014; Selles et al., 2005). For this, peaks at the start and end of the stance phase in the anterior-posterior direction allow for the detection of two gait phases (Taborri et al., 2016). Accelerometers have also been shown to provide accurate spatiotemporal parameters for stroke subjects, both in the laboratory and community (Moore et al., 2017). In these and other applications, signal shaping procedures are necessary to improve performance; a machine learning approach is not mandatory, however. Despite their many applications, accelerometers are imperfect: gravity must be compensated for when calculating body segment acceleration; short term drift error occurs when calculating linear velocity and position due to numerical integration of the signal and high frequency noise, resulting in a process that is com-

putationally expensive; and it is required to place the sensors correctly on the body segment, along with a calibration procedure.

Gyroscopes are sensors used to measure angular velocity and are a popular solution for gait detection as they are not influenced by gravity or vibrations due to heel strike (Taborri et al., 2015a; Mayagoitia et al., 2002). Just as with accelerometers, sensor placement has been extensively studied (Catalfamo et al., 2010; Mannini and Sabatini, 2011, 2012; Taborri et al., 2015b). Based on this, gyroscopes can detect up to six separate gait phases when combined with machine learning approaches, even during daily activity. However, also like accelerometers, they are affected by drift errors due to numerical integrations to compute angular position; this drift occurs more slowly, however.

Combining multiple different sensors - such as an accelerometer, gyroscope and magnetometer - IMUs are able to compensate for the drawbacks of each using sensor fusion techniques such as Kalman filtering. For example, they address drift error found in both accelerometers and gyroscopes, and use magnetometers as a heading reference in the calculation of global variables; this allows for more robust analyses and the computation of spatiotemporal parameters and kinematic variables such as joint angles (Evans and Arvind, 2014; Donath et al., 2016; Dejnabadi et al., 2005). IMUs can allow for the estimation of ground reaction forces through their measurement of kinematic data, using either biomechanical models (Aurbach et al., 2017; Karatsidis et al., 2016; Ohtaki et al., 2001) or machine learning (Guo et al., 2017; Wouda et al., 2018). From these approaches, vertical ground reaction forces can be accurately estimated during the single stance phase; calculating the forces for each limb during double support is more difficult. While there are methods of estimating anterior-posterior ground reaction forces, they face difficulties with neurologically impaired populations, as current methods depend on assumptions of healthy, consistent walking

patterns (Ancillao et al., 2018; Shahabpoor and Pavic, 2018).

Specialized laboratory equipment, such as force plates and optical motion tracking systems, are the gold standard for measuring kinetics and kinematics in both healthy (Franz and Kram, 2013b; Franz et al., 2014) and neurologically impaired walking (Bethoux and Bennett, 2011; Bowden et al., 2006; Farris et al., 2015; Halliday et al., 1998; Hass et al., 2012; Jonkers et al., 2009; Martin et al., 2002; Turns et al., 2007). Located either in the ground or as part of an instrumented treadmill, force plates measure kinetic forces using multicomponent load cells. Combined with optical motion tracking - which uses reflective markers to track body segments - joint forces and moments can be obtained through the use of inverse dynamics. While their level of accuracy is unparalleled, there are difficulties with these instruments. In regards to in-ground force plates, several trials are needed to obtain an adequate amount of data as often only a few strides can be obtained due to the number of force plates and inconsistent foot strikes; this limits the number of consecutive gait cycles that can be reviewed. Instrumented treadmills solve the problem of limited gait cycles, but introduce the difficulty of unrealistically consistent gait speed and the differences in biomechanics between overground and treadmill walking. These, along with optical motion capture, require both large, dedicated spaces and are prohibitively expensive. Due to this, they are often found only in laboratory settings and are inaccessible to most clinicians.

Wearable, inertial sensors provide a way to solve these complications presented by force plates and optical motion tracking. Many studies have shown that inertial sensors can produce data of comparable quality to these gold standards, such as: spatiotemporal parameters at different speeds and slopes, in the community and across different populations (Donath et al., 2016; Moore et al., 2017; Trojaniello et al., 2014); kinematic parameters at different speeds (Mayagoitia et al., 2002); quantification of

gait symmetry (Zhang et al., 2018) and estimations of kinetic parameters such as ground reaction forces (Pieper et al., 2019; Karatsidis et al., 2019; Ryu and Park, 2018; Lim et al., 2019). Of all the benefits of wearable sensors, the ability to study gait parameters outside the laboratory may be the most promising (Bejarano et al., 2015; Boutaayamou et al., 2015; Miyazaki et al., 2019; Peruzzi et al., 2011; Seel et al., 2014; Yang et al., 2013). As they are inexpensive and do not require large amounts of dedicated space, wearable sensors can help bridge the gap between the laboratory and clinic.

Wearable sensors are not without their drawbacks when compared to specialized laboratory equipment. State-of-the-art technology is often more accurate and precise, as they are the gold standard that others are compared to. For example, force plates can measure ground reaction forces directly, while any kinetic measure taken from IMUs will be estimations. Wearable, inertial sensors also lack the ability to find an absolute heading and accurate joint angle measurements without a magnetometer; the latter can be dealt with through the assumption of an always-zero yaw, limiting joint angle measurements to two dimensions.

One potential use for wearable sensors in a clinical setting is biofeedback. Biofeedback is the use of technology to provide information to the learner that is not consciously available by transforming biological signals into an output they can understand and has been found to improve walking and balance when compared to usual therapy (Stanton et al., 2017). Training of this kind is advantageous to conventional therapy due to it's targeted intervention, sensitive and immediate feedback, motivating effect due to game-like features and the ability to perform supervised exercises from home (Horak et al., 2015; Zijlstra et al., 2010).

Information such as the kinetics, kinematics and muscle activation of walking can be delivered through the visual (Karatsidis et al., 2018), auditory and tactile senses

(Schenck et al., 2019). Of these, visual feedback is the most commonly reported in the literature, as is information on kinematic parameters (van Gelder et al., 2018). It has been found that providing feedback using multiple senses results in more positive outcomes than separate modes of feedback (Sigrist et al., 2013); in addition, combined visual and sensory feedback has been shown to outperform each individually (Yen et al., 2014). Biofeedback on muscle activity lags behind those of kinematic, kinetic and spatial-temporal parameters (Tate and Milner, 2010). In regards to ground reaction forces, kinetic biofeedback increased both propulsive forces and muscle activation during push-off compared to feedback on muscle activity (Franz et al., 2014); however, there is no consensus of any form of feedback being superior in regards to ground reaction forces (Agresta and Brown, 2015).

Biofeedback may provide an avenue to assist in the rehabilitation of walking after stroke, particularly in regards to propulsion (Franz et al., 2014). An increase in peak anterior-posterior ground reaction forces in a single, unilateral leg after a period of biofeedback training encourages the use of such training in post-stroke individuals (Schenck and Kesar, 2017). Using biofeedback focused on increasing anterior-posterior ground reaction forces has shown to produce positive changes in trailing limb angle and ankle plantarflexor moment and push off in those post-stroke (Genthe et al., 2018); both of these parameters encourage the retraining of a more normal, economic gait pattern. This could also occur with the reduction of hip power output during walking, known as distal-to-proximal redistribution, that is common in post-stroke individuals (Browne and Franz, 2019). Along with a myriad of other positive changes, such as peak ankle power and stride length (Jonsdottir et al., 2010), biofeedback may provide an opportunity of effective gait retraining in those post-stroke.

Unfortunately, the vast majority of published literature on biofeedback is in a laboratory setting (van Gelder et al., 2018) and is focused on upper limb, cognitive

and balance rehabilitation, particularly though the use of virtual reality and video game applications (Gamito et al., 2017; Li et al., 2016; Perez-Marcos et al., 2017; Laver et al., 2017). There remains a need for walking biofeedback interventions using wearable sensors that can be translated to a clinical setting. There is a potential tool in using wearable sensors for walking biofeedback in machine learning.

1.3 Statistical and Machine Learning

Machine learning can be defined as programming a computer to adapt or change its outputs based on certain inputs, where the goal is to have those outputs reflect the correct ones (Marsland, 2011). By this definition, the algorithms produced are said to be learning if they adapt so that their performance improves - in short, they learn from the data (Hastie et al., 2009).

A common framework of machine learning is statistical learning, where statistics and functional analysis is used to find a predictive function based on the data (Hastie et al., 2009).

Classification consists of taking a vector of inputs and deciding which of a number of classes they belong to (Marsland, 2011; Bishop and Nasrabadi, 2006). In most cases, the classes are taken to be separate, so that each input belongs to a single class and the set of classes accounts for all of the possible output space; this is not always realistic, and at times fuzzy classifiers or output histograms are used to solve this problem (Marsland, 2011; Bishop and Nasrabadi, 2006; Goodfellow et al., 2016). Though there are different ways of finding a solution, they all aim to do the same thing: determine different decision regions separated by decision boundaries, the most simple example being linear boundaries (Marsland, 2011; Bishop and Nasrabadi, 2006; Hastie et al., 2009).

Regression is a statistical solution that aims to fit a linear mathematical function

that passes as close as possible to all the data points (Marsland, 2011). Because they are linear, the change in the dependent variable is predicted by the independent variable in a constant rate. Whether done by directly constructing an appropriate function or modeling the predictive distribution, linear models for regression are simple, and are often sufficient to gain insight into how the inputs affect the output in lower dimensional spaces (Bishop and Nasrabadi, 2006). Though this can be useful in higher dimensional spaces, other methods may be better suited for that purpose; however, it does form the foundation of more sophisticated models, including nonlinear ones (Marsland, 2011; Bishop and Nasrabadi, 2006; Hastie et al., 2009).

Multiple or multiregressor regression is an extension of simple linear regression where multiple independent variables are used to predict the outcome variable (Marsland, 2011). This is done by modeling the relationship between dependent and independent variables simultaneously. Multioutput regression extends this to cases where there are multiple output variables, allowing one to identify the best subset of predictor variables that explain the variance in each output (Marsland, 2011).

A common method of building a multiregressor model is forward stepwise regression, an iterative algorithm that selects variables based on their contribution to the predictive power of the model by minimizing the loss function, typically mean squared error (MSE) (Hastie et al., 2009)Ṫhis is done by starting with a model with no regressors and sequentially adding variables based on their correlation with the current residuals.

Machine learning using statistical analysis can come in different forms. In supervised learning, quantitative or categorical outcome measurements are based on individual measurable properties or characteristics of what is being observed, referred to as features (Bishop and Nasrabadi, 2006). In turn, they are developed using a training set of data with the correct responses to allow the algorithm to generalize

its response to all possible inputs (Hastie et al., 2009; Marsland, 2011). This generalization is important: the ability to produce sensible outputs for inputs which were not met during learning - and dealing with the noise found in real-world data - is the central goal of machine learning (Marsland, 2011; Bishop and Nasrabadi, 2006). Two of the most common examples of supervised learning are regression and classification.

In unsupervised learning, the correct responses are not provided; only the features are observed and the algorithm is tasked with describing how the data are organized or grouped based on similarities (Hastie et al., 2009; Marsland, 2011). This can take the form of finding groups of similar examples in the data, known as clustering; determining the distribution of data within an input space, called density estimation; or to simplify data from a high dimensional space down to two or three dimensions, referred to as dimensionality reduction or component analysis (Bishop and Nasrabadi, 2006; Goodfellow et al., 2016). Reinforcement learning falls somewhere between supervised and unsupervised learning: the algorithm is told when the answer is wrong but is not told the correct answer, requiring it to explore and try different possibilities (Marsland, 2011). Here, there should be a balance between exploration (trying new actions to test their effectiveness) and exploitation (making use of actions that are known to be successful) as a focus on either will lead to poor results (Bishop and Nasrabadi, 2006). Finally, evolutionary learning takes inspiration from biological evolution, using an idea of fitness to determine the viability of the current solution (Marsland, 2011).

Although machine learning can take many forms, there are key ideas that provide the foundation to common problems and solutions. Regardless of the machine learning approach taken, a training set will be used to tune the parameters of an adaptive model (Bishop and Nasrabadi, 2006). This set is made up of data that has been inspected in advanced in order to train the model. Once trained, the performance of the model can be tested on a test set in order to determine generalization of the

model. As the input vectors of the training data can only contain a small fraction of all possible inputs, the generalization of the model is a central goal.

A common method to evaluate performance and generalization of a machine learning model is cross-validation (Brunton and Kutz, 2021). This method allows the model to perform on unseen data by splitting the dataset into testing and training sets multiple times, with performance metrics being averaged across each split. The most robust form of cross-validation is leave-one-out (LOO) cross-validation, where a single observation is used for testing and all others are used in the training set, allowing for each observation to be used in both the training and testing sets (Brunton and Kutz, 2021). As this results in k-1 models being trained, LOO cross-validation is computationally expensive; however, it is often the most complete form of cross-validation as it provides the most accurate estimation of model performance.

For classification, accuracy, sensitivity and specificity are commonly used to determine performance:

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \\ Sensitivity &= \frac{TP}{TP + FN} \times 100\% \\ Specificity &= \frac{TN}{TN + FP} \times 100\% \end{aligned} \tag{1.3}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN is the number of false negatives (Begg and Kamruzzaman, 2005). Studies that focused on real-time estimations also compared the computational load of their models (Taborri et al., 2014).

Performance of a regression analysis can determined using root mean squared error (RMSE), mean absolute percent error (MAPE), R^2, and adjusted R^2 (Hastie et al., 2009).

$$RMSE = \frac{\sqrt{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}}{N} \tag{1.4}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{1.5}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{N}(Y_i - \bar{Y}_i)^2} \tag{1.6}$$

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n-1)}{n-k-1} \tag{1.7}$$

RMSE measures the average of the squared difference between the predictions and the ground truth, allowing for the detection of outliers; it also is in the same units as the target variable, making it more interpretable and easier to compare across models. MAPE measures the average absolute difference between the predicted and actual target values, allowing for a clear understanding of how the model has performed in relation to the true values. Finally, R^2 uses the regression and total sum of squares to determine the variance between the model and a linear model; R^2_{adj} adds a penalty for the total number of observations, allowing for a better comparison between models with a different amount of variables.

When studying bipedal locomotion biomechanics, three methods are common in the literature: support vector machines (Tahir and Manap, 2012; Begg et al., 2005; Begg and Kamruzzaman, 2005; Mannini et al., 2016), neural networks (Tahir and Manap, 2012; Kaczmarczyk et al., 2009; Alaqtash et al., 2011) and Hidden Markov Models (Mannini et al., 2016; Cuzzolin et al., 2017).

When studying bipedal locomotion biomechanics, one common machine learning method is support vector machines (Tahir and Manap, 2012; Begg et al., 2005; Begg and Kamruzzaman, 2005; Mannini et al., 2016). Using support vector machines (SVMs), known as support vector regression (SVR) when used in regression analysis,

a feature vector with dimension m, (Begg et al., 2005) describes a hyperplane in m dimensional space is found that linearly separates the two classes on either side of the hyperplane, or decision surface, whose equation is:

$$\vec{w}^T\vec{x} + b = 0 \tag{1.8}$$

where $\vec{w}$ is the adjustable weight vector and b is the hyperplane bias. With a classification output of $\{+1, -1\}$, the linearly separable case can be represented as:

$$\begin{aligned} \vec{w}^T\vec{x} + b \leq 0 \quad for \quad d_i = -1 \\ \vec{w}^T\vec{x} + b > 0 \quad for \quad d_i = +1. \end{aligned} \tag{1.9}$$

As mentioned previously, in realistic situations - such as bipedal locomotion biomechanics - datasets might not be linearly separable. This has two potential solutions: applying nonlinear transforms, such as binning, or using kernels (Begg et al., 2005). In binning, values of a continuous variable are put into bins with values around it; this may address issues such as missing values, presence of outliers, statistical noise and data scaling. Unfortunately, this can be unmanageable with larger values of m (Begg et al., 2005). This leads to the use of kernels, which use kernel functions (such as polynomials, sigmoid functions and radial basis functions) to map the non-linear observations into a higher dimensional space where they are separable (Marsland, 2011; Bishop and Nasrabadi, 2006; Hastie et al., 2009). These are similar to m-dimensional vectors, with an exception: the inner products required by SVM are computed without explicitly building the high-dimensional representations. How well a hyperplane fits in feature space is measured as the distance between the hyperplane and the support vectors (Begg et al., 2005). An advantage of SVMs is its low computational cost; the solution of the optimization is straightforward due to the objective function

being convex, and the number of basis functions in the model is much smaller than the number of training points (Bishop and Nasrabadi, 2006).

An alternative to SVMs is neural networks (NNs). Started as an attempt to mathematically model information processing in biological systems, NNs can be thought of as a system of neurons that take inputs and produce outputs, often represented by a neural diagram as in Figure 1·1 (Bishop and Nasrabadi, 2006; Hastie et al., 2009). NNs that contain more than a single hidden layer are often called deep neural networks.

In this approach, the number of basis functions are fixed in advance and allowed to be adaptive during training (Bishop and Nasrabadi, 2006). With this, it has come to include a large amount of models and learning methods. At their core, they remain fairly simple: they are nonlinear statistical models (Hastie et al., 2009). In a K-class classification, there are K target measurements (Y_k, k = 1,...,K), each coded as a 0 or 1 variable for the kth class. Linear combinations of inputs for the derived features (Z_m) are again linearly combined to create the target (Y_k), as shown in (Hastie et al., 2009):

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, ...M, \\ T_k &= \beta_k + \beta_k^T Z, k = 1, ...K, \\ f_k(X) &= g_k(T), k = 1, ...K, \end{aligned} \tag{1.10}$$

where $Z = (Z_1, Z_2, ...Z_M)$ and $T = (T_1, T_2, ...T_K)$. The activation function $\sigma(v)$ is often chosen to be the sigmoid function $\sigma(v) = 1/(1 + e^{-v})$; alternatively, Gaussian radial basis functions are also used (Hastie et al., 2009).

In probability theory, a chain is a sequence of possible states whose probability is a function of the previous states. Markov chains use the Markov property: the

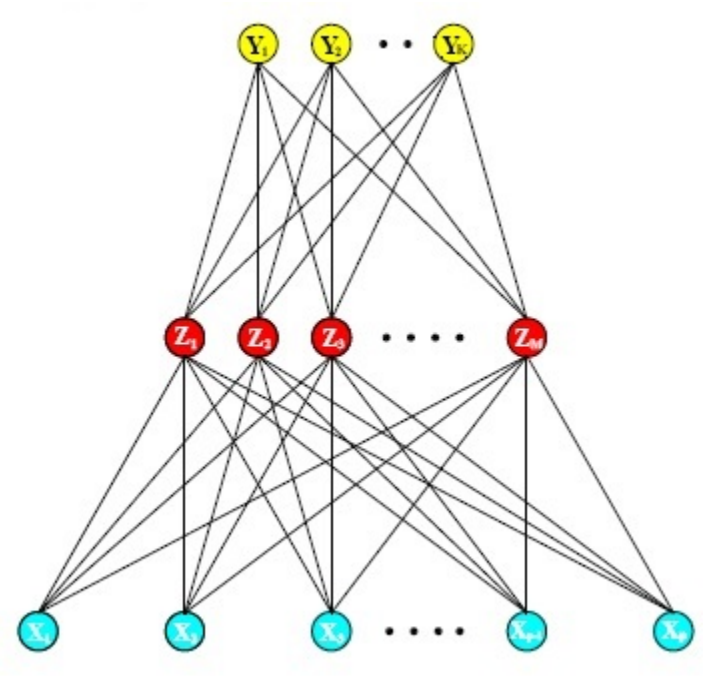


Figure 1·1: Example of a single hidden layer, feed-forward neural network (Hastie et al., 2009). The top layer (Y) are in inputs; the bottom layer (X) are the outputs; and the middle layer (Z) is the hidden layer; each layer can have any number of neurons.

probability at any given time (t) is only dependent on the time state immediately before it ($t-1$) (Marsland, 2011). These set of states are linked by the likelihood of moving from the current state to another state, or transition probabilities (Marsland, 2011; Bishop and Nasrabadi, 2006).

As described in (Marsland, 2011), a HMM is made up of three parameters: transition probabilities ($a_{i,j}$), observation probabilities ($b_j(o_k)$) and the probability at starting at any state (π_i). Defining a sequence of probabilities (O) and a sequence of possible states (Ω),

$$P(O) = \sum_{r=1}^{R} P(O|\Omega_r)P(\Omega_r) \tag{1.11}$$

where r describes a sequence of states. Using the Markov property, we can obtain:

$$P(\Omega_r) = \prod_{t=1}^{T} P(\omega_j(t)|\omega_j(t-1)) = \prod_{t=1}^{T} a_{i,j} \tag{1.12}$$

$$P(O|\Omega_r) = \prod_{t=1}^{T} P(o_k(t)|\omega_j(t)) = \prod_{t=1}^{T} (b_j(o_k). \tag{1.13}$$

Following this, equation 1.11 can be rewritten as:

$$P(O) = \sum_{r=1}^{R} \prod_{t=1}^{T} (b_j(o_k) a_{i,j}. \tag{1.14}$$

While this outcome is simplified, it poses a critical issue: assuming there are N hidden states and N^T possible sequences, the computational cost of calculating all T probabilities would be far too great ($O(N^T T)$).

However, as stated previously, the Markov property states that the probability of each state depends only on the current and previous state $(o(t), \omega(t), \omega(t-1))$. This allows for the computation of $P(O)$ to be performed one step at a time. Introducing $\alpha_i(t)$ as the probability that the state is ω_i at time t, and that the first $(t-1)$ steps all matched the observations $o(t)$:

$$\alpha_j(t) = \begin{cases} 0 & t = 0, j \neq initialstate \\ 1 & t = 0, j = initialstate \\ \sum_i \alpha_i(t-1) a_{i,j} b_j(o_t) & otherwise. \end{cases} \tag{1.15}$$

This requires only the probability of observation $o(t)$ to contribute to the sum, allowing the computation of $P(O)$ to be a much improved $O(N^2 T)$. Continuing, if $a_{i,j}$ is the probability of transitioning from state i to state j, $a_{i,t}$ is then the probability of obtaining the observation sequence until a time t and being in the state i at time t.

As this is conditioned on the model and the observations, the probability of moving forward through the entire observation sequence is given as:

$$\sum_{i=1}^{N} \alpha_{i,T}. \tag{1.16}$$

Moving backwards through the sequence requires a similar algorithm based on the transmission and observation probability matrices:

$$\beta_{i,t} = \sum_{j=1}^{N} a_{i,j} b_j(o_{t+1}) \beta_{j,t+1}. \tag{1.17}$$

An alternative use of machine learning in the study of bipedal locomotion is the analysis and estimation of gait parameters using multiple sensors and a variety of machine learning methods (Caldas et al., 2017; Figueiredo et al., 2018). Termed "sensor fusion," this method combines data from multiple sources to produce outputs. Regression is used to estimate outputs such as walking speed (McGinnis et al., 2017) and ground reaction forces (Kieron Jie-Han et al., 2018); classification is the method of choice for gait phase detection (Taborri et al., 2015b, 2014), gait event detection (Novak et al., 2013; Mannini et al., 2014), gait symmetry (Yang et al., 2013) and other spatiotemporal parameters (Shetty and Rao, 2016). This is an emerging research area with positive promise; however, it does suffer from a lack of standardization in both performing and reporting evaluations (Caldas et al., 2017; Figueiredo et al., 2018).

With a large number of possible solutions, it is important to follow a set of guidelines to properly select, apply and evaluate machine learning algorithms to a particular problem as proposed by (Marsland, 2011). First and foremost is data collection and preparation. When asking a new question, it is important to determine what data will be collected; this can be done by collecting a moderate sized data set with features that are believed to be useful and experimenting to find which are of use. In doing this, it is important that the data is clean, meaning it does not have significant errors

or missing data. When using supervised learning, target data is also needed to train the model, often requiring involvement of experts in the field and significant investments of time. Also to be considered is the quantity of data; significant amounts are needed to train an algorithm, but computational costs must also be considered. This step is crucial and can highly affect the potential solutions.

Next, feature selection should be considered. Prior knowledge of the problem and the data is important when identifying features that will be the most useful. After the data set and features have been selected, knowledge of the underlying principles of different types of algorithms and their uses is paramount for selecting the optimum algorithm for the problem. When applicable, manual selection of parameters should result from prior knowledge and experimentation to identify appropriate values. Careful selection of the training data set and parameters must also be considered. Finally, the algorithm must be tested for accuracy on data that it was not trained on; this may include comparisons to human experts in the field.

Authors	Method	Data Types	Outcome
(Tahir and Manap, 2012)	SVM, NN	spatiotemporal, kinetic, kinematic	healthy and pathological
(Kaczmarczyk et al., 2009)	NN	kinematic, qualitative	pathological classification
(Begg and Kamruzzaman, 2005)	SVM	kinetic, kinematic	young and ageing
(Alaqtash et al., 2011)	NN	kinetic	pathological classification
(Mannini et al., 2016)	SVM, HMM	spatiotemporal, kinematic	healthy and pathological gait
(Cuzzolin et al., 2017)	HMM	kinematic	healthy and pathological gait

Table 1.1: Studies using classification, in order of dataset size.

Across the literature (Tables 1.1, 1.2), there are varying practices with regards to the above guideline, as well as various levels of reporting. In the study of bipedal locomotion in healthy and neurologically impaired populations, the large majority of studies use time series data such as saptiotemporal, kinetic and kinematic data (Tables 1.1, 1.2); some also used point metrics of time series data such as maximum and

Authors	Method	Data Types	Sensors
(McGinnis et al., 2017)	SVM	kinematic	accelerometer
(Farah et al., 2017)	SVM	kinetic, kinematic	optical, forceplate
(Taborri et al., 2015b)	HMM	kinematic	IMU
(Taborri et al., 2014)	HMM	kinematic	IMU
(Novak et al., 2013)	SVM	spatiotemporal, kinetic, kinematic	IMU, insole
(Potluri et al., 2019)	SVM, NN	spatiotemporal, kinematic	IMU, insole
(Mannini et al., 2014)	HMM	kinematic	gyroscope
(Kieron Jie-Han et al., 2018)	NN	kinematics	accelerometer

Table 1.2: Studies using sensor fusion, in order of dataset size.

minimum forces and joint angles (Tahir and Manap, 2012; Begg and Kamruzzaman, 2005). All these are most often collected via optical motion tracking, force plates or wearable sensors. Data preparation often takes the form of normalization, such as ground reaction forces normalized to body weight (Alaqtash et al., 2011) and intra and inter group data normalization (Tahir and Manap, 2012). In relevant studies, gait phase segmentation was manually computed to provide added features (McGinnis et al., 2017; Taborri et al., 2015b). When multiple sensors were used, sensor fusion techniques such as Kalman filters (for reducing drift) and Butterworth filters (Novak et al., 2013) were applied.

Many studies do not report how features were extracted or chosen (Begg and Kamruzzaman, 2005; McGinnis et al., 2017); some use content expertise to drive their decisions (Alaqtash et al., 2011); others use methods such as HMMs (Mannini et al., 2016); alternatively, the features at times are chosen "arbitrarily" (Kaczmarczyk et al., 2009). Model selection was similar across both outcomes (Tables 1.1, 1.2): predominantly SVMs, NNs and HMM were used, with other models such as nearest neighbor classifiers, principal component analysis and cluster analysis used as points of comparison.

How the models were trained also varied. A number of studies used cross-validation due to its usefulness with limited data sets (Tahir and Manap, 2012; McGinnis et al., 2017; Taborri et al., 2014; Novak et al., 2013); others used supervised learning approaches to train the models on a subset of data (Mannini et al., 2016). Evaluation of the models were performed by comparing them to a gold standard, that being technology (Begg and Kamruzzaman, 2005; Mannini et al., 2016; Taborri et al., 2015b) or an external standard (Kaczmarczyk et al., 2009).

In the field of bipedal locomotion biomechanics, machine learning is quickly emerging as a hot topic. However, the use of machine learning and wearable sensors to estimate anterior-posterior ground reaction forces - and in turn, propulsion - has not been studied. There remains a need for literature dedicated to studying propulsion and to closing the gap between laboratory-based measurements of propulsion and those accessible to clinicians. Machine learning, combined with wearable sensors, may be able to close that gap.

1.4 Research Questions

The 6-Minute Walk Test (6MWT) is a popular outcome measure used to evaluate long distance walking function after stroke; however, the most often outcome measured, total distance walked, does not directly speak to the underlying neuromotor impairment. In order to promote individualized rehabilitation interventions that can target specific propulsion deficits, the question of what contributes biomechanically to reduced 6MWT performance is necessary. Biomechanical variables such as the peak propulsion force (the peak of the anterior ground reaction force) and the propulsion impulse (the delivered force of the anterior ground reaction force) have been shown to contribute to walking function after stroke (Awad et al., 2015a). However, these magnitude-based metrics do not account for temporal aspects of propulsion function,

such as propulsion peak timing (Kuhman and Hurt, 2019); as they are propulsion-based, they also do not take into consideration braking magnitude and timing. There remains a need to determine if the timing and braking aspects of poststroke walking gait contribute to long-distance walking function.

As such, the first aim of this dissertation is to **evaluate the effects of propulsion metrics, beyond peak propulsion magnitude and propulsion impulse, on long distance walking function**. The hypotheses for this aim are: 1) other propulsion metrics will contribute to the relationship of 6MWT performance, and peak propulsion and propulsion impulse, respectively; and 2) Propulsion timing metrics will moderate the relationship between 6MWT performance, and peak propulsion and propulsion impulse, respectively.

While the study and development of interventions targeting propulsion function after neurological injury is a highly active area of research (Browne and Franz, 2017; Boutaayamou et al., 2015; Miyazaki et al., 2019; Awad et al., 2017b; Genthe et al., 2018; Kesar et al., 2009; McCain et al., 2019; Penke et al., 2019; Phadke, 2012; Takahashi et al., 2015), the clinical use of these approaches is limited due to reduced access to the gold standard technology used to directly measure anterior-posterior ground reaction forces (APGRFs) (Farris et al., 2015; Bowden et al., 2006; Jonkers et al., 2009; Turns et al., 2007; Bethoux and Bennett, 2011; Franz et al., 2014; Franz and Kram, 2013b; Martin et al., 2002). As long as the instruments used to measure propulsion remain inaccessible to clinicians, interventions targeting specific deficits in walking function will continue to be out of reach.

Wearable sensors are a potential solution to this issue: they have been used to collect a number of gait measurements outside the laboratory (Boutaayamou et al., 2015; Miyazaki et al., 2019; Peruzzi et al., 2011; Seel et al., 2014; Yang et al., 2013), and have proven effective at indirect measurements of ground reaction forces during

walking (Karatsidis et al., 2019; Ryu and Park, 2018; Lim et al., 2019; Shahabpoor and Pavic, 2018). Specifically, inertial measurement units (IMUs) have recently shown to be useful in measuring propulsion metrics, such as peak propulsion magnitude and timing, and propulsion impulse, in hemiparetic walking (Pieper et al., 2019; Revi et al., 2020). With the help of machine learning, IMUs may be the key to extending the study of paretic limb propulsion to the clinic.

As such, the second aim of this dissertation is to **validate the accuracy of an IMU based algorithm in estimating propulsion metrics versus the gold standard**. The hypotheses for this aim are: 1) this IMU based machine learning algorithm will accurately estimate all propulsion metrics against the gold standard; and 2) this IMU based machine learning algorithm will outperform the published literature in estimating all propulsion metrics. The final aim will be to **validate the accuracy of an IMU based algorithm in estimating the APGRF time series curve versus the gold standard**. The hypotheses for this aim are: 1) this IMU based machine learning algorithm will accurately estimate the APGRF time series curve versus the gold standard; and 2) this IMU based machine learning algorithm will outperform the published literature in estimating the APGRF time series curve.

Chapter 2

Evaluation of propulsion metrics on long distance walking function

2.1 Introduction

Stroke is a leading cause of disability that results in neuromotor impairments that contribute to slower and more inefficient walking. As a consequence, walking rehabilitation is a major focus after stroke. The 6MWT is a popular outcome measure used to assess functional walking capacity after stroke; however, the performance-based metrics often evaluated, such as the total distance walked, do not identify the underlying neuromotor impairment. Insight into the biomechanical contributors to reduced 6MWT performance is necessary for the advancement of individualized rehabilitation interventions that target specific deficits limiting function.

Biomechanical variables such as peak propulsion (the peak of the anterior ground reaction force) and propulsion impulse (the integral of the anterior ground reaction force) have been shown to contribute to walking function after stroke (Awad et al., 2015a; Alvarez et al., 2020); however, these magnitude-based metrics do not account for key temporal aspects of propulsion function that may affect walking after stroke, such as the timing of peak propulsion (Kuhman and Hurt, 2019). Also, there may be other metrics of the APGRF curve that contribute to walking function after stroke that are not currently considered.

The objective of this study is to determine which APGRF curve metrics most

contribute to 6MWT performance and thus walking function after stroke. In order to reduce bias towards current biomechanical understandings, no metrics will be chosen a priori; instead, a statistical approach will be taken to select those metrics that are the most relevant in explaining 6MWT performance.

2.2 Methods

2.2.1 Participants

Data was collected as part of studies done at the Neuromotor Recovery Laboratory at Boston University. Thirty individuals with poststroke hemiparesis were recruited (Table 2.1). Study inclusion criteria included: being greater than six months after stroke, ambulatory but with residual gait deficits, and having the ability to walk on a treadmill without orthotic support. Study exclusion criteria included: cerebellar stroke, lower extremity joint replacement or other orthopedic conditions that change walking ability, pain that limits walking ability, inability to communicate with investigators, neglect or hemianopia, or unexplained dizziness, and more than two falls in the previous month. All study procedures were approved by the Institutional Review Board of Boston University. Written informed consent was secured from all study participants prior to initiation of study procedures.

As an additional exclusion criteria, participants were excluded if they did not have a propulsion phase, defined as the crossing of the APGRF curve from a negative to positive value, indicating the transition from braking to propulsion, beginning near the midpoint of the stance phase and ending with the termination of contact. Additionally, a minimum of 10 paretic strides of sufficient quality were necessary.

Participant ID	Side of Paresis	Stroke Onset (y)	Sex	Age (y)	Height (cm)	Weight (kg)
01	Right	10	M	39	179.1	77.1
02	Left	8	M	61	179.1	73.7
03	Left	15	M	49	176.8	85.14
04	Right	6	M	61	173.6	98.0
05	Right	6	M	35	183.0	93.2
06	Right	4	M	56	179.5	88.0
07	Left	7	M	78	178.5	102.9
08	Right	2	M	65	171.5	77.1
09	Left	6	M	62	176.0	99.8
10	Right	3	M	62	173.5	85.1
11	Left	1	M	67	184.3	88.5
12	Left	2	M	47	179.8	107.6
13	Left	6	F	52	161.8	45.4
14	Left	5	M	46	178.9	74.4
15	Left	6	M	65	165.5	78.0
16	Left	9	F	42	164.3	66.7
17	Right	11	M	67	165.4	66.7
18	Right	1	M	62	181.8	95.3
19	Right	4	F	60	165.4	66.2
20	Right	4	M	78	173.7	78.7
21	Left	5	M	55	185.7	95.7

Table 2.1: Study 1 Participant Data

2.2.2 Data Collection

Kinematic data were captured by an 18-camera motion capture system (Qualisys, Göteborg, Sweden) based on the motion of reflective markers (Figure 2·1). Single 'landmark' markers were placed to establish anatomy, including: the first and fifth metatarsals; the medial and lateral malleoli; the medial and lateral femoral condyles; the greater trochanters; the anterior superior iliac spines; and the iliac crests. Multiple 'cluster' markers were placed to track the motion of the body segments, including: the pelvis, the upper legs (thighs), and the lower legs (shanks); two markers were also placed on the heel to create a cluster with the metatarsal markers. All markers were sampled at a rate of 200 Hz.

Kinetic data were captured using 6 inground force platforms, each with 6-degrees of freedom (Bertec Corporation, Worthington, OH), placed on a 10m strait and sampling at 2000 Hz. At the start of each visit, participants completed a standing static trial on top for the inground force plates to record their body mass and calibrate the motion capture system. The primary clinical outcome was total distance walked during the 6MWT, conducted by licensed physical therapists, performed around a

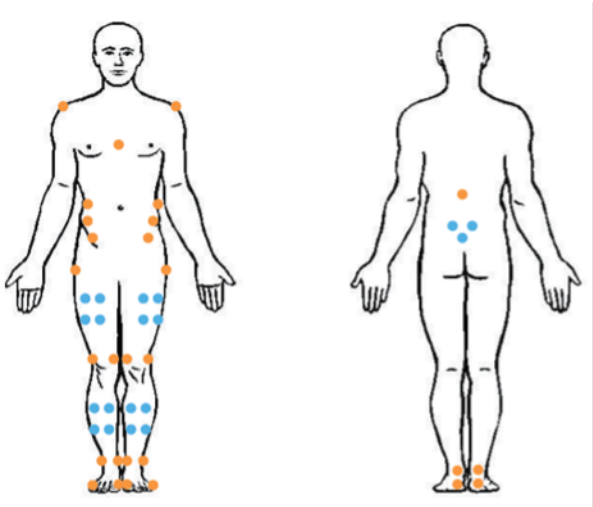


Figure 2·1: Motion Capture Marker Placement

26.6m oval indoor track consisting of two 10m straight with 3.3m turns on either end.

2.2.3 Data Processing

Marker motion capture data were cleaned in Qualysis Track Manager; marker and force plate data were filtered in Visual3D using a bi-directional Butterworth low pass filter at 10 Hz, then entered into MATLAB for further processing. Using MATLAB 2019a (Mathworks), strides were combined across all passes to create the data sets for each participant. Each stride was normalized to percent body weight (%BW) as per the literature, and percent stance time (%SP, defined as the duration of the stance phase, from initial contact to final contact). From this, the following metrics were calculated (Table 2.2).

2.2.4 Statistical Analysis

Using MATLAB 2019a, bivariate relationships between the propulsion metrics and 6MWT distance were evaluated. Using the 'stepwiselm' function in MATLAB, a forward stepwise regression was preformed: an F test was performed between the metrics and 6MWT, and the metric with the largest F-score was added to the model

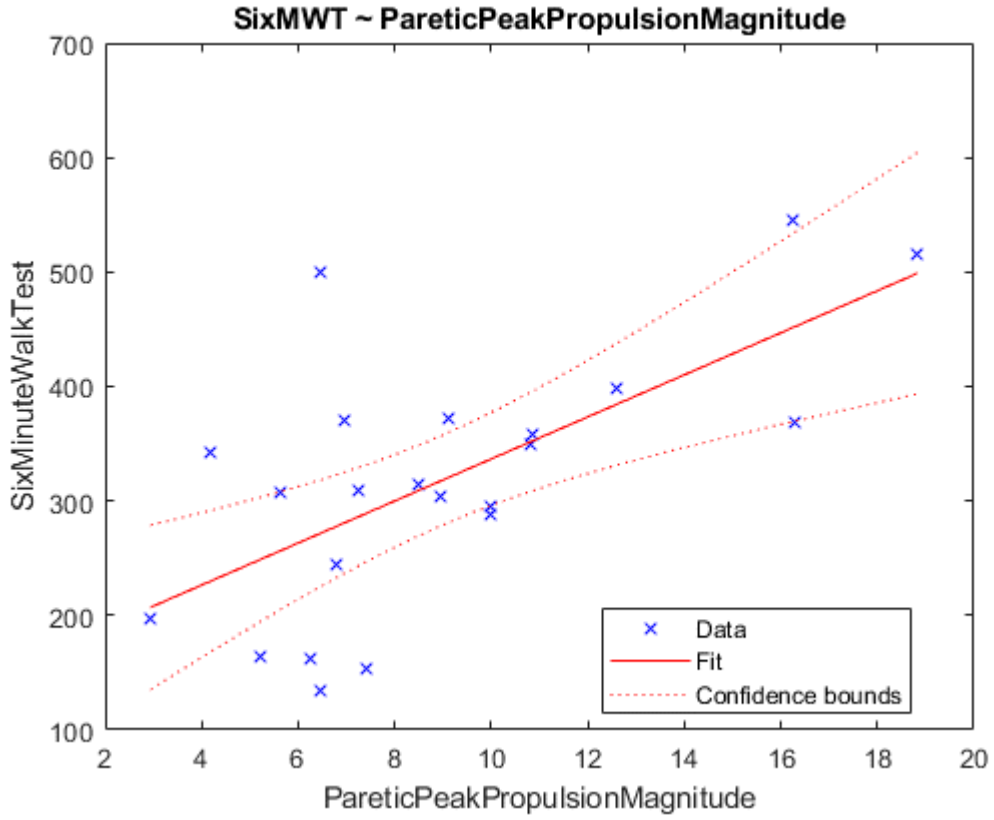
Metric	Definition	Units
Braking Onset	Beginning of braking phase	%SP
Peaking Magnitude	Maximum braking force	%BW
Peak Timing	Timing of peak magnitude	%SP
Impulse	Total braking force	-
Total Time	Time spent in braking phase	%SP
Propulsion Onset	Beginning of propulsion phase	%SP
Peaking Magnitude	Maximum propulsion force	%BW
Peak Timing	Timing of peak magnitude	%SP
Impulse	Total propulsion force	-
Total Time	Time spent in propulsion phase	%SP

Table 2.2: Calculated Propulsion Metrics

– provided that it's p-value was below 0.05. The F tests were then repeated on all the remaining metrics with this resulting model, adding metrics only when their inclusion was measured as significant.

The above was performed three separate times with different sets of metrics: only propulsion metrics from the paretic limb; propulsion and braking metrics from the paretic limb; and propulsion and braking metrics from both the paretic and nonparetic limb. The resulting models were then graphed to visualize their fit with the data.

2.3 Results



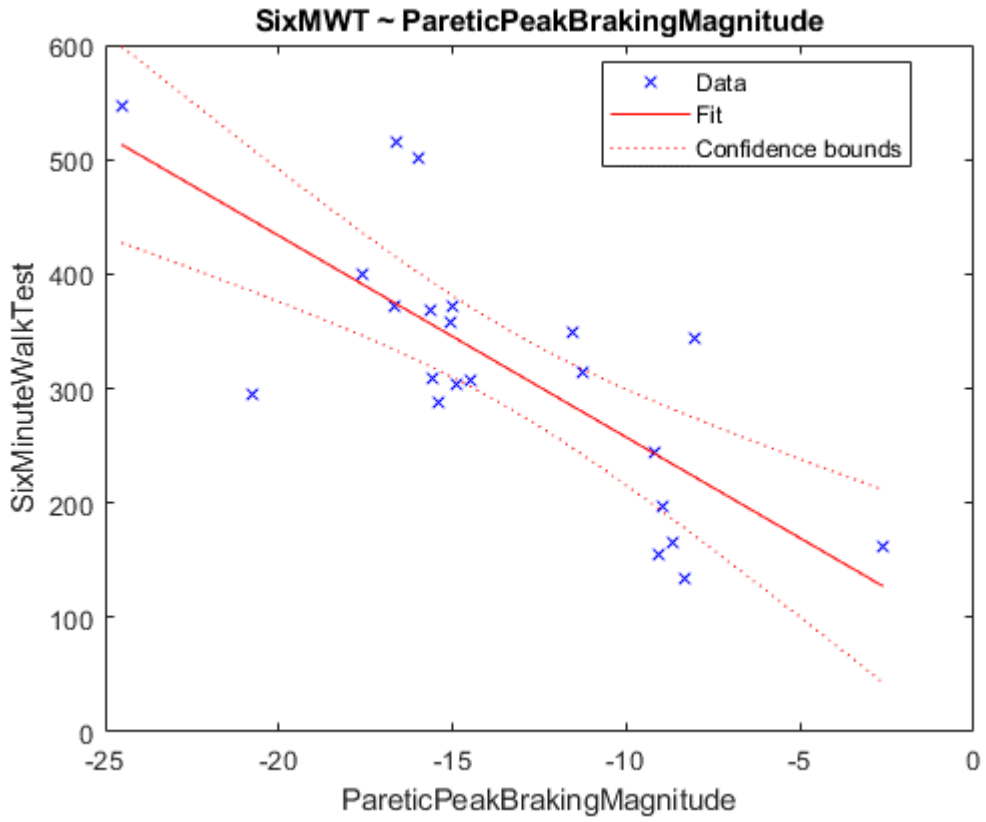
	Estimate	SE	tStat	p-value
Intercept	153.04	46.79	3.27	<0.01
PareticPeakPropulsionMagnitude	18.37	4.77	3.85	<0.01

Model	R^2	$_{adj}R^2$	RMSE	
1	0.43	-	88.45	

Figure 2·2: 6MWT distance as predicted by Peak Propulsion Magnitude of paretic limb.

The first model was performed using only propulsion metrics from the paretic limb. The stepwise model deemed the peak propulsion magnitude as the most important metric, as it alone explains 43% of the variance in 6MWT performance. No other metrics were added to this model, as it was determined that their addition would not

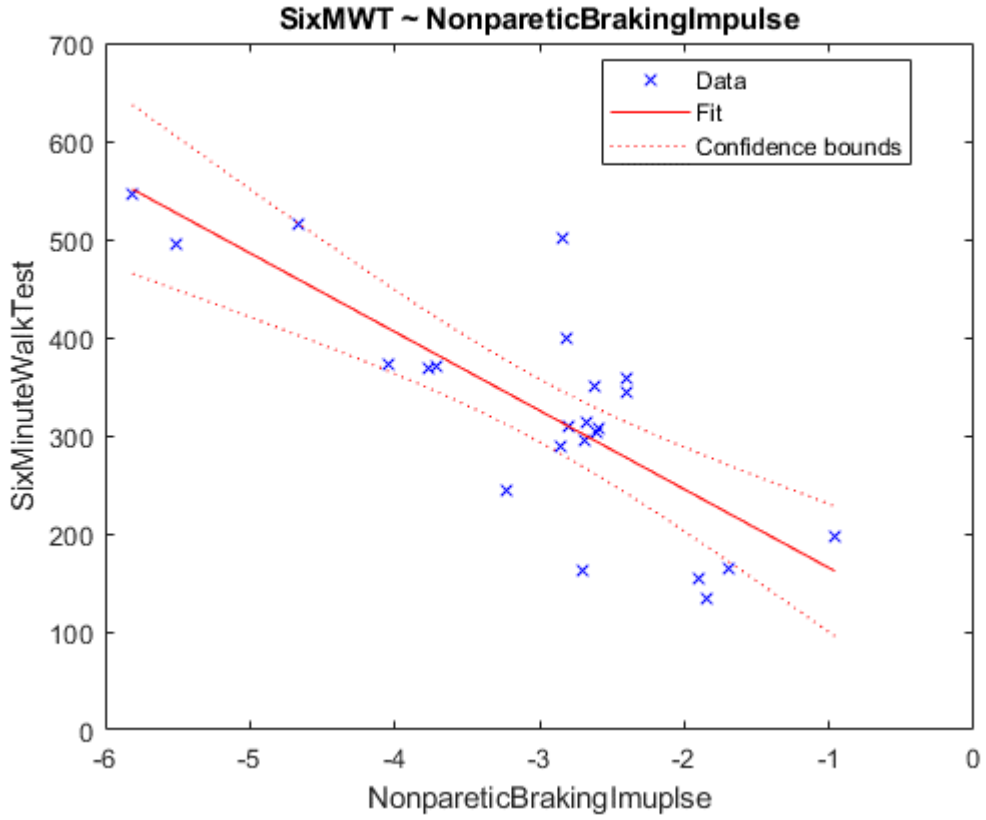
be significant.

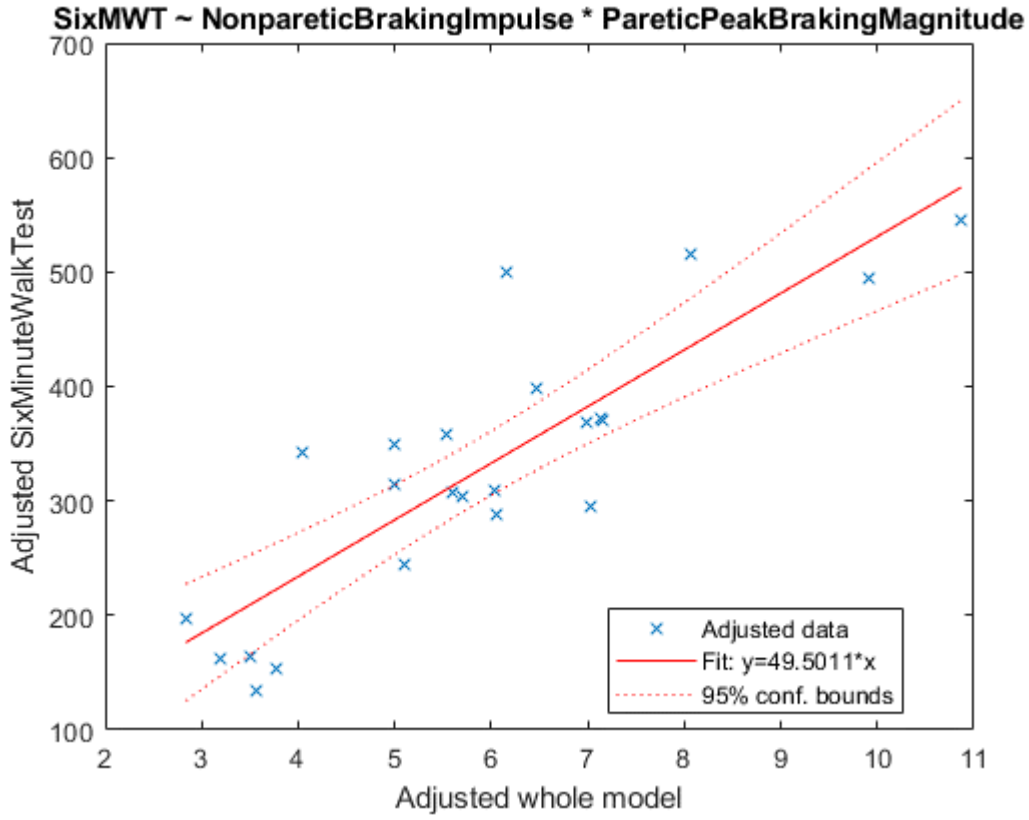


	Estimate	SE	tStat	p-value
Intercept	75.90	46.32	1.64	0.12
PareticPeakBrakingMagnitude	-18.11	3.16	-5.73	<0.01
Model	R^2	${}_{adj}R^2$	RMSE	
1	0.61	-	74.97	

Figure 2·3: 6MWT distance as predicted by Peak Braking Magnitude of paretic limb.

For the second model, both propulsion and braking metrics of the paretic limb were allowed to be chosen. The stepwise model deemed the peak braking magnitude as the most important metrics, as it alone explains 61% of the variance in 6MWT performance. No other metrics were added to this model, as it was determined that their addition would not be significant.


SixMWT ~ NonpareticBrakingImpulse
Data
Fit
Confidence bounds
SixMinuteWalkTest
NonpareticBrakingImuplse
700
600
500
400
300
200
100
0
-6
-5
-4
-3
-2
-1
0



	Estimate	SE	tStat	p-value
Intercept	84.75	43.85	1.93	0.07
NonpareticBrakingImpulse	-80.24	13.67	-5.87	<0.01

	Estimate	SE	tStat	p-value
Intercept	35.97	41.95	0.86	0.40
NonpareticBrakingImpulse	-10.47	3.76	-2.79	0.01
PareticPeakBrakingMagnitude	-48.38	16.49	-2.94	0.01

Model	R^2	$_{adj}R^2$	RMSE	
1	0.62	-	73.86	
2	0.73	0.70	64.22	

Figure 2·4: 6MWT distance as predicted by interaction between Peak Braking Magnitude of the paretic limb and Braking Impulse of the nonparetic limb.

For the final model, propulsion and braking metrics of both the paretic and nonparetic limb were allowed to be chosen. The stepwise model first selected the nonparetic braking impulse, as it alone explains 62% of the variance in 6MWT perfor-

mance. Next, the model selected paretic peak braking magnitude, as its addition increased the model's ability to explain the variance in 6MWT performance to 70%. No other metrics were added to this model, as it was determined that their addition would not be significant.

2.4 Discussion

The goal of this study was to allow a statistical method, particularly a stepwise linear regression model, to select the most significant features of the APGRF curve that were most significant in explaining 6MWT distance. This would allow human bias to be as removed as possible, and potentially statistically validating the prevailing literature claiming that paretic propulsion is one of the most important factors in long distance walking function poststroke.

The results of this study show that, when limiting the choices to those of propulsion metrics in the paretic limb, a stepwise linear regression model selects peak propulsion magnitude as the most significant variable and does not include another. This follows previous work in the literature, though differs slightly from preliminary work of this study (Alvarez et al., 2020). In said work, peak propulsion magnitude explained less of the variance than in the current work ((R^2) = 0.37 vs 0.43); however, including peak propulsion timing and its interaction with peak propulsion force lifts it above the results of the current work.

Interestingly, when allowing braking metrics to be selected, the stepwise linear regression model found that peak braking magnitude was superior at explaining the variance than any propulsion metric. In fact, peak propulsion braking magnitude alone performed better (R^2 = 0.61) than the combined model of peak propulsion magnitude and propulsion timing found in the previous study. Further, when allowing selection of propulsion and braking metrics from either limb, the stepwise

linear regression model found that nonparetic braking impulse as well as paretic peak braking magnitude offered the best explanation of the variance ($R^2 = 0.70$).

This differs greatly not only from the preliminary work, but from the majority of the literature, which focuses primarily on propulsion of the paretic limb. This is understandable, as the biomechanics of walking, as currently understood, relies on the idea that it is the propulsion of the trailing limb that moves the body forward. While this likely remains true, this work shows that braking must also be addressed. This follows biomechanically: if one were to increase their propulsion without also increasing their ability to brake, they would likely fall; they must then reduce their propulsion to be more in line with their braking ability. Addressing both propulsion production and braking ability may then lead to greater rehabilitation outcomes.

To strengthen the conclusion of this study, more data should be examined. Also, as the population and walking conditions examined are of a specific kind, the results of these studies may not be suitable for consideration in other populations or in lower-level walkers in the poststroke population. Finally, while allowing for the most prominent features to be selected via a statistical method can reduce human bias, the features were derived by a human; therefore, it should not be assumed that these results are completely free from bias.

Chapter 3

Validation of wearable sensors for use in estimating poststroke propulsion in the clinic

3.1 Introduction

Stroke is a leading cause of disability that results in neuromotor impairments that contribute to slower and more inefficient walking. As a consequence, walking rehabilitation is a major focus after stroke, particularly propulsion function. While the study and development of interventions targeting propulsion function after neurological injury is a highly active area of research (Browne and Franz, 2017; Boutaayamou et al., 2015; Miyazaki et al., 2019; Awad et al., 2017a; Genthe et al., 2018; Kesar et al., 2009; McCain et al., 2019; Penke et al., 2019; Phadke, 2012; Takahashi et al., 2015), the clinical use of these approaches is limited due to reduced access to the gold standard technology used to directly measure APGRFs (Farris et al., 2015; Bowden et al., 2006; Jonkers et al., 2009; Turns et al., 2007; Bethoux and Bennett, 2011; Franz and Kram, 2013a, 2014; Martin et al., 2002). As long as the instruments used to measure propulsion remain inaccessible to clinicians, interventions targeting specific deficits in walking function will continue to be out of reach.

Wearable sensors are a potential solution to this issue: they have been used to collect a number of gait measurements outside the laboratory (Boutaayamou et al., 2015; Miyazaki et al., 2019; Peruzzi et al., 2011; Seel et al., 2012; Yang et al., 2013),

and have proven effective at indirect measurements of ground reaction forces during walking (Karatsidis et al., 2016; Ryu and Park, 2018; Lim et al., 2016; Shahabpoor and Pavic, 2018). Specifically, IMUs have shown to be useful in measuring propulsion metrics, such as peak propulsion magnitude and timing, and propulsion impulse, in hemiparetic walking (Pieper et al., 2019; Revi et al., 2020). With the help of machine learning, IMUs may be the key to extending the study of paretic limb propulsion to the clinic.

The objective of this study is to validate the accuracy of an IMU based algorithm in estimating propulsion metrics versus the gold standard technology. This study will also work to validate the accuracy of an IMU based algorithm in estimating the APGRF time series curve versus the gold standard technology. In order to reduce bias towards current biomechanical understanding, no features will be chosen a priori; instead, a statistical approach will be taken to select those features that are most relevant in estimating propulsion metrics and the APGRF time series curve.

3.2 Methods

3.2.1 Participants

Data was collected as part of a study at the Neuromotor Recovery Laboratory at Boston University. Seven individuals with chronic poststroke hemiparesis participated (Table 3.1). Study inclusion criteria included: being greater than six months after stroke, ambulatory but with residual gait deficits, and having the ability to walk on a treadmill without orthotic support. Study exclusion criteria included: cerebellar stroke, lower extremity joint replacement or other orthopedic conditions that change walking ability, pain that limits walking ability, inability to communicate with investigators, neglect or hemianopia, or unexplained dizziness, and more than two falls in the previous month. All study procedures were approved by the Institutional Review

Board of Boston University. Written informed consent was secured from all study participants prior to initiation of study procedures.

Participant ID	Side of Paresis	Stroke Onset (y)	Sex	Age (y)	Height (cm)	Weight (kg)
1	Right	1	Male	62	181.5	94.5
2	Left	9	Male	80	179.5	102.7
3	Right	8	Male	47	181.7	100.2
4	Right	5	Male	65	173.5	81.7
5	Left	2	Male	48	180.3	111.4
6	Left	2	Male	47	177.4	92.3
7	Left	3	Male	59	187.0	89.13

Table 3.1: Study 2 Participant Data

3.2.2 Data Collection

Kinematic data were captured by an 18-camera motion capture system (Qualisys, Göteborg, Sweden) based on the motion of reflective markers (Figure 3·1). Single 'landmark' markers were placed to establish anatomy, including: the first and fifth metatarsals; the medial and lateral malleoli; the medial and lateral femoral condyles; the greater trochanters; the anterior superior iliac spines; and the iliac crests. Multiple 'cluster' markers were placed to track the motion of the body segments, including: the pelvis, the upper legs (thighs), and the lower legs (shanks); two markers will also be placed on the heel to create a cluster with the metatarsal markers. All markers were sampled at a rate of 200 Hz.

Participants performed 10-Meter Walk Tests, conducted by licensed physical therapists, to obtain their self-selected walking speed (i.e. comfortable walking speed, CWS). Kinetic data was collected during two separate 2-minute bouts of treadmill walking at 80% of their CWS, as per study protocols. Ground reaction forces (GRFs) were collected using a dual-belt instrumented treadmill with 2 independent 6-degree of freedom force platforms (Bertec Corporation, Worthington, OH) sampling at 2000 Hz. A total of 7 IMUs (Xsens Technologies B.V., Enschede, Netherlands), sampling at 100 Hz, were collected concurrently during the bouts of treadmill walking. They were placed on the pelvis, as well as bilaterally on the thigh, shank and foot seg-

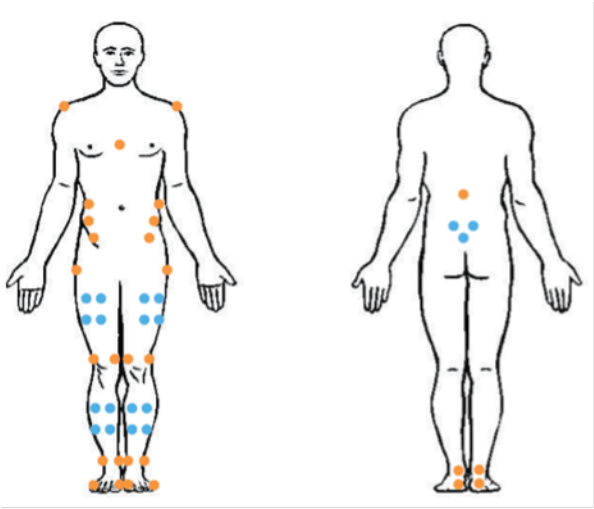


Figure 3·1: Motion Capture Marker Placement

ments (Figure 3·2). The IMUs were placed so that their axes matched that of the motion capture space: medial/lateral movement was on the X axis, anterior/posterior movement on the Y axis and vertical movement on the Z axis.

3.2.3 Data Processing

Marker motion capture data were cleaned in Qualysis Track Manager; marker and force plate data were filtered in Visual3D using a bi-directional Butterworth low pass filter at 30 Hz, then entered into MATLAB for further processing. Using MATLAB 2019a (Mathworks, Natick, MA), the total number of strides per participant were combined across both walking bouts. Each stride will be normalized to percent body weight (%BW) as per the literature, and percent stance phase (%SP, defined as the duration of the stance phase, from initial contact to final contact); the latter decision was made to better focus future analysis on the effects of propulsion, regardless of other factors in the gait cycle, as well as its common use by clinicians. Propulsion metrics found to be significant as part of Study 1 were also calculated. The IMU data were cleaned and filtered in MATLAB, then conducted into features for use in the machine learning algorithm (Table 3.2). For the first aim, the inputs to the

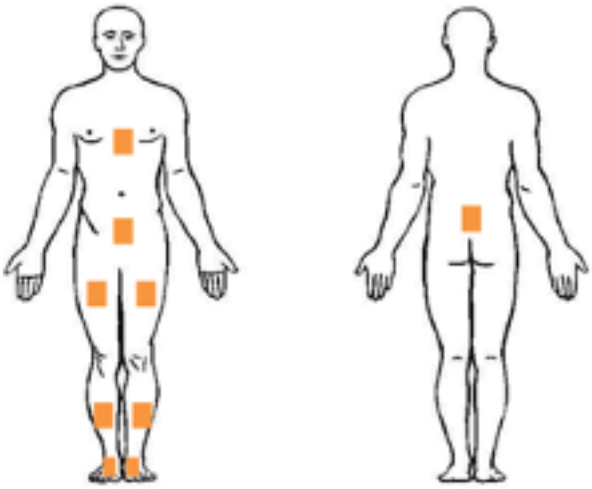


Figure 3·2: IMU Placement

model were the features listed in Table 3.2; the response variable will be the metrics calculated in the first study (2.2). For the second aim, the inputs to the model will be the features listed in Table 3.2; the response variable will be the APGRF time series curve taken from the gold standard, i.e. the instrumented treadmill.

Source	Definition	Directions	Features
Accelerometer	Linear acceleration	X, Y, Z	3
Gyroscope	Angular velocity	X, Y, Z	3

Table 3.2: IMU Features

The remaining work was done in Python 3.13.2 (Python Software Foundation).

Feature ranking examined the importance of each predictor individually by using a LOO cross-validation method. This approach evaluated performance of the model with all features included; features were then iteratively removed one at a time, the model was retrained and its performance was evaluated on a validation set. This performance was measured using RMSE, the average of the squared difference between predictions and the ground truth. The model chosen for the propulsion metrics analysis was a simple linear regression model (LinearRegression) from the popular machine learning package scikit-learn 1.6.1 (Pedregosa et al., 2011). The three most impor-

tant features were then included in the training model. To train the propulsion metric model, a LOO cross-validation method was used. The mean and standard deviation across all folds were reported for the estimation of the propulsion metric, as well as for the evaluation metrics.

The above LOO cross-validation model was also used to choose the features included in the APGRF model, replacing the simple linear regression model with a stochastic gradient descent model (SGDRegressor), also from scikit-learn. The APGRF model was trained using the SGDRegressor model's *partial_fit* attribute; this allowed the model to be trained one stride at a time, allowing it to learn and correct from previous errors. This was done on a training set made up of 70% of the data; the remaining 30% were used as part of the testing set to evaluate the model.

3.2.4

To determine the performance of the propulsion metrics model, root mean squared error (RMSE) and mean absolute percent error (MAPE) were used; these, as well as R^2, were used to measure the performance of the APGRF model. RMSE measures the average of the squared difference between the predictions and the ground truth, allowing for the detection of outliers; MAPE is the absolute value of the percent difference between the predictions and the ground truth; R^2 uses the regression and total sum of squares to determine the variance between the model and a linear model.

For the first model, propulsion metrics taken from the gold standard of force plate data were used as the response variable. For the second model, the APGRF time series curve taken from the gold standard of force plate data will be used as the response variable. The models were then compared to those in the literature, with performance metrics as well as features used to construct the models being analyzed.

3.3 Results

3.3.1 Propulsion Metrics Model

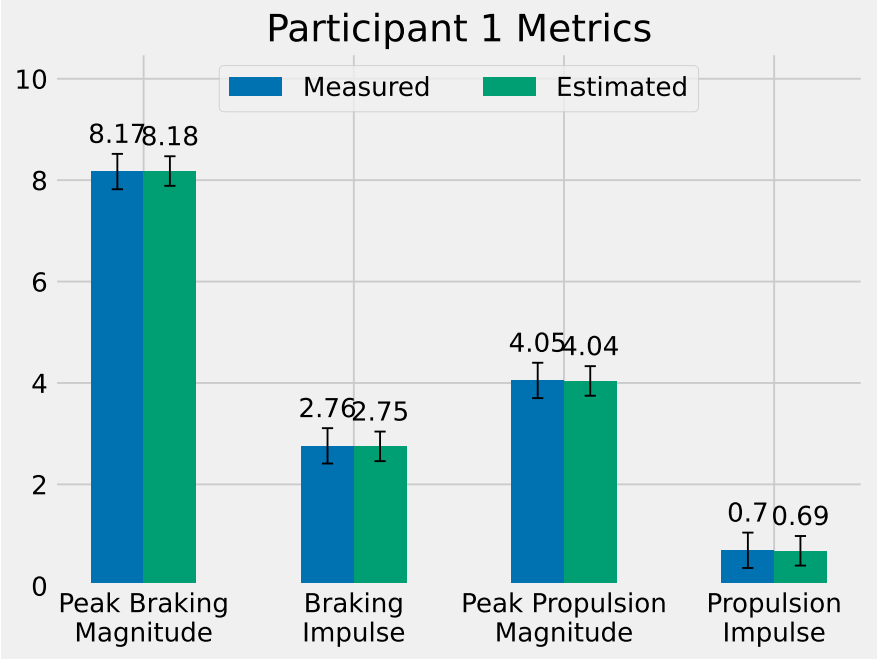


Figure 3.3: Results of propulsion metrics model for Participant 1.

Peak Braking Magnitude:
Foot Acc X, Thigh Gyro Z, Thigh Gyro X

	Mean	Std
Measured	-8.17	0.70
Estimate	-8.18	0.29
MAPE	0.07	0.06
RMSE	-0.57	0.47

Peak Propulsion Magnitude:
Thigh Acc Y, Shank Gyro X, Foot Acc X

	Mean	Std
Measured	4.05	0.70
Estimate	4.04	0.26
MAPE	0.16	0.09
RMSE	-0.63	0.40

Braking Impulse:
Thigh Gyro Z, Foot Acc X, Foot Gyro X

	Mean	Std
Measured	-2.76	0.22
Estimate	-2.75	0.05
MAPE	0.07	0.06
RMSE	-0.18	0.15

Propulsion Impulse:
Thigh Acc Y, Shank Acc Z, Shank Gyro X

	Mean	Std
Measured	0.70	0.16
Estimate	0.69	0.07
MAPE	0.19	0.14
RMSE	-0.13	0.10

Table 3.3: Propulsion metric model results for Participant 1.

Executing the feature selection for Participant 1, the three most important features to estimate peak braking magnitude were foot acceleration in the X direction, thigh gyroscopic force in the Z direction and thigh gyroscopic force in the X direction, resulting in a MAPE of 7% and a RMSE of -0.57. For braking impulse, they were thigh gyroscopic force in the Z direction, foot acceleration in the X direction and foot gyroscopic force in the X direction, resulting in a MAPE of 7% and a RMSE of -0.18. For peak propulsion magnitude, they were thigh acceleration in the Y direction, shank gyroscopic force in the X direction and foot acceleration in the X direction, resulting in a MAPE of 16% and a RMSE of -0.63. For propulsion impulse, they were thigh acceleration in the Y direction, shank acceleration in the Z direction and shank gyroscopic force in the X direction, resulting in a MAPE of 19% and a RMSE of -0.13.

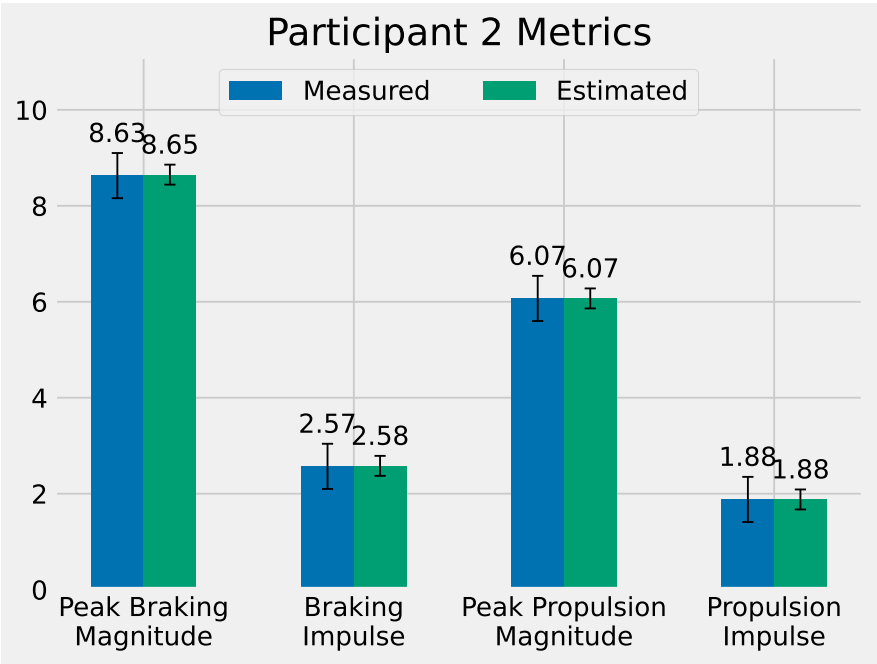


Figure 3·4: Results of propulsion metrics model for Participant 2.

[H]

Peak Braking Magnitude:
Thigh Gyro X, Thigh Gyro Y, Shank Gyro Z

	Mean	Std
Measured	-8.63	0.94
Estimate	-8.65	0.21
MAPE	0.09	0.09
RMSE	-0.77	0.71

Braking Impulse:
Thigh Gyro X, Thigh Gyro Y, Shank Gyro Z

	Mean	Std
Measured	-2.57	0.40
Estimate	-2.58	0.12
MAPE	0.14	0.14
RMSE	-0.33	0.30

Peak Propulsion Magnitude:
Shank Acc Y, Foot Acc X, Thigh Acc Y

	Mean	Std
Measured	6.07	0.97
Estimate	6.07	0.36
MAPE	0.13	0.10
RMSE	-0.79	0.59

Propulsion Impulse:
Shank Acc Y, Shank Gyro Z, Foot Gyro Y

	Mean	Std
Measured	1.88	0.37
Estimate	1.88	0.18
MAPE	0.16	0.13
RMSE	-0.28	0.21

Table 3.4: Propulsion metric model results for Participant 2.

Executing the feature selection for Participant 2, the three most important features to estimate peak braking magnitude were thigh gyroscopic force in the X direction, thigh gyroscopic force in the Y direction and shank gyroscopic force in the Z direction, resulting in a MAPE of 9% and a RMSE of -0.77. For braking impulse, they were thigh gyroscopic force in the Y direction, thigh gyroscopic force in the X direction and shank gyroscopic force in the Z direction, resulting in a MAPE of 14% and a RMSE of -0.33. For peak propulsion magnitude, they were shank acceleration in the Y direction, foot acceleration in the X direction and thigh acceleration in the Y direction, resulting in a MAPE of 13% and a RMSE of -0.79. For propulsion impulse, they were shank acceleration in the Y direction, shank gyroscopic force in the Z direction and foot gyroscopic force in the Y direction, resulting in a MAPE of 16% and a RMSE of -0.28.

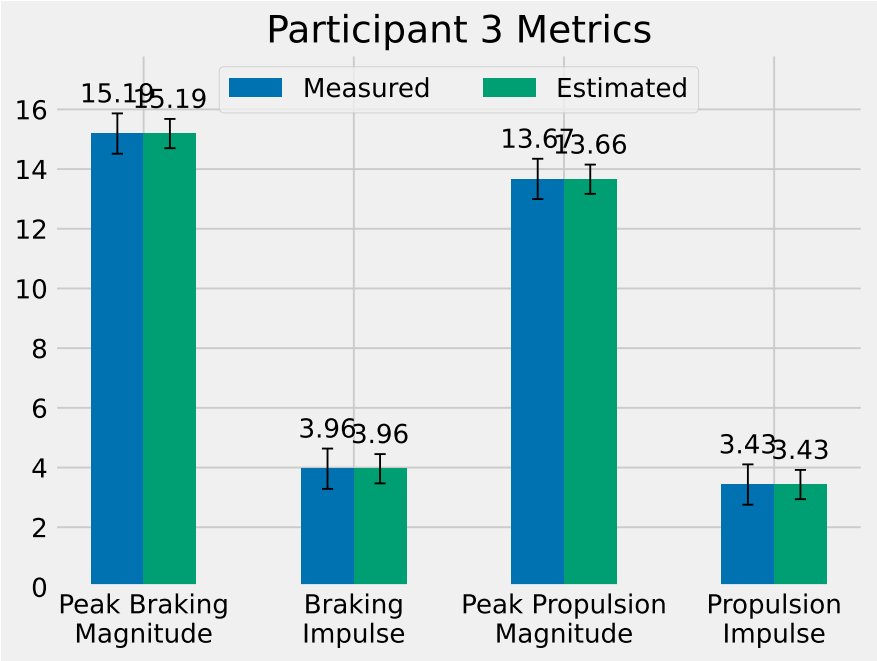


Figure 3·5: Results of propulsion metrics model for Participant 3.

Peak Braking Magnitude:
Shank Acc Y, Shank Acc X, Thigh Acc Y

	Mean	Std
Measured	-15.19	1.35
Estimate	-15.19	0.49
MAPE	0.07	0.06
RMSE	-1.06	0.81

Braking Impulse:
Shank Acc Y, Shank Gyro X, Foot Acc Y

	Mean	Std
Measured	-3.96	0.37
Estimate	-3.96	0.09
MAPE	0.08	0.07
RMSE	-0.29	0.24

Peak Propulsion Magnitude:
Thigh Gyro X, Shank Gyro Y, Foot Gyro Y

	Mean	Std
Measured	13.67	1.04
Estimate	13.66	0.19
MAPE	0.06	0.05
RMSE	-0.85	0.65

Propulsion Impulse:
Shank Gyro Y, Foot Gyro Y, Thigh Gyro Y

	Mean	Std
Measured	3.43	0.25
Estimate	3.43	0.06
MAPE	0.06	0.06
RMSE	-0.19	0.17

Table 3.5: Propulsion metric model results for Participant 3.

Executing the feature selection for Participant 3, the three most important features to estimate peak braking magnitude were shank acceleration in the Y direction, shank

acceleration in the X direction and thigh acceleration in the Y direction, resulting in a MAPE of 7% and a RMSE of -1.06. For braking impulse, they were shank acceleration in the Y direction, shank gyroscopic force in the X direction and foot acceleration in the Y direction, resulting in a MAPE of 8% and a RMSE of -0.29. For peak propulsion magnitude, they were thigh gyroscopic force in the X direction, shank gyroscopic force in the Y direction and foot gyroscopic force in the Y direction, resulting in a MAPE of 6% and a RMSE of -0.85. For propulsion impulse, they were shank gyroscopic force in the Y direction, foot gyroscopic force in the Y direction and thigh gyroscopic force in the Y direction, resulting in a MAPE of 6% and a RMSE of -0.19.

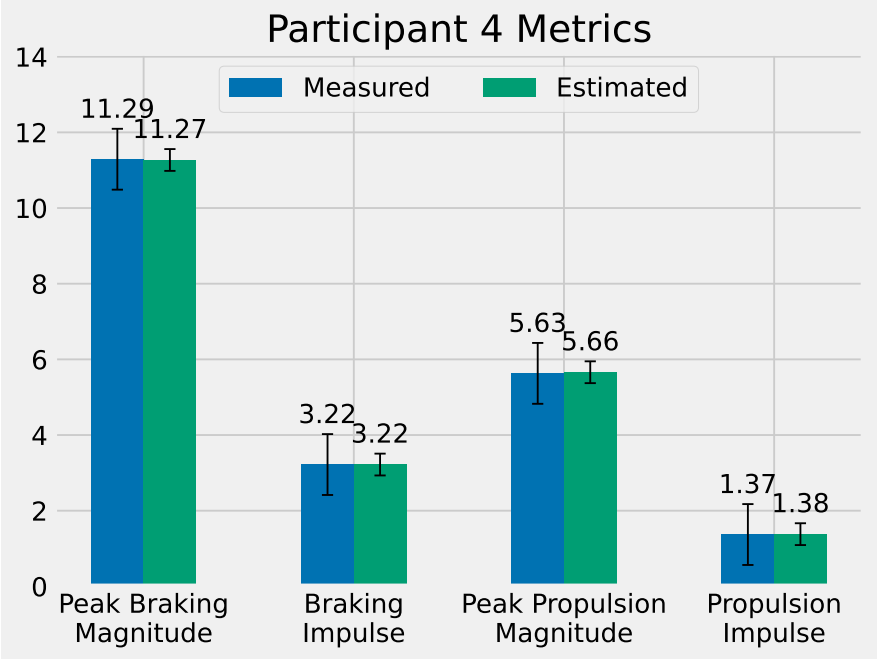


Figure 3·6: Results of propulsion metrics model for Participant 4.

Peak Braking Magnitude:
Thigh Gyro X, Shank Gyro Z, Foot Gyro Y

	Mean	Std
Measured	-11.29	1.61
Estimate	-11.27	0.29
MAPE	0.13	0.12
RMSE	-1.39	1.03

Braking Impulse:
Thigh Gyro Y, Thigh Acc Z, Shank Acc Z

	Mean	Std
Measured	-3.22	0.45
Estimate	-3.22	0.15
MAPE	0.12	0.13
RMSE	-0.37	0.30

Peak Propulsion Magnitude:
Thigh Gyro Y, Shank Acc X, Shank Gyro Z

	Mean	Std
Measured	5.63	1.17
Estimate	6.29	10.39
MAPE	0.20	0.17
RMSE	-1.05	0.73

Propulsion Impulse:
Thigh Gyro Y, Shank Acc X, Shank Gyro Y

	Mean	Std
Measured	1.37	0.47
Estimate	1.38	0.18
MAPE	0.36	0.31
RMSE	-0.42	0.29

Table 3.6: Propulsion metric model results for Participant 4.

Executing the feature selection for Participant 4, the three most important features to estimate peak braking magnitude were thigh gyroscopic force in the X direction, shank gyroscopic force in the Z direction and foot gyroscopic force in the Y direction, resulting in a MAPE of 13% and a RMSE of -1.39. For braking impulse, they were thigh gyroscopic force in the Y direction, thigh acceleration in the Z direction and shank acceleration in the Z direction, resulting in a MAPE of 12% and a RMSE of -0.37. For peak propulsion magnitude, they were thigh gyroscopic force in the Y direction, shank acceleration in the X direction and shank gyroscopic force in the Z direction, resulting in a MAPE of 20% and a RMSE of -1.05. For propulsion magnitude, they were thigh gyroscopic force in the Y direction, shank acceleration in the X direction and shank gyroscopic force in the Y direction, resulting in a MAPE of 36% and a RMSE of -0.42.

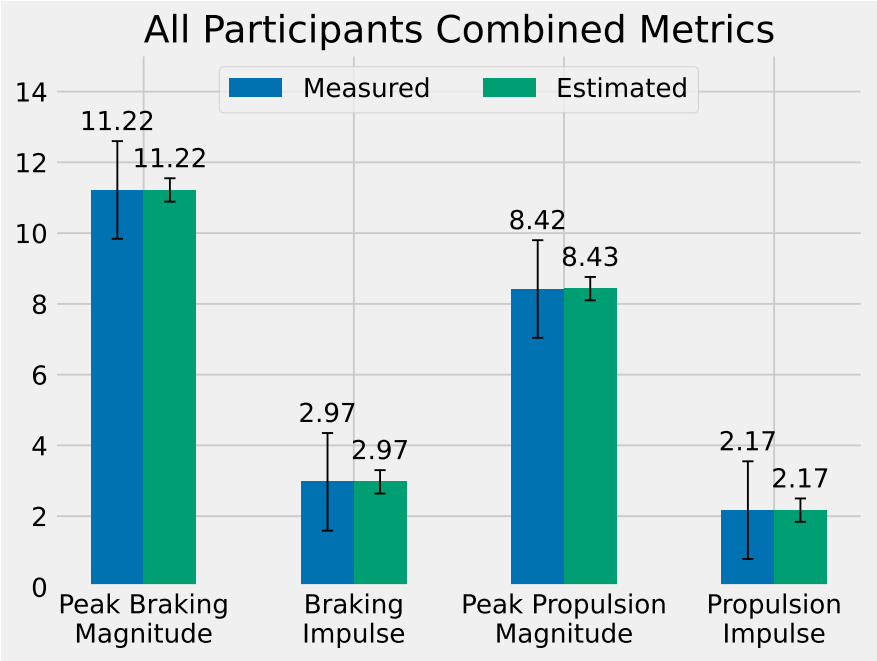


Figure 3·7: Results of propulsion metrics model for all participants.

Peak Braking Magnitude:
Shank Gyro Y, Thigh Acc X, Foot Gyro Y

	Mean	Std
Measured	-11.22	2.76
Estimate	-11.22	0.33
MAPE	0.22	0.16
RMSE	-2.33	1.49

Braking Impulse:
Thigh Gyro X, Foot Acc Y, Thigh Acc X

	Mean	Std
Measured	-2.97	0.65
Estimate	-2.97	0.39
MAPE	0.15	0.12
RMSE	-0.43	0.30

Peak Propulsion Magnitude:
Thigh Acc Y, Foot Gyro X, Shank Acc Y

	Mean	Std
Measured	8.42	3.71
Estimate	8.43	1.77
MAPE	0.43	0.37
RMSE	-2.90	1.71

Propulsion Impulse:
Thigh Acc Y, Foot Gyro X, Foot Acc Y

	Mean	Std
Measured	2.17	1.02
Estimate	2.17	0.59
MAPE	0.49	0.48
RMSE	-0.74	0.42

Table 3.7: Propulsion metric model results for all participants.

Executing the feature selection for the combination of all participants, the three most important features to estimate peak braking magnitude were shank gyroscopic

force in the Y direction, thigh acceleration in the X direction and foot gyroscopic force in the Y direction, resulting in a MAPE of 22% and a RMSE of -2.33. For braking impulse, they were thigh gyroscopic force in the X direction, foot acceleration in the Y direction and thigh acceleration in the X direction, resulting in a MAPE of 15% and a RMSE of -0.43. For peak propulsion magnitude, they were thigh acceleration in the Y direction, foot gyroscopic force in the X direction and shank acceleration in the Y direction, resulting in a MAPE of 43% and a RMSE of -2.90. For propulsion impulse, they were thigh acceleration in the Y direction, foot gyroscopic force in the X direction and foot acceleration in the Y direction, resulting in a MAPE of 49% and a RMSE of -0.74.

3.3.2 APGRF Model

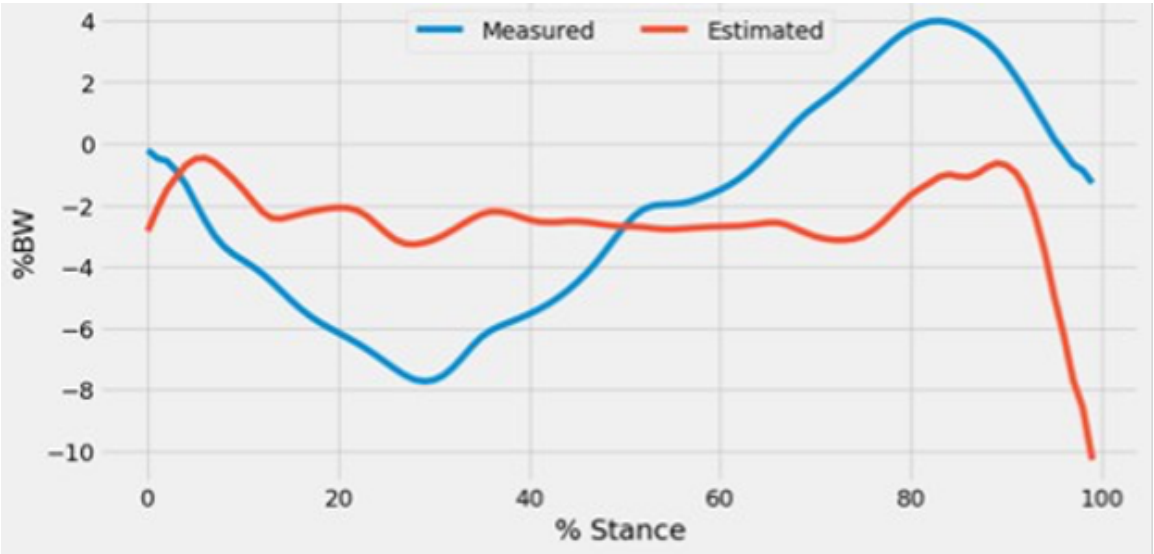


Figure 3·8: Results of APGRF model for Participant 1.

When including all features in the estimation of the APGRF for Participant 1, the resulting MAPE was 130%, the RMSE as 2.54 and the amount of variance explained was 50%.

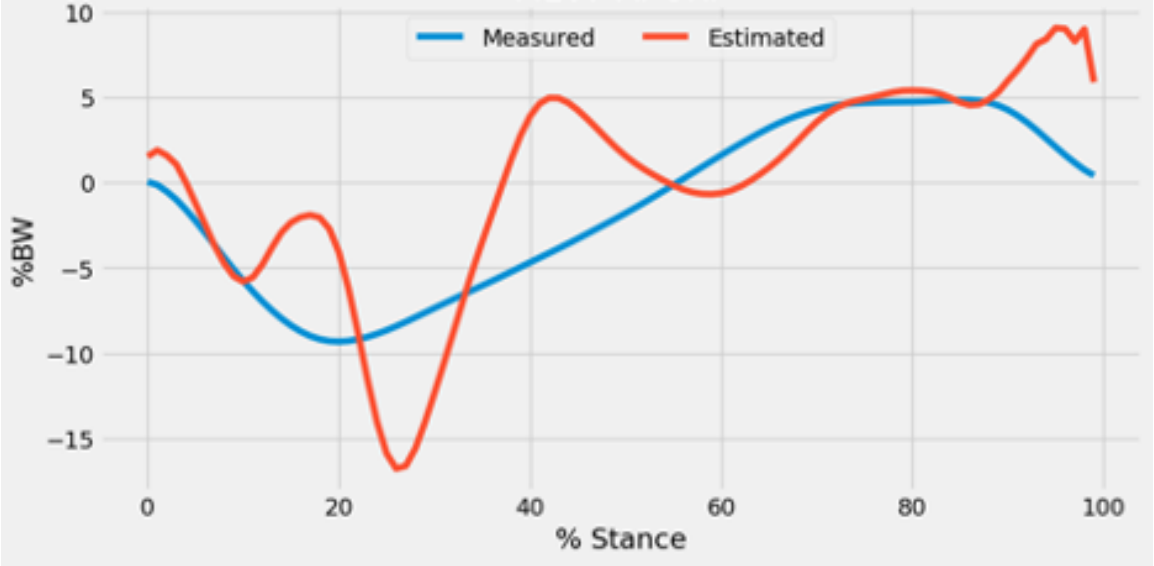


Figure 3·9: Results of APGRF model for Participant 2.

When including all features in the estimation of the APGRF for Participant 2, the resulting MAPE was 121%, the RMSE was 1.87 and the amount of variance explained was 85%.

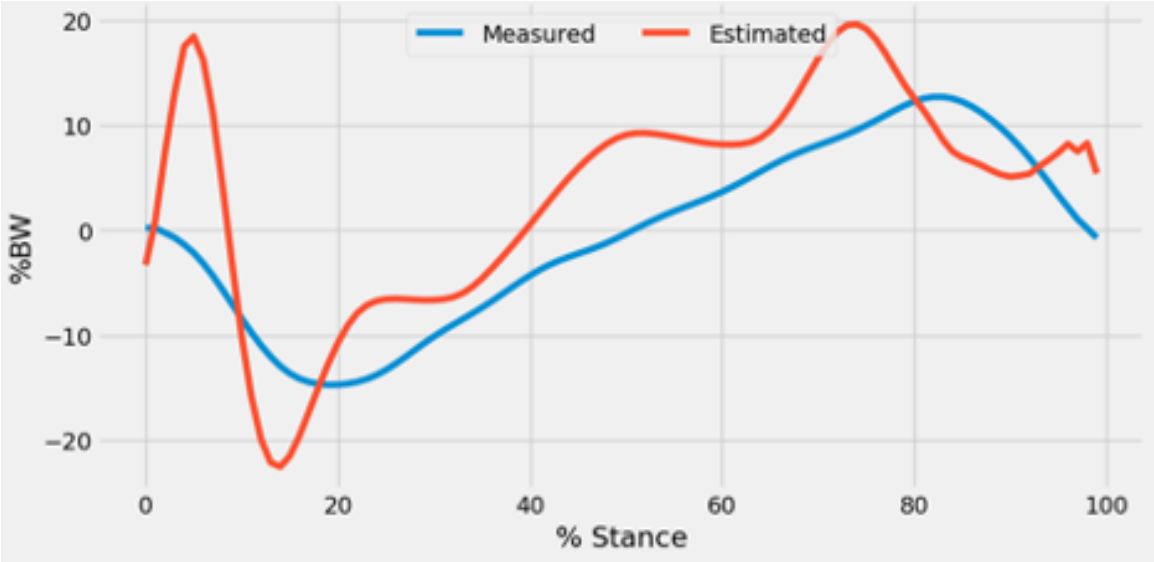


Figure 3·10: Results of APGRF model for Participant 3.

When including all features in the estimation of the APGRF for Participant 3, the

resulting MAPE was 107%, the RMSE was 2.44 and the amount of variance explained was 92%.

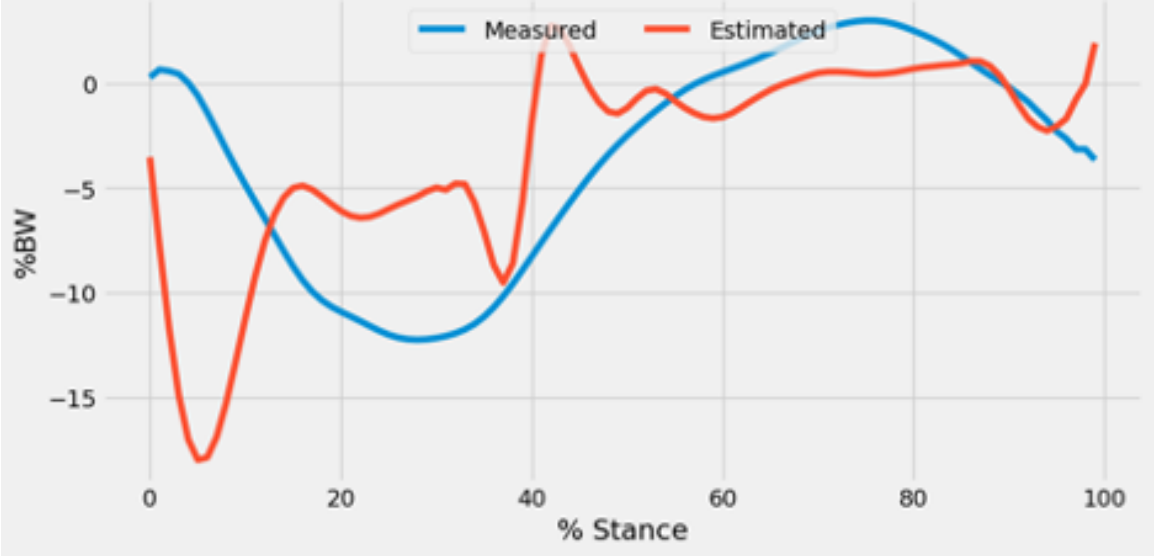


Figure 3·11: Results of APGRF model for Participant 4.

When including all features in the estimation of the APGRF for Participant 4, the resulting MAPE was 269%, the RMSE was 2.81, and the amount of variance explained was 71%.

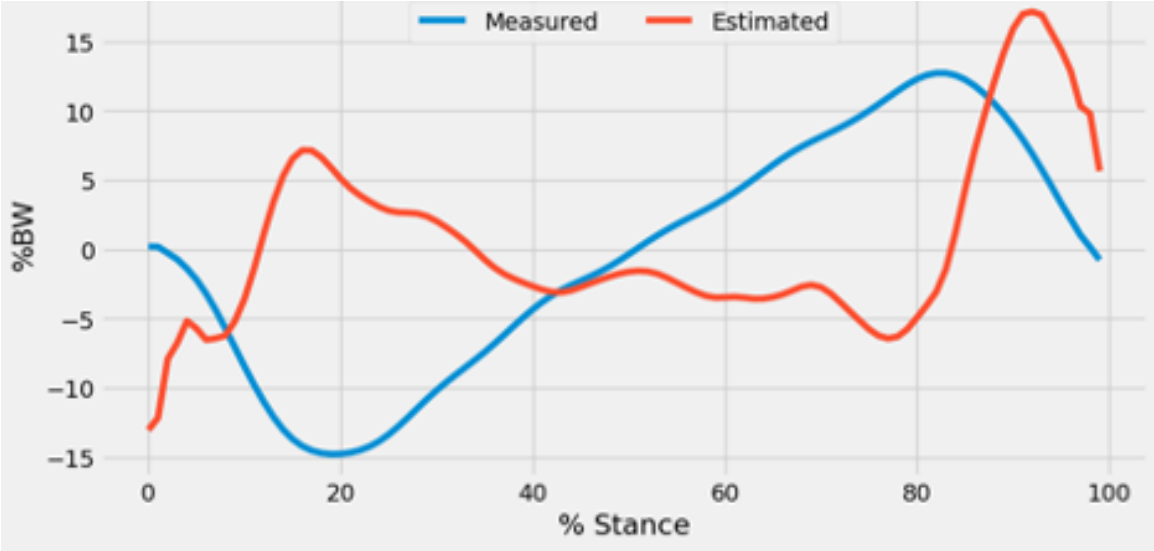


Figure 3·12: Results of APGRF model for all participants.

When including all features in the estimation of the APGRF for the combined set of all participants, the resulting MAPE was 84%, the RMSE was 2.94 and the amount of variance explained was 88%.

Participant	**MAPE**	**RMSE**	R^2
1	1.30	2.54	0.50
2	1.21	1.87	0.85
3	1.07	2.44	0.92
4	2.69	2.81	0.71
All	0.84	2.94	0.88

Table 3.8: Evaluation metrics for results of APGRF models.

3.4 Discussion

3.4.1 Propulsion Metrics Model

The goal of this study was to allow statistical methods, particularly LOO feature selection and linear regression, to select the most significant features from wearable sensors that were most significant in estimating APGRF curve metrics. This would allow human bias to be removed as much as possible, while allowing comparison to the current literature as to the best features produced by wearable sensors to estimate APGRF metrics.

The results of this study show that the significant features from wearables sensors differ at both the metric and participant level; there were few, if any, consistent selections to estimate a particular metric across all participants or to explain multiple metrics within the same participant. In addition, the MAPE and RMSE for each model fell outside the stated goal of <5%.

Interestingly, when examining the model trained on all participants, thigh acceleration in the Y (anterior/posterior) direction and foot gyroscopic force in the X (medial/lateral) direction were both selected to estimate peak propulsion magnitude and propulsion impulse; this follows previous literature identifying limb acceleration and ankle moment as key contributors to walking biomechanics (Zelik and Adamczyk,

2016; Hsiao et al., 2016a). However, these produced the largest errors when compared to the measured values.

While this study had negative results, there remain potential avenues for further exploration. First, a larger dataset would likely be beneficial as statistical and machine learning methods are better suited for larger amounts of data. Second, it is possible that using a LOO method for both the feature selection and the model training resulted in a model that was overfitting, leaving it susceptible to outliers; as poststroke walking often leads to more variance in APGRFs, this also would affect the model's ability to perform. The variability of poststroke walking compensatory strategies also introduces complexities in training statistical and machine learning models; perhaps classifying participants into groups based on these strategies or other similarities may lead to more accurate models. Finally, while allowing for the most prominent features to be selected via statistical and machine learning methods can reduce human bias, the features were limited by the wearable sensors technology; this includes drift in the component signals and the need for them to be securely mounted to the body segments to minimize soft tissue artifacts and avoid sensors moving relative to the body segment.

3.4.2 APGRF Models

The goal of this study was to allow statistical and machine learning methods to select the most significant features from wearable sensors that most accurately estimate APGRF curves. This would allow human bias to be removed as much as possible, while allowing comparison to the current literature as to the optimal wearable sensor features to estimate APGRFs.

The results of this study show that this approach is entirely insufficient. Even when including all available features, the APGRF estimations were not serviceable. This may be due in part to APGRFs being time-series data, i.e. each point follows the

previous in time and that the order of the data is vital. A large majority of statistical and machine learning approaches are not suitable for time-series data; those that are often employ auto-regression and moving averages to help estimate future data. These methods were not used in this study as the stated goal was to estimate APGRF curves using only data from wearable sensors. This was done in order to identify the most relevant data from wearable sensors in estimating poststroke walking biomechanics.

While this study had negative results, there remain potential avenues for further exploration. First, a larger dataset would likely be beneficial as statistical and machine learning methods are better suited for larger amounts of data. Second, more work can be done to explore potential solutions to estimating time-series data without the use of past data; more advanced techniques such as NNs or large, pretrained models may be solutions.

Chapter 4

Conclusions

The combined goals of these studies were to help validate the existing literature as to the most important factors in long distance walking function, as well as validate technologies to help clinicians measure these factors in the clinic. Towards the first goal, the first study was informative: while finding that propulsion metrics in the paretic limb do contribute, braking metrics in both the paretic and nonparetic limbs may be more important factors. While it remains important to address propulsion deficits, this study illustrates that improving braking function must also be considered when determining long distance walking function poststroke.

With regards to the second goal, the second study provided no positive results. The first aim of estimating propulsion metrics using IMU data was accurate, but did not achieve the stated goal of being below a 5% error rate; while a slightly higher error rate, such as those of some participant specific metrics, may be useful in the clinic, most were much higher. The second aim of estimating APGRF time series curves using IMU data was entirely inaccurate; attempting to reduce human bias proved to not be nearly as useful as having domain specific knowledge to craft a biomechanically relevant approach (Revi et al., 2020)

These studies show that, much like negative results, human bias has its use in science.

Appendix A

Code for Chapter 2

This appendix contains the programming code for Chapter 2: Evaluation of propulsion metrics on long distance walking function. All code was written in MATLAB 2019a (Mathworks).

A.1 T1_ExtractData.m

Code to extract motion capture data:

```
clear
close all
clc

%%

cd 'filepath'

% Get ID and paretic side information
%%% this was done manually and put in a .csv file; just reading
    that
df = table2struct(readtable('Subject_Info.csv'));

All.ID(:,1) = string({df.ID});
All.Paretic(:,1) = string({df.PareticSide});
All.Mass(:,1) = [df.Weight];

Subject = All.ID;
Side = ["Paretic", "Nonparetic"];
Day = ["NS", "S"];
Condition = ["Pre", "Post"];

```

```
22 %% Read in Ascii files
23
24 %%% reads in each file from ID list
25
26 for i = 1:length(Subject)
27
28     sub = Subject(i);
29     folder.Data = 'filepath';
30
31     % Create file names
32     file1 = strcat(sub, '_Left.txt');
33     file2 = strcat(sub, '_Right.txt');
34
35     % Set up file paths
36     folder.NS.Pre = strcat(folder.Data, sub, '\No Suit\4 Exported
       Data\Pre');
37     folder.NS.Post = strcat(folder.Data, sub, '\No Suit\4
      Exported Data\Post');
38
39     folder.S.Pre = strcat(folder.Data, sub, '\Suit\4 Exported
      Data\Pre');
40     folder.S.Post = strcat(folder.Data, sub, '\Suit\4 Exported
      Data\Post');
41
42     % Check which side is paretic
43     paretic = All.Paretic(i);
44
45     for j = 1:length(Day)
46
47         for k = 1:length(Condition)
48
49             day = Day(j);
50             cond = Condition(k);
51
52             cd(folder.(day).(cond));
53
54             % Import files
55             left = importdata(file1);
56             right = importdata(file2);
57
58             if paretic == 'Right'
59
```

```
                All.(sub).(day).(cond).Paretic = right;
                All.(sub).(day).(cond).Nonparetic = left;

            else

                All.(sub).(day).(cond).Paretic = left;
                All.(sub).(day).(cond).Nonparetic = right;

            end


        end

    end

    clear file1 file2 left right sub sns prpo folder paretic

end


cd 'filepath'


%% Extract and Normalize GRFs

% g = 9.8;

for i = 1:length(Subject) % subject

    sub = Subject(i);

    for j = 1:length(Day) % day

        day = Day(j);

        for k = 1:length(Condition) % condition

            con = Condition(k);

            for l = 1:length(Side) % side

                side = Side(l);
```

```
102
103                 % Get data and replace NaNs w/ zeros
104                 blah = All.(sub).(day).(con).(side).data(:,2:end)
    ;
105                 blah(isnan(blah)) = 0;
106
107                 % Initialize counter variable
108                 m = 1;
109
110                 % For each stride (set of x, y and z)
111                 for n = 1:3:size(blah,2)
112
113                     % Get single stride
114                     stride = blah(:,n:(n+2));
115
116                     % Remove zeros from arrays
117                     x = nonzeros(stride(:,1));
118                     y = nonzeros(stride(:,2));
119                     z = nonzeros(stride(:,3));
120
121                     % Resample to 100 points
122                     X = resample(x,100,length(x));
123                     Y = resample(y,100,length(y));
124                     Z = resample(z,100,length(z));
125
126                     % Move to structure
127                     All.(sub).(day).(con).(side).GRF.X(:,m) = X
    *100;
128                     All.(sub).(day).(con).(side).GRF.Y(:,m)  = Y
    *100;
129                     All.(sub).(day).(con).(side).GRF.Z(:,m)  = Z
    *100;
130
131                     % Add to counter
132                     m = m+1;
133
134                 end
135
136                 clear stride m x y z X Y Z
137
138             end
139
```

```
            clear side blah

        end

        clear con

    end

    clear day

end

clear sub g


%%
save('All.mat', 'All')
```

A.2 T2_CalculateMetrics.m

Code to calculate metrics of APGRF curve:

```
clear
close all
clc

% Load file w/ data
load('All.mat')

% Get number of subjects
Subject = All.ID;
Side = ["Paretic", "Nonparetic"];
Day = ["NS", "S"];
Condition = ["Pre", "Post"];


%% Calculate metrics - % stance time

% For each subject
for i = 1:length(Subject)

    sub = Subject(i);
```

```
21
22      % For each day
23      for j = 1:length(Day)
24
25          day = Day(j);
26
27          % For each condition
28          for k = 1:length(Condition)
29
30              con = Condition(k);
31
32              % For each side
33              for l = 1:length(Side)
34
35                  side = Side(l);
36
37                  % Get APGRF
38                  obj = All.(sub).(day).(con).(side).GRF.Y;
39
40                  % For each stride
41                  for n = 1:size(obj,2)
42
43                      % Get stride
44                      stride = obj(:,n);
45
46                      %% Find when max braking and propulsion
    magnitude occurs for each stride
47                      prop.peak.mag(n,:) = max(stride(40:end));  %
    assume prop happens after 40% stance
48                      brake.peak.mag(n,:) = min(stride(10:50)); %
    assume braking happens 10-50% stance
49
50                      brake.peak.time(n,:) = find(stride == brake.
    peak.mag(n)); % location of peak brake per stride
51                      prop.peak.time(n,:) = find(stride == prop.
    peak.mag(n)); % location of peak prop per stride
52
53                      %% Find propulsion onset (zero crossing)
54                      section = stride(brake.peak.time(n):prop.peak
    .time(n)); % get array from braking to prop maxes
55                      pos = find(section >= 0, 1); % find first
    propulsion onset value
```

```
56
57                  % If there's no braking phase
58                  if brake.peak.mag(n) >= 0
59
60                      prop.onset(n,:) = brake.peak.time(n);
61
62                  % If there's a point at y=0 (highly unlikely)
63                  elseif section(pos) == 0
64
65                      prop.onset(n,:) = brake.peak.time(n) +
   pos - 1;
66
67                  % If there's no propulsion
68                  elseif isempty(pos)
69
70                      prop.onset(n,:) = 0;
71
72                  % If pos ~= 0 (very likely)
73                  else
74
75                      neg = pos - 1; % get last negative value
76                      rise = section(pos) + (-section(neg)); %
   calculate change in y
77                      run = pos - neg; % calculate change in x
   (probably 1)
78                      slope = rise / run; % find slope b/w last
    negative and first positive points
79                      chy = abs(section(neg)); % change in y
   from last negative to y=0
80                      prchy = chy / slope; % percent change in
   y compared to slope
81                      chx = run * prchy; % percent change in x
   from last negative to y=0
82                      cross = neg + chx; % change in x from
   last negative to y=0
83                      prop.onset(n,:) = brake.peak.time(n) +
   cross - 1; % calculate x when y=0
84
85                  end
86
87                  % Clear variables for future use
```

```
88                  clear section pos neg rise run slope chy
    prchy chx cross
89
90                  %% Find end of propulsion
91                  section = stride(prop.peak.time(n):end); %
    from peak propulsion to end of stride
92                  neg = find(section <=0,1); % find that dip
    below zero at the end of stance
93
94                  % If there's no dip below zero at the end
95                  if isempty(neg)
96
97                      prop.end(n,:) = length(stride);
98
99                  % If there's a point at y=0 (highly unlikely)
100                 elseif section(neg) == 0
101
102                     prop.end(n,:) = prop.peak.time(n) + neg -
     1;
103
104                 % If there's no propulsion
105                 elseif prop.onset(n,:) == 0
106
107                     prop.end(n,:) = 0;
108
109                 % If neg ~= 0 (very likely)
110                 else
111
112                     pos = neg - 1; % get last positive value
113                     rise = section(pos) + (-section(neg)); %
    calculate change in y
114                     run = neg - pos; % calculate change in x
    (probably 1)
115                     slope = rise / run; % find neg b/w last
    negative and first positive points
116                     chy = rise - (-section(neg)); % change in
     y from last negative to y=0
117                     prchy = chy / slope; % percent change in
    y compared to slope
118                     chx = run * prchy; % percent change in x
    from last negative to y=0
```

```
119                         cross = pos + chx; % change in x from
    last negative to y=0
120                         prop.end(n,:) = prop.peak.time(n) + cross
     - 1; % calculate x when y=0
121 
122                     end
123 
124                     % Clear variables for future use
125                     clear section pos neg rise run slope chy
    prchy chx cross
126 
127                     %% Calculate total propulsion time
128                     prop.time(n,:) = prop.end(n) - prop.onset(n);
129 
130                     %% Calculate propulsion impulse
131 
132                     % If there's no proulsion
133                     if prop.onset(n,:) == 0
134 
135                         prop.impulse(n,:) = 0;
136 
137                     % If there is propulsion
138                     else
139 
140                         section = stride(ceil(prop.onset(n)):ceil
    (prop.end(n))); % get propulsion part
141                         prop.impulse(n,:) = trapz(section) /
    length(stride); % take integral, divide by total number of
    points
142 
143                     end
144 
145                     % Clear variables for future use
146                     clear section
147 
148                     %% Find braking onset
149                     section = stride(1:brake.peak.time(n)); % get
     array from beginning to peak braking
150                     pos = find(section >=0, 1, 'last'); % find
    all positive values - last one will be right before braking
151 
152                     % If there's no braking phase
```

```
153                     if brake.peak.mag(n) >= 0
154
155                         brake.onset(n,:) = NaN;
156
157                      % If there's no propulsion before peak
    braking
158                     elseif isempty(pos)
159
160                         brake.onset(n,:) = 1;  % set braking
    onset at beginning
161
162                     % If there's some propulsion before peak
    braking
163                     else
164
165                         % If there's a point at y=0 (highly
    unlikely)
166                         if pos == 0
167
168                             brake.onset(n,:) = pos - 1;
169
170                         % If pos ~= 0 (very likely)
171                         else
172
173                             neg = pos + 1; % get first negative
    value
174                             rise = section(pos) + (-section(neg))
    ; % calculate change in y
175                             run = pos - neg; % calculate change
    in x (probably -1)
176                             slope = rise / run; % find slope b/w
    last positive and first negative points
177                             chy = section(pos); % change in y
    from last positive to y=0
178                             prchy = chy / slope; % percent change
     in y compared to slope
179                             chx = run * prchy; % percent change
    in x from last positive to y=0
180                             cross = pos + chx; % change in x from
     last negative to y=0
181                             brake.onset(n,:) = cross; % calculate
     x when y=0
```

```
182
183                         end
184
185                     end
186
187                     % Clear variables for future use
188                     clear section pos neg rise run slope chy
    prchy chx cross
189
190                     %% Calculate total braking time
191
192                     % If there's no braking phase
193                     if brake.peak.mag(n) >= 0
194
195                         brake.time(n,:) = 0;
196
197                     % If there IS a braking phase
198                     else
199
200                         brake.time(n,:) = prop.onset(n) - brake.
    onset(n);
201
202                     end
203
204                     %% Calculate braking impulse
205
206                     % If there's no braking phase
207                     if brake.peak.mag(n) >= 0
208
209                         brake.impulse(n,:) = 0;
210
211                     % If there IS a braking phase
212                     else
213
214                         section = stride(ceil(brake.onset(n)):
    floor(prop.onset(n))); % get braking part
215                         brake.impulse(n,:) = trapz(section) /
    length(stride); % take integral, divide by total number of
    points
216
217                     end
218
```

```
219                     % Clear variables for future use
220                     clear section
221 
222                     %% Calculate time b/w peak braking and peak
    propulsion
223                     time_bw(n,:) = prop.peak.time(n) - brake.peak
    .time(n);
224 
225                 end
226 
227                 %% Transfer things to the structure
228 
229                 % Braking onset time
230                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    onset.raw...
231                     = brake.onset; % raw data
232                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    onset.mean...
233                     = mean(brake.onset); % mean
234                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    onset.median...
235                     = median(brake.onset); % median
236                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    onset.std...
237                     = std(brake.onset); % standard deviation
238 
239                 % Braking peak time
240                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    peak_time.raw...
241                     = brake.peak.time; % raw data
242                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    peak_time.mean...
243                     = mean(brake.peak.time); % mean
244                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    peak_time.median...
245                     = median(brake.peak.time); % median
246                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    peak_time.std...
247                     = std(brake.peak.time); % standard deviation
248 
249                 % Braking peak magnitude
```

```
250                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    peak_mag.raw...
251                     = brake.peak.mag; % raw data
252                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    peak_mag.mean...
253                     = mean(brake.peak.mag); % mean
254                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    peak_mag.median...
255                     = median(brake.peak.mag); % median
256                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    peak_mag.std...
257                     = std(brake.peak.mag); % standard deviation
258
259                 % Braking impulse
260                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    impulse.raw...
261                     = brake.impulse; % raw data
262                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    impulse.mean...
263                     = mean(brake.impulse); % mean
264                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    impulse.median...
265                     = median(brake.impulse); % median
266                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    impulse.std...
267                     = std(brake.impulse); % standard deviation
268
269                 % Braking total time
270                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    total_time.raw...
271                     = brake.time; % raw data
272                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    total_time.mean...
273                     = mean(brake.time); % mean
274                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    total_time.median...
275                     = median(brake.time); % median
276                 All.(sub).(day).(con).(side).GRF.metrics.brake.
    total_time.std...
277                     = std(brake.time); % standard deviation
278
279                 % Propulsion onset time
```

```
280                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    onset.raw...
281                     = prop.onset; % raw data
282                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    onset.mean...
283                     = mean(prop.onset); % mean
284                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    onset.median...
285                     = median(prop.onset); % median
286                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    onset.std...
287                     = std(prop.onset); % standard deviation
288
289                 % Propulsion peak time
290                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    peak_time.raw...
291                     = prop.peak.time; % raw data
292                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    peak_time.mean...
293                     = mean(prop.peak.time); % mean
294                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    peak_time.median...
295                     = median(prop.peak.time); % median
296                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    peak_time.std...
297                     = std(prop.peak.time); % standard deviation
298
299                 % Propulsion peak magnitude
300                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    peak_mag.raw...
301                     = prop.peak.mag; % raw data
302                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    peak_mag.mean...
303                     = mean(prop.peak.mag); % mean
304                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    peak_mag.median...
305                     = median(prop.peak.mag); % median
306                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    peak_mag.std...
307                     = std(prop.peak.mag); % standard deviation
308
309                 % Propulsion impulse
```

```
310                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    impulse.raw...
311                     = prop.impulse; % raw data
312                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    impulse.mean...
313                     = mean(prop.impulse); % mean
314                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    impulse.median...
315                     = median(prop.impulse); % median
316                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    impulse.std...
317                     = std(prop.impulse); % standard deviation
318
319                 % Propulsion total time
320                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    total_time.raw...
321                     = prop.time; % raw data
322                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    total_time.mean...
323                     = mean(prop.time); % mean
324                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    total_time.median...
325                     = median(prop.time); % median
326                 All.(sub).(day).(con).(side).GRF.metrics.prop.
    total_time.std...
327                     = std(prop.time); % standard deviation
328
329                 % Time b/w peaks
330                 All.(sub).(day).(con).(side).GRF.metrics.bw.time.
    raw...
331                     = time_bw; % raw data
332                 All.(sub).(day).(con).(side).GRF.metrics.bw.time.
    mean...
333                     = mean(time_bw); % mean
334                 All.(sub).(day).(con).(side).GRF.metrics.bw.time.
    median...
335                     = median(time_bw); % median
336                 All.(sub).(day).(con).(side).GRF.metrics.bw.time.
    std...
337                     = std(time_bw); % standard deviation
338
339
```

```
340                 %% Clear all variables
341
342                 clear obj prop brake time_bw section pos neg rise
    run slope chy...
343                     prchy chx cross
344
345             end
346
347         end
348
349     end
350
351 end
352
353
354
355
356 %% Save
357 save('All.mat', 'All')
```

Appendix B

Code for Chapter 3

This appendix contains the programming code for Chapter 3: Validation of wearable sensors for use in estimating poststroke propulsion in the clinic. Code for the importing and cleaning of motion capture and IMU data was written in MATLAB 2019a (Mathworks). Code for processing, testing and training models was written in Python 3.12.2 (Python Software Foundation).

B.1 D1_ExtractData.m

Code to extract IMU data from .txt files, written in MATLAB:

```
clear
close all
clc

%% Set up

% Go to main folder
cd 'filepath'

% Get ID and paretic side information
%%% this was done manually and put in a .csv file; just reading
    that
df = table2struct(readtable('Subject_Info.csv'));

IMU.ID(:,1) = string({df.ID});
IMU.Paretic(:,1) = string({df.PareticSide});
IMU.Mass(:,1) = [df.Weight];

% Assign things
```

```
Subject = IMU.ID;
Side = ["Paretic", "Nonparetic"];
Day = ["NS", "S"];
Condition = ["Pre", "Post"];

%% Read in Ascii files

%%% reads in each file from ID list

Lthigh = "00B41CE8.txt";
Lshank = "00B41CE5.txt";
Lfoot = "00B41CE4.txt";
Rthigh = "00B41CB0.txt";
Rshank = "00B41CA6.txt";
Rfoot = "00B41CDA.txt";

for i = 1:length(Subject)

    % Get subject ID
    sub = Subject(i);

    % Assign file path where data is located
    folder.Data = strcat('filepath');

    % Create file names
    left.thigh.pre.walk = strcat(sub, '_preCWS-000_', Lthigh);
    left.shank.pre.walk = strcat(sub, '_preCWS-000_', Lshank);
    left.foot.pre.walk = strcat(sub, '_preCWS-000_', Lfoot);
    right.thigh.pre.walk = strcat(sub, '_preCWS-000_', Rthigh);
    right.shank.pre.walk = strcat(sub, '_preCWS-000_', Rshank);
    right.foot.pre.walk = strcat(sub, '_preCWS-000_', Rfoot);

    left.thigh.post.walk = strcat(sub, '_postCWS-000_', Lthigh);
    left.shank.post.walk = strcat(sub, '_postCWS-000_', Lshank);
    left.foot.post.walk = strcat(sub, '_postCWS-000_', Lfoot);
    right.thigh.post.walk = strcat(sub, '_postCWS-000_', Rthigh);
    right.shank.post.walk = strcat(sub, '_postCWS-000_', Rshank);
    right.foot.post.walk = strcat(sub, '_postCWS-000_', Rfoot);


    %% Set up file paths
```

```
60      folder.NS = strcat(folder.Data, sub, '\No Suit\IMUs\2
    Exported Data');
61
62      folder.S = strcat(folder.Data, sub, '\Suit\IMUs\2 Exported
    Data');
63
64      %% Check which side is paretic
65      paretic = IMU.Paretic(i);
66
67      % For each day (No Suit, Suit)
68      for j = 1:length(Day)
69
70          % For each condition (Pre, Post)
71          for k = 1:length(Condition)
72
73              % Assign day and cond
74              day = Day(j);
75              con = Condition(k);
76
77              % Skip trials that don't have IMU data
78              if sub == "ME03" && day == "NS" ||...
79                      sub == "ME06" && day == "NS"||...
80                      sub == "ME06" && day == "S" && con == "Pre"
    ||...
81                      sub == "ME07" && day == "NS" ||...
82                      sub == "ME10" && day == "NS" && con == "Post"
     ||...
83                      sub == "ME14" && day == "S" && con == "Post"
    ||...
84                      sub == "ME15" && day == "S" && con == "Post"
85
86                  break
87
88              end
89
90              % Go to folder
91              cd(folder.(day));
92
93              %% Import files
94
95              if paretic == 'Right'
96
```

```
                % Paretic Thigh
                IMU.(sub).(day).Pre.Paretic.Thigh.Walk = ...
                    importdata(right.thigh.pre.walk);
                IMU.(sub).(day).Post.Paretic.Thigh.Walk = ...
                    importdata(right.thigh.post.walk);

                % Paretic Shank
                IMU.(sub).(day).Pre.Paretic.Shank.Walk = ...
                    importdata(right.shank.pre.walk);
                IMU.(sub).(day).Post.Paretic.Shank.Walk = ...
                    importdata(right.shank.post.walk);

                % Paretic Foot
                IMU.(sub).(day).Pre.Paretic.Foot.Walk = ...
                    importdata(right.foot.pre.walk);
                IMU.(sub).(day).Post.Paretic.Foot.Walk = ...
                    importdata(right.foot.post.walk);

                % Nonparetic Thigh
                IMU.(sub).(day).Pre.Nonparetic.Thigh.Walk = ...
                    importdata(left.thigh.pre.walk);
                IMU.(sub).(day).Post.Nonparetic.Thigh.Walk = ...
                    importdata(left.thigh.post.walk);

                % Nonparetic Shank
                IMU.(sub).(day).Pre.Nonparetic.Shank.Walk = ...
                    importdata(left.shank.pre.walk);
                IMU.(sub).(day).Post.Nonparetic.Shank.Walk = ...
                    importdata(left.shank.post.walk);

                % Nonparetic Foot
                IMU.(sub).(day).Pre.Nonparetic.Foot.Walk = ...
                    importdata(left.foot.pre.walk);
                IMU.(sub).(day).Post.Nonparetic.Foot.Walk = ...
                    importdata(left.foot.post.walk);


            else

                % Nonparetic Thigh
                IMU.(sub).(day).Pre.Nonparetic.Thigh.Walk = ...
                    importdata(right.thigh.pre.walk);
```

```
                IMU.(sub).(day).Post.Nonparetic.Thigh.Walk = ...
                    importdata(right.thigh.post.walk);

                % Nonparetic Shank
                IMU.(sub).(day).Pre.Nonparetic.Shank.Walk = ...
                    importdata(right.shank.pre.walk);
                IMU.(sub).(day).Post.Nonparetic.Shank.Walk = ...
                    importdata(right.shank.post.walk);

                % Nonparetic Foot
                IMU.(sub).(day).Pre.Nonparetic.Foot.Walk = ...
                    importdata(right.foot.pre.walk);
                IMU.(sub).(day).Post.Nonparetic.Foot.Walk = ...
                    importdata(right.foot.post.walk);


                % Paretic Thigh
                IMU.(sub).(day).Pre.Paretic.Thigh.Walk = ...
                    importdata(left.thigh.pre.walk);
                IMU.(sub).(day).Post.Paretic.Thigh.Walk = ...
                    importdata(left.thigh.post.walk);

                % Paretic Shank
                IMU.(sub).(day).Pre.Paretic.Shank.Walk = ...
                    importdata(left.shank.pre.walk);
                IMU.(sub).(day).Post.Paretic.Shank.Walk = ...
                    importdata(left.shank.post.walk);

                % Paretic Foot
                IMU.(sub).(day).Pre.Paretic.Foot.Walk = ...
                    importdata(left.foot.pre.walk);
                IMU.(sub).(day).Post.Paretic.Foot.Walk = ...
                    importdata(left.foot.post.walk);

            end


        end

    end

    clear left right sub sns prpo folder paretic
```

```
181
182 end
183
184 %% Return to main folder
185
186 cd 'filepath'
187
188
189 %%
190 save('IMU.mat', 'IMU')
```

B.2 D2_CleanandTrim.m

Code that cleans and trims IMU data, written in MATLAB:

```
clear
close all
clc

%% Go to main folder

cd 'filepath'

% Load IMU strut
load('IMU.mat')

% Load 'All' strut so we can save to it
load('All.mat')


%% Filter and cut to stance phase

% Assign things
Subject = IMU.ID;
Side = ["Paretic", "Nonparetic"];
Day = ["NS", "S"];
Condition = ["Pre", "Post"];
Segment = ["Thigh", "Shank", "Foot"];
Trial = ["Static", "Walk"];

% For each subject
for i = 1:length(Subject)

    % For each day (No Suit, Suit)
    for j = 1:length(Day)

        % For each condition (Pre, Post)
        for k = 1:length(Condition)

            % For each side (Paretic, Nonparetic)
            for l = 1:length(Side)


```

```
39                sub = Subject(i);
40                day = Day(j);
41                con = Condition(k);
42                side = Side(l);
43                tri = Trial(2); % only use walking trials
44
45                % Skip trials that don't have IMU data
46                if sub == "ME03" && day == "NS" ||...
47                        sub == "ME04" && day == "S" && con == "
    Post"||...
48                        sub == "ME06" && day == "NS"||...
49                        sub == "ME06" && day == "S" && con == "
    Post" ||...
50                        sub == "ME07" && day == "NS" ||...
51                        sub == "ME10" && day == "NS" && con == "
    Post" ||...
52                        sub == "ME14" && day == "S" && con == "
    Post" ||...
53                        sub == "ME15" && day == "S" && con == "
    Post"
54
55                    break
56
57                end
58
59                % Combine acc and gyro data from all segments
60                thigh = IMU.(sub).(day).(con).(side).Thigh.(tri).
    data(2:end,2:7);
61                shank = IMU.(sub).(day).(con).(side).Shank.(tri).
    data(2:end,2:7);
62                foot = IMU.(sub).(day).(con).(side).Foot.(tri).
    data(2:end,2:7);
63
64                % Cut to final 30 seconds
65                Fs = 100; % collection frequency
66                trim = 30; % number of seconds being trimed
67                last30 = trim*Fs; % number of data points
68
69                thigh = thigh((end-last30):end, :);
70                shank = shank((end-last30):end, :);
71                foot = foot((end-last30):end, :);
72
```

```
73
74              %% Flip necessary data
75              % Right side IMUs will have their Y (Anterior/
    Posterior) and Z
76              % (Medial/Lateral) flipped to match the lab
    coordinate system;
77              % left side IMUs already match the lab
78
79              % Check which side is paretic
80              paretic = All.Paretic(i);
81
82              % If paretic == Right, flip paretic side
83              if paretic == "Right" && side == "Paretic"
84
85                  thigh(:,[2,3,5,6]) = -thigh(:,[2,3,5,6]);
86                  shank(:,[2,3,5,6]) = -shank(:,[2,3,5,6]);
87                  foot(:,[2,3,5,6]) = -foot(:,[2,3,5,6]);
88
89              % If paretic == Left, flip nonparetic side
90              elseif paretic == "Left" && side == "Nonparetic"
91
92                  thigh(:,[2,3,5,6]) = -thigh(:,[2,3,5,6]);
93                  shank(:,[2,3,5,6]) = -shank(:,[2,3,5,6]);
94                  foot(:,[2,3,5,6]) = -foot(:,[2,3,5,6]);
95
96              end
97
98              % Combine data into single matrix
99              dataIn = horzcat(thigh, shank, foot);
100
101             % Filter data using a lowpass, 2nd order
102             % butterworth filter at 10 Hz
103             nth = 2; % order
104             Wn = (1/10); % filter frequency
105             dataMid = IMUFilter(dataIn, nth, Wn); % filtered
    data
106             % above uses custom function 'IMUFilter'
107
108             % Trim data to stance phase
109             dataOut = IMUStance(dataMid, sub, day, con ,side)
    ; % stance phase data
110             % above uses custom function 'IMUStance'
```

```
111
112                 % Parse out accelerometer and gyro data
113                 All.(sub).(day).(con).(side).IMU.Thigh = ...
114                     dataOut.thigh;
115                 All.(sub).(day).(con).(side).IMU.Shank = ...
116                     dataOut.shank;
117                 All.(sub).(day).(con).(side).IMU.Foot = ...
118                     dataOut.foot;
119
120
121                 clear sub day con side tri thigh shank foot
    paretic dataIN...
122                     Fs trim last30 nth Wn dataMid dataOut
123
124
125             end
126
127         end
128
129     end
130
131 end
132
133 %%
134 save('IMU.mat', 'IMU')
135 save('All.mat', 'All')
```

B.3 D3_FeaturesandResponses.m

Code that seperates IMU and motion capture data into features and responses, written in MATLAB:

```
clear all
close all
clc

%%

load('All.mat')

Subject = All.ID;
Side = ["Paretic", "Nonparetic"];
Day = ["NS", "S"];
Condition = ["Pre", "Post"];
Segment = ["Thigh", "Shank", "Foot"];
Trial = ["Walk"];

%% Create subject specific models

% Table headers
tblhead = ["ThighAccX", "ThighAccY", "ThighAccZ", "ThighGyroX", "
    ThighGyroY", "ThighGyroZ",...
     "ShankAccX", "ShankAccY", "ShankAccZ", "ShankGyroX", "
    ShankGyroY", "ShankGyroZ",...
     "FootAccX", "FootAccY", "FootAccZ", "FootGyroX", "FootGyroY",
     "FootGyroZ"];

% Initialize strut variables
for i = 1:length(Subject)

    for j = 1:length(Side)

        for m = 1:length(Segment)

            sub = Subject(i);
            side = Side(j);
            seg = Segment(m);

            All.(sub).Features.(side).(seg).AccX = [];
```

```
            All.(sub).Features.(side).(seg).AccY = [];
            All.(sub).Features.(side).(seg).AccZ = [];
            All.(sub).Features.(side).(seg).GyroX = [];
            All.(sub).Features.(side).(seg).GyroY = [];
            All.(sub).Features.(side).(seg).GyroZ = [];



        end

        All.(sub).Responses.(side).APGRF = [];
        All.(sub).Responses.(side).BrakeMag = [];
        All.(sub).Responses.(side).BrakeImpulse = [];
        All.(sub).Responses.(side).PropMag = [];
        All.(sub).Responses.(side).PropImpulse = [];

    end

end

%%
% Create strut w/ all features
for i = 1:length(Subject)

    for j = 1:length(Day)

        for k = 1:length(Condition)

            % Use try/catch to skip trials that dont' have IMU
   data
            try

                for l = 1:length(Side)

                    for m = 1:length(Segment)

                        sub = Subject(i);
                        day = Day(j);
                        con = Condition(k);
                        side = Side(l);
                        seg = Segment(m);

```

```
76                          % Grab IMU data (features)
77                          AccX = All.(sub).(day).(con).(side).IMU.(
    seg).Acc.X;
78                          AccY = All.(sub).(day).(con).(side).IMU.(
    seg).Acc.Y;
79                          AccZ = All.(sub).(day).(con).(side).IMU.(
    seg).Acc.Z;
80                          GyroX = All.(sub).(day).(con).(side).IMU
    .(seg).Gyro.X;
81                          GyroY = All.(sub).(day).(con).(side).IMU
    .(seg).Gyro.Y;
82                          GyroZ = All.(sub).(day).(con).(side).IMU
    .(seg).Gyro.Z;
83
84                          % Add IMU data to subject specific
    structs
85                          All.(sub).Features.(side).(seg).AccX =...
86                              horzcat(All.(sub).Features.(side).(
    seg).AccX,...
87                              AccX);
88                          All.(sub).Features.(side).(seg).AccY =...
89                              horzcat(All.(sub).Features.(side).(
    seg).AccY,...
90                              AccY);
91                          All.(sub).Features.(side).(seg).AccZ =...
92                              horzcat(All.(sub).Features.(side).(
    seg).AccZ,...
93                              AccZ);
94
95                          All.(sub).Features.(side).(seg).GyroX
    =...
96                              horzcat(All.(sub).Features.(side).(
    seg).GyroX,...
97                              GyroX);
98                          All.(sub).Features.(side).(seg).GyroY
    =...
99                              horzcat(All.(sub).Features.(side).(
    seg).GyroY,...
100                             GyroY);
101                         All.(sub).Features.(side).(seg).GyroZ
    =...
```

```
102                             horzcat(All.(sub).Features.(side).(
    seg).GyroZ,...
103                             GyroZ);
104 
105                         % Clear relevant variables after use
106                         clear AccX AccY AccZ GyroX GyroY GyroZ
    Brake Prop
107 
108                     end
109 
110                     % Grab GRF metrics (responses)
111                     BrakeMag = All.(sub).(day).(con).(side).GRF.
    metrics.brake.peak_mag.raw;
112                     BrakeImp = All.(sub).(day).(con).(side).GRF.
    metrics.brake.impulse.raw;
113                     PropMag = All.(sub).(day).(con).(side).GRF.
    metrics.prop.peak_mag.raw;
114                     PropImp = All.(sub).(day).(con).(side).GRF.
    metrics.prop.impulse.raw;
115 
116                     % Grab APGRF (responses)
117                     APGRF = All.(sub).(day).(con).(side).GRF.Y;
118 
119                     % Add metrics to subject specific struct
120                     All.(sub).Responses.(side).BrakeMag = ...
121                         vertcat(All.(sub).Responses.(side).
    BrakeMag, BrakeMag);
122                     All.(sub).Responses.(side).BrakeImpulse = ...
123                         vertcat(All.(sub).Responses.(side).
    BrakeImpulse, BrakeImp);
124                     All.(sub).Responses.(side).PropMag = ...
125                         vertcat(All.(sub).Responses.(side).
    PropMag, PropMag);
126                     All.(sub).Responses.(side).PropImpulse = ...
127                         vertcat(All.(sub).Responses.(side).
    PropImpulse, PropImp);
128                     All.(sub).Responses.(side).APGRF = ...
129                         horzcat(All.(sub).Responses.(side).APGRF,
     APGRF);
130 
131                     % Clear relevant variables after use
```

```
132                     clear BrakeMag BrakeImp PropMag PropImp
    BrakeTime PropTime
133
134                 end
135
136             catch
137
138                 % If there is an error - most likely to due w/
    not having
139                 % IMU data for this trial - display message
140                 disp(strcat(sub, " does not have IMU data for the
     ",...
141                     day, " ", con, " condition."))
142
143             end % for try/catch
144
145
146
147         end
148
149     end
150
151 end
152
153
154 %% Rearrange data for ease of use
155
156 Subject = All.ID;
157 Side = ["Paretic", "Nonparetic"];
158 Day = ["NS", "S"];
159 Condition = ["Pre", "Post"];
160 Segment = ["Thigh", "Shank", "Foot"];
161
162 % For each subject
163 for i = 1:length(Subject)
164
165     % For each side (Paretic, Nonparetic)
166     for l = 1:length(Side)
167
168         % Assign things
169         sub = Subject(i);
170         side = Side(l);
```

```
171
172         allsgm = []; % Initialize variable
173
174         % For each segment (Thigh, Shank, Foot)
175         for m = 1:length(Segment)
176
177             % Assign things
178             seg = Segment(m);
179
180             % Set up variable names for table columns
181             VarNames = cellstr({strcat(sub, side, seg, "AccX"),
    ...
182                 strcat(sub, side, seg, "AccY"), strcat(sub, side,
     seg, "AccZ"),...
183                 strcat(sub, side, seg, "GyroX"), strcat(sub, side
    , seg, "GyroY"),...
184                 strcat(sub, side, seg, "GyroZ")});
185
186             % Get data for this segment in this direction and
    assign
187             % names
188             sgm = struct2table(All.(sub).Features.(side).(seg));
189             sgm.Properties.VariableNames = VarNames;
190
191             % Combine this w/ other segments
192             allsgm = horzcat(allsgm, sgm);
193
194             % Clear variables after use
195             clear sgm VarNames
196
197         end
198
199         % Assign data from both directions to struct
200         All.(sub).Features.(side).Combined = allsgm;
201
202         % Clear variables after use
203         clear allsgm
204
205     end
206
207
208 end
```

B.4 IMUFilter.m

Code used to filter IMU data, written in MATLAB:

```
function dataOut = IMUFilter(dataIn, n, Wn)

    [b,a] = butter(n, Wn);

    dataOut = filter(b,a,dataIn);


end
```

B.5 IMUStance.m

Code to cut IMU data into seperate stance phases, written in MATLAB:

```
function dataOut = IMUStance(dataIn, sub, day, con, side)

% Parse out input variable
thigh = dataIn(:,1:6);
shank = dataIn(:,7:12);
foot = dataIn(:,13:18);

% Get vertical acc of the foot for stance trimming
footX = foot(:,1);

% Find negative peaks; must be >30% of the max negative peak
[pk_neg, loc_neg] = findpeaks(-footX, 'MinPeakDistance', 10,...
    'MinPeakHeight', max(-footX)*0.3);

% Find positive peaks
[pk_pos, loc_pos] = findpeaks(footX, 'MinPeakDistance', 1,...
    'MinPeakHeight', 1);

%% Exploit the pattern in foot vertical acc data
% to determine initial contact (IC) and toe off (TO)

% Stance phase starts w/ IC, so if there's any TOs before the
   first IC, get
% rid of them

b=1; % Counting variable

% For all the positive peaks
for a = 1:length(loc_pos)

    % If there's a positive peak before the first negative peak
    if loc_pos(a) < loc_neg(1)

        % Adjust counter
        b=b+1;

    end

```

```
38 end
39
40 % Trim vector of positive peaks to start after the first negative
       peak
41 loc_pos = loc_pos(b:end);
42
43 % Stance phase ends w/ TO, so if there are any ICs after the last
       TO, get
44 % rid of them
45
46 b = 0; % Counting variable
47 flip_neg = flip(loc_neg); % Flip the neg peak array so that the
       loop runs through in reverse order
48
49 % For each negative peak (in reverse order)
50 for a = 1:length(flip_neg)
51
52     % If there's a negative peak after the last positive peak
53     if flip_neg(a) >= loc_pos(end)
54
55         % Adjust counter
56         b=b+1;
57
58     end
59
60 end
61
62 % Trim vector of negative peaks to end after the last positive
       peak
63 loc_neg = loc_neg(1:end-b);
64
65 %%
66 b = 1; % Counting variable
67 negdist = 65; % Minimum number of points b/w negative peaks
68 posdist = 50; % Minimum number of points b/w negative and
       positive peaks
69
70 % For each negative peak
71 for a = 2:length(loc_neg)
72
73
```

```
74      % If more than a certain number of points pass b/w this
    negative peak
75      % and the next one, this peak is the IC
76      if loc_neg(a) >= (loc_neg(a-1) + negdist)
77
78          cut(b,1) = loc_neg(a-1); % IC
79
80          % Find the first positive peak after IC, at least 25
    points away
81          here = find(loc_pos >= loc_neg(a-1) + posdist, 1);
82
83          % Mark that positive peak as TO
84          cut(b,2) = loc_pos(here);
85
86
87
88          % Some checks based on visual inspection
89          if a > 2 && b > 2
90
91              % If the last TO is the same as this TO, skip
92              if cut(b,2) == cut(b-1,2)
93
94                  cut(b,:) = [];
95
96              end
97
98              % If this IC comes before the last TO, skip last IC
    and TO
99              if cut(b,1) <= cut(b-1,2)
100
101                 cut(b-1,:) = [];
102
103             end
104
105         end
106
107         b=b+1; % Adjust counting variable
108
109     end
110
111
112
```

```
113 end
114
115 % Because 'cut' ends up w/ zeros sometimes and I can't figure out
        why
116 cut = [nonzeros(cut(:,1)) nonzeros(cut(:,2))];
117
118 %% Graph check
119
120 fh = figure('visible', 'off');
121
122 plot(footX)
123 hold on
124 title(strcat(sub, day, con, side, 'Foot Vert Acc'))
125 for a = 1:length(cut)
126
127     plot(cut(a,1), footX(cut(a,1)), 'rx', 'markersize', 10) % IC
128     plot(cut(a,2), footX(cut(a,2)), 'gx', 'markersize', 10) % TO
129
130 end
131
132 print('-bestfit', strcat(sub, day, con, side), '-dpdf')
133 close gcf
134
135
136 %% Trim to stance phase
137 for a = 1:size(cut,1)
138
139     % Get IC and TO for this stride
140     IC = cut(a,1);
141     TO = cut(a,2);
142
143     % Cut segment data to this stride
144     Fcut = foot(IC:TO, :);
145     Scut = shank(IC:TO, :);
146     Tcut = thigh(IC:TO, :);
147
148     % Resample to 100 points and assign foot data to strut
149     df.foot.Acc.X(:,a) = resample(Fcut(:,1), 100, length(Fcut));
150     df.foot.Acc.Y(:,a) = resample(Fcut(:,2), 100, length(Fcut));
151     df.foot.Acc.Z(:,a) = resample(Fcut(:,3), 100, length(Fcut));
152
153     df.foot.Gyro.X(:,a) = resample(Fcut(:,4), 100, length(Fcut));
```

```
154     df.foot.Gyro.Y(:,a) = resample(Fcut(:,5), 100, length(Fcut));
155     df.foot.Gyro.Z(:,a) = resample(Fcut(:,6), 100, length(Fcut));
156 
157     % Resample to 100 points and assign shank data to strut
158     df.shank.Acc.X(:,a) = resample(Scut(:,1), 100, length(Scut));
159     df.shank.Acc.Y(:,a) = resample(Scut(:,2), 100, length(Scut));
160     df.shank.Acc.Z(:,a) = resample(Scut(:,3), 100, length(Scut));
161 
162     df.shank.Gyro.X(:,a) = resample(Scut(:,4), 100, length(Scut))
    ;
163     df.shank.Gyro.Y(:,a) = resample(Scut(:,5), 100, length(Scut))
    ;
164     df.shank.Gyro.Z(:,a) = resample(Scut(:,6), 100, length(Scut))
    ;
165 
166     % Resample to 100 points and assign thigh data to strut
167     df.thigh.Acc.X(:,a) = resample(Tcut(:,1), 100, length(Tcut));
168     df.thigh.Acc.Y(:,a) = resample(Tcut(:,2), 100, length(Tcut));
169     df.thigh.Acc.Z(:,a) = resample(Tcut(:,3), 100, length(Tcut));
170 
171     df.thigh.Gyro.X(:,a) = resample(Tcut(:,4), 100, length(Tcut))
    ;
172     df.thigh.Gyro.Y(:,a) = resample(Tcut(:,5), 100, length(Tcut))
    ;
173     df.thigh.Gyro.Z(:,a) = resample(Tcut(:,6), 100, length(Tcut))
    ;
174 
175 
176 
177 end
178 
179 
180 %% Output data
181 dataOut = df;
182 
183 
184 end
```

B.6 Main.py

Main script to run functions that train and test models, written in Python:

```
1
2  import dismod
3
4  plt.style.use("seaborn-v0_8-colorblind")
5  color_pal = sns.color_palette()
6
7  # %%
8  ME03 = mod.get_data("ME03")
9  ME04 = mod.get_data("ME04")
10 ME06 = mod.get_data("ME06")
11 ME07 = mod.get_data("ME07")
12 ME10 = mod.get_data("ME10")
13 ME14 = mod.get_data("ME14")
14 ME15 = mod.get_data("ME15")
15 ME_All = mod.get_data("ME_All")
16
17 full_part = [ME04, ME07, ME14, ME15]
18 all_part = [ME03, ME04, ME06, ME07, ME10, ME14, ME15]
19
20 # %%
21
22 for participant in full_part:
23
24     mod.plot_metrics(participant)
25
26 mod.plot_metrics(ME_All)
27 # %%
28
29 for participant in full_part:
30
31     mod.run_apgrf(participant, 18)
32
33 mod.run_apgrf(ME_All, 18)
```

B.7 dismod.py

Module containing functions and dataclasses, written in Python:

```
1
2  import pickle
3  import warnings
4  from dataclasses import asdict, dataclass
5
6  import matplotlib.pyplot as plt
7  import numpy as np
8  import pandas as pd
9  import seaborn as sns
10 from lofo import Dataset, LOFOImportance, plot_importance
11 from sklearn import linear_model
12 from sklearn.feature_selection import f_regression
13 from sklearn.metrics import (mean_absolute_percentage_error,
       r2_score,
14                               root_mean_squared_error)
15 from sklearn.model_selection import (LeaveOneOut,
       cross_val_predict,
16                                       cross_validate,
       train_test_split)
17
18
19 plt.style.use("fivethirtyeight")
20 color_pal = sns.color_palette()
21
22
23 @dataclass(frozen=True)
24 class Participant:
25     def __or__(self, other):
26         return self.__class__(**asdict(self) | asdict(other))
27
28     name: str
29     apgrf: float
30
31     brake_mag: float
32     brake_imp: float
33
34     brake_thigh_accX: float
35     brake_thigh_accY: float
```

```
    brake_thigh_accZ: float
    brake_thigh_gyroX: float
    brake_thigh_gyroY: float
    brake_thigh_gyroZ: float

    brake_shank_accX: float
    brake_shank_accY: float
    brake_shank_accZ: float
    brake_shank_gyroX: float
    brake_shank_gyroY: float
    brake_shank_gyroZ: float

    brake_foot_accX: float
    brake_foot_accY: float
    brake_foot_accZ: float
    brake_foot_gyroX: float
    brake_foot_gyroY: float
    brake_foot_gyroZ: float

    prop_mag: float
    prop_imp: float

    prop_thigh_accX: float
    prop_thigh_accY: float
    prop_thigh_accZ: float
    prop_thigh_gyroX: float
    prop_thigh_gyroY: float
    prop_thigh_gyroZ: float

    prop_shank_accX: float
    prop_shank_accY: float
    prop_shank_accZ: float
    prop_shank_gyroX: float
    prop_shank_gyroY: float
    prop_shank_gyroZ: float

    prop_foot_accX: float
    prop_foot_accY: float
    prop_foot_accZ: float
    prop_foot_gyroX: float
    prop_foot_gyroY: float
    prop_foot_gyroZ: float
```

```
78
79      thigh_accX: float
80      thigh_accY: float
81      thigh_accZ: float
82      thigh_gyroX: float
83      thigh_gyroY: float
84      thigh_gyroZ: float
85
86      shank_accX: float
87      shank_accY: float
88      shank_accZ: float
89      shank_gyroX: float
90      shank_gyroY: float
91      shank_gyroZ: float
92
93      foot_accX: float
94      foot_accY: float
95      foot_accZ: float
96      foot_gyroX: float
97      foot_gyroY: float
98      foot_gyroZ: float
99
100
101 @dataclass(frozen=True)
102 class Results:
103     def __or__(self, other):
104         return self.__class__(**asdict(self) | asdict(other))
105
106     brake_mag: object
107     brake_imp: object
108
109     prop_mag: object
110     prop_imp: object
111
112     metrics_plot: object
113
114     apgrf: object
115
116
117 def get_data(participant_name):
118     file_name = f"data/{participant_name}.xls"
119     df = pd.ExcelFile(file_name)
```

```
120
121     apgrf = pd.read_excel(df, "Sheet1", index_col=None, header=
    None)
122
123     brake = pd.read_excel(df, "Sheet2", index_col=None, header=
    None)
124     prop = pd.read_excel(df, "Sheet3", index_col=None, header=
    None)
125
126     brake_mag = brake.iloc[:, 0]
127     brake_imp = brake.iloc[:, 1]
128     prop_mag = prop.iloc[:, 0]
129     prop_imp = prop.iloc[:, 1]
130
131     brake_thigh_accX = brake.iloc[:, 2]
132     brake_thigh_accY = brake.iloc[:, 3]
133     brake_thigh_accZ = brake.iloc[:, 4]
134     brake_thigh_gyroX = brake.iloc[:, 5]
135     brake_thigh_gyroY = brake.iloc[:, 6]
136     brake_thigh_gyroZ = brake.iloc[:, 7]
137
138     brake_shank_accX = brake.iloc[:, 8]
139     brake_shank_accY = brake.iloc[:, 9]
140     brake_shank_accZ = brake.iloc[:, 10]
141     brake_shank_gyroX = brake.iloc[:, 11]
142     brake_shank_gyroY = brake.iloc[:, 12]
143     brake_shank_gyroZ = brake.iloc[:, 13]
144
145     brake_foot_accX = brake.iloc[:, 14]
146     brake_foot_accY = brake.iloc[:, 15]
147     brake_foot_accZ = brake.iloc[:, 16]
148     brake_foot_gyroX = brake.iloc[:, 17]
149     brake_foot_gyroY = brake.iloc[:, 18]
150     brake_foot_gyroZ = brake.iloc[:, 19]
151
152     prop_thigh_accX = prop.iloc[:, 2]
153     prop_thigh_accY = prop.iloc[:, 3]
154     prop_thigh_accZ = prop.iloc[:, 4]
155     prop_thigh_gyroX = prop.iloc[:, 5]
156     prop_thigh_gyroY = prop.iloc[:, 6]
157     prop_thigh_gyroZ = prop.iloc[:, 7]
158
```

```
159     prop_shank_accX = prop.iloc[:, 8]
160     prop_shank_accY = prop.iloc[:, 9]
161     prop_shank_accZ = prop.iloc[:, 10]
162     prop_shank_gyroX = prop.iloc[:, 11]
163     prop_shank_gyroY = prop.iloc[:, 12]
164     prop_shank_gyroZ = prop.iloc[:, 13]
165 
166     prop_foot_accX = prop.iloc[:, 14]
167     prop_foot_accY = prop.iloc[:, 15]
168     prop_foot_accZ = prop.iloc[:, 16]
169     prop_foot_gyroX = prop.iloc[:, 17]
170     prop_foot_gyroY = prop.iloc[:, 18]
171     prop_foot_gyroZ = prop.iloc[:, 19]
172 
173     thigh_accX = pd.read_excel(df, "Sheet4", index_col=None,
    header=None)
174     thigh_accY = pd.read_excel(df, "Sheet5", index_col=None,
    header=None)
175     thigh_accZ = pd.read_excel(df, "Sheet6", index_col=None,
    header=None)
176     thigh_gryoX = pd.read_excel(df, "Sheet7", index_col=None,
    header=None)
177     thigh_gyroY = pd.read_excel(df, "Sheet8", index_col=None,
    header=None)
178     thigh_gyroZ = pd.read_excel(df, "Sheet9", index_col=None,
    header=None)
179 
180     shank_accX = pd.read_excel(df, "Sheet10", index_col=None,
    header=None)
181     shank_accY = pd.read_excel(df, "Sheet11", index_col=None,
    header=None)
182     shank_accZ = pd.read_excel(df, "Sheet12", index_col=None,
    header=None)
183     shank_gryoX = pd.read_excel(df, "Sheet13", index_col=None,
    header=None)
184     shank_gyroY = pd.read_excel(df, "Sheet14", index_col=None,
    header=None)
185     shank_gyroZ = pd.read_excel(df, "Sheet15", index_col=None,
    header=None)
186 
187     foot_accX = pd.read_excel(df, "Sheet16", index_col=None,
    header=None)
```

```
188     foot_accY = pd.read_excel(df, "Sheet17", index_col=None,
    header=None)
189     foot_accZ = pd.read_excel(df, "Sheet18", index_col=None,
    header=None)
190     foot_gryoX = pd.read_excel(df, "Sheet19", index_col=None,
    header=None)
191     foot_gyroY = pd.read_excel(df, "Sheet20", index_col=None,
    header=None)
192     foot_gyroZ = pd.read_excel(df, "Sheet21", index_col=None,
    header=None)
193 
194     return Participant(
195         participant_name,
196         apgrf,
197         brake_mag,
198         brake_imp,
199         brake_thigh_accX,
200         brake_thigh_accY,
201         brake_thigh_accZ,
202         brake_thigh_gyroX,
203         brake_thigh_gyroY,
204         brake_thigh_gyroZ,
205         brake_shank_accX,
206         brake_shank_accY,
207         brake_shank_accZ,
208         brake_shank_gyroX,
209         brake_shank_gyroY,
210         brake_shank_gyroZ,
211         brake_foot_accX,
212         brake_foot_accY,
213         brake_foot_accZ,
214         brake_foot_gyroX,
215         brake_foot_gyroY,
216         brake_foot_gyroZ,
217         prop_mag,
218         prop_imp,
219         prop_thigh_accX,
220         prop_thigh_accY,
221         prop_thigh_accZ,
222         prop_thigh_gyroX,
223         prop_thigh_gyroY,
224         prop_thigh_gyroZ,
```

```
        prop_shank_accX,
        prop_shank_accY,
        prop_shank_accZ,
        prop_shank_gyroX,
        prop_shank_gyroY,
        prop_shank_gyroZ,
        prop_foot_accX,
        prop_foot_accY,
        prop_foot_accZ,
        prop_foot_gyroX,
        prop_foot_gyroY,
        prop_foot_gyroZ,
        thigh_accX,
        thigh_accY,
        thigh_accZ,
        thigh_gryoX,
        thigh_gyroY,
        thigh_gyroZ,
        shank_accX,
        shank_accY,
        shank_accZ,
        shank_gryoX,
        shank_gyroY,
        shank_gyroZ,
        foot_accX,
        foot_accY,
        foot_accZ,
        foot_gryoX,
        foot_gyroY,
        foot_gyroZ,
    )


def get_brake(participant):

    brake_mag = participant.brake_mag
    brake_imp = participant.brake_imp

    X = pd.DataFrame(
        [
            participant.brake_thigh_accX,
            participant.brake_thigh_accY,
```

```
            participant.brake_thigh_accZ,
            participant.brake_thigh_gyroX,
            participant.brake_thigh_gyroY,
            participant.brake_thigh_gyroZ,
            participant.brake_shank_accX,
            participant.brake_shank_accY,
            participant.brake_shank_accZ,
            participant.brake_shank_gyroX,
            participant.brake_shank_gyroY,
            participant.brake_shank_gyroZ,
            participant.brake_foot_accX,
            participant.brake_foot_accY,
            participant.brake_foot_accZ,
            participant.brake_foot_gyroX,
            participant.brake_foot_gyroY,
            participant.brake_foot_gyroZ,
        ]
    ).T

    return X, brake_mag, brake_imp


def get_prop(participant):

    prop_mag = participant.prop_mag
    prop_imp = participant.prop_imp

    X = pd.DataFrame(
        [
            participant.prop_thigh_accX,
            participant.prop_thigh_accY,
            participant.prop_thigh_accZ,
            participant.prop_thigh_gyroX,
            participant.prop_thigh_gyroY,
            participant.prop_thigh_gyroZ,
            participant.prop_shank_accX,
            participant.prop_shank_accY,
            participant.prop_shank_accZ,
            participant.prop_shank_gyroX,
            participant.prop_shank_gyroY,
            participant.prop_shank_gyroZ,
            participant.prop_foot_accX,
```

```
            participant.prop_foot_accY,
            participant.prop_foot_accZ,
            participant.prop_foot_gyroX,
            participant.prop_foot_gyroY,
            participant.prop_foot_gyroZ,
        ]
    ).T

    return X, prop_mag, prop_imp


def LOFO(X, y):
    labels = [
        "y",
        "Thigh Acc X",
        "Thigh Acc Y",
        "Thigh Acc Z",
        "Thigh Gyro X",
        "Thigh Gyro Y",
        "Thigh Gyro Z",
        "Shank Acc X",
        "Shank Acc Y",
        "Shank Acc Z",
        "Shank Gyro X",
        "Shank Gyro Y",
        "Shank Gyro Z",
        "Foot Acc X",
        "Foot Acc Y",
        "Foot Acc Z",
        "Foot Gyro X",
        "Foot Gyro Y",
        "Foot Gyro Z",
    ]

    df = pd.concat([y, X], axis=1)
    df.columns = labels

    model = linear_model.LinearRegression()
    cv = LeaveOneOut()

    dataset = Dataset(df=df, target="y", features=[col for col in
    df if col != "y"])
```

```
    lofo_imp = LOFOImportance(
        dataset, model=model, cv=cv, scoring="
    neg_root_mean_squared_error"
    )
    importance_df = lofo_imp.get_importance()

    # hold = importance_df[importance_df['importance_mean'] >
    0]['feature']
    hold = importance_df["feature"][0:3]
    selected = [labels.index(hold) if hold in labels else -1 for
    hold in hold]
    print(f"Features selected: {list(hold)}")

    return selected, hold


def LOOF(X, y, data):

    if data == "metrics":

        strides = np.size(y)
        metrics = np.shape(X)[1]
        final_ranks = np.empty(metrics) * 0
        for i in range(metrics):

            X_train = np.delete(X, i, axis=1)
            f_stat, p_values = f_regression(X_train, y)
            order = f_stat.argsort()
            ranks = order.argsort()

            final_ranks = final_ranks + np.insert(ranks, i, 0)

        hold = np.argpartition(final_ranks, 3)

    if data == "apgrf":

        strides = np.shape(y)[1]
        metrics = np.shape(X)[2]
        final_ranks = np.empty(metrics) * 0
        for i in range(metrics):

            X_hold = np.delete(X, i, axis=2)
```

```
389
390             for k in range(strides):
391
392                 f_stat, p_values = f_regression(X_hold[k, :, :],
    y[:, k])
393                 order = f_stat.argsort()
394                 ranks = order.argsort()
395
396                 final_ranks = final_ranks + np.insert(ranks, i,
    0)
397
398         hold = np.argpartition(final_ranks, 3)
399
400     return hold
401
402
403 def train_metrics(X, y):
404
405     model = linear_model.LinearRegression()
406     cv = LeaveOneOut()
407     scoring = [
408         "neg_mean_absolute_percentage_error",
409         "neg_root_mean_squared_error",
410     ]  # , 'r2']
411     scores = cross_validate(
412         model, X, y, scoring=scoring, cv=cv, return_train_score=
    True
413     )
414     y_pred = cross_val_predict(model, X, y, cv=cv)
415
416     y_mean = np.mean(y)
417     y_std = np.std(y)
418     y_pred_mean = np.mean(y_pred)
419     y_pred_std = np.std(y_pred)
420     MAPE_mean = np.mean(np.abs(scores["
    test_neg_mean_absolute_percentage_error"]))
421     MAPE_std = np.std(np.abs(scores["
    test_neg_mean_absolute_percentage_error"]))
422     RMSE_mean = np.mean(scores["test_neg_root_mean_squared_error"
    ])
423     RMSE_std = np.std(scores["test_neg_root_mean_squared_error"])
424
```

```
425     print(f"Y mean: {y_mean:.2f}, StD: {y_std:.2f}")
426     print(f"Y Est mean: {y_pred_mean:.2f}, StD: {y_pred_std:.2f}"
        )
427     print(f"MAPE mean: {MAPE_mean:.2f}, StD: {MAPE_std:.2f}")
428     print(f"RMSE mean: {RMSE_mean:.2f}, StD: {RMSE_std:.2f}")
429 
430     return model, y_pred_mean, y_pred_std, MAPE_mean, MAPE_std,
        RMSE_mean, RMSE_std
431 
432 
433 def run_brake(participant):
434 
435     X, mag, imp = get_brake(participant)
436 
437     print("Metric: Peak Brake Magnitude")
438     print("-----------------")
439     # selected =  LOOF(X, mag, 'metrics')
440     selected, feature_list = LOFO(X, mag)
441     X_selected = X.iloc[:, selected]
442     mag_model, mag_mean, mag_std, mape_mean, mape_std, rmse_mean,
        rmse_std = (
443         train_metrics(X_selected, mag)
444     )
445     save_metrics(
446         "Peak Braking Magnitude",
447         participant.name,
448         feature_list,
449         mag,
450         mag_mean,
451         mag_std,
452         mape_mean,
453         mape_std,
454         rmse_mean,
455         rmse_std,
456     )
457     print("\n")
458 
459     print("Metric: Brake Impulse")
460     print("-----------------")
461     # selected =  LOOF(X, imp, 'metrics')
462     selected, feature_list = LOFO(X, imp)
463     X_selected = X.iloc[:, selected]
```

```
464     imp_model, imp_mean, imp_std, mape_mean, mape_std, rmse_mean,
        rmse_std = (
465         train_metrics(X_selected, imp)
466     )
467     save_metrics(
468         "Braking Impulse",
469         participant.name,
470         feature_list,
471         imp,
472         imp_mean,
473         imp_std,
474         mape_mean,
475         mape_std,
476         rmse_mean,
477         rmse_std,
478     )
479     print("\n")
480 
481     means = [mag_mean, imp_mean]
482     stds = [mag_std, imp_std]
483 
484     mag_fn = f"models/{participant.name}_brakemag"
485     pickle.dump(mag_model, open(mag_fn, "wb"))
486     # participant.model_brakemag = mag_model
487 
488     imp_fn = f"models/{participant.name}_brakeimp"
489     pickle.dump(imp_model, open(imp_fn, "wb"))
490     # participant.model_brakeimp = imp_model
491 
492     return means, stds
493 
494 
495 def run_prop(participant):
496 
497     X, mag, imp = get_prop(participant)
498 
499     print("Metric: Peak Propulsion Magnitude")
500     print("-----------------")
501     # selected =  LOOF(X, mag, 'metrics')
502     selected, feature_list = LOFO(X, mag)
503     X_selected = X.iloc[:, selected]
```

```
504     mag_model, mag_mean, mag_std, mape_mean, mape_std, rmse_mean,
        rmse_std = (
505         train_metrics(X_selected, mag)
506     )
507     save_metrics(
508         "Peak Propulsion Magnitude",
509         participant.name,
510         feature_list,
511         mag,
512         mag_mean,
513         mag_std,
514         mape_mean,
515         mape_std,
516         rmse_mean,
517         rmse_std,
518     )
519     print("\n")
520
521     print("Metric: Propulsion Impulse")
522     print("------------------")
523     # selected =  LOOF(X, imp, 'metrics')
524     selected, feature_list = LOFO(X, imp)
525     X_selected = X.iloc[:, selected]
526     imp_model, imp_mean, imp_std, mape_mean, mape_std, rmse_mean,
        rmse_std = (
527         train_metrics(X_selected, imp)
528     )
529     save_metrics(
530         "Propulsion Impulse",
531         participant.name,
532         feature_list,
533         imp,
534         imp_mean,
535         imp_std,
536         mape_mean,
537         mape_std,
538         rmse_mean,
539         rmse_std,
540     )
541     print("\n")
542
543     means = [mag_mean, imp_mean]
```

```
544     stds = [mag_std, imp_std]
545 
546     mag_fn = f"models/{participant.name}_propmag"
547     pickle.dump(mag_model, open(mag_fn, "wb"))
548     # participant.model_propmag = mag_model
549 
550     imp_fn = f"models/{participant.name}_propimp"
551     pickle.dump(imp_model, open(imp_fn, "wb"))
552     # participant.model_propimp = imp_model
553 
554     return means, stds
555 
556 
557 def save_metrics(
558     metric_name,
559     participant_name,
560     feature_list,
561     y,
562     y_pred_mean,
563     y_pred_std,
564     mape_mean,
565     mape_std,
566     rmse_mean,
567     rmse_std,
568 ):
569     y_mean = np.mean(y)
570     y_std = np.std(y)
571 
572     metrics = pd.DataFrame([y_mean, y_std, y_pred_mean,
    y_pred_std]).T
573     errors = pd.DataFrame([mape_mean, mape_std, rmse_mean,
    rmse_std]).T
574     df = pd.concat([metrics, errors])
575 
576     file_path = f"{participant_name}_metrics.txt"
577     with open(file_path, "a") as f:
578         f.write(f"{metric_name}\n")
579         f.write("--------------------\n")
580         f.write(f"{list(feature_list)}\n")
581         f.write(f"Measured: {y_mean:.2f}, StD: {y_std:.2f}\n")
582         f.write(f"Estimated: {y_pred_mean:.2f}, StD: {y_pred_std
    :.2f}\n")
```

```
583             f.write(f"MAPE mean: {mape_mean:.2f}, StD: {mape_std:.2f
    }\n")
584             f.write(f"RMSE mean: {rmse_mean:.2f}, StD: {rmse_std:.2f
    }\n")
585             f.write("\n")
586
587
588 def plot_metrics(participant):
589     print(f"{participant.name}")
590     brake_mag, brake_imp = get_brake(participant)[1:]
591     prop_mag, prop_imp = get_prop(participant)[1:]
592
593     brake_mag_mean = np.mean(brake_mag)
594     brake_mag_std = np.std(brake_mag)
595     brake_imp_mean = np.mean(brake_imp)
596     brake_imp_std = np.std(brake_imp)
597     prop_mag_mean = np.mean(prop_mag)
598     prop_mag_std = np.std(prop_mag)
599     prop_imp_mean = np.mean(prop_imp)
600     prop_imp_std = np.std(prop_imp)
601     brake_est_means, brake_est_stds = run_brake(participant)
602     prop_est_means, prop_est_stds = run_prop(participant)
603
604     metrics = (
605         "Peak Braking\nMagnitude",
606         "Braking\nImpulse",
607         "Peak Propulsion\nMagnitude",
608         "Propulsion\nImpulse",
609     )
610     heights = {
611         "Measured": np.round(
612             (-brake_mag_mean, -brake_imp_mean, prop_mag_mean,
    prop_imp_mean), 2
613         ),
614         "Estimated": np.round(
615             (
616                 -brake_est_means[0],
617                 -brake_est_means[1],
618                 prop_est_means[0],
619                 prop_est_means[1],
620             ),
621             2,
```

```
        ),
    }

    std_bars = [
        brake_mag_std / 2,
        brake_est_stds[0],
        brake_imp_std / 2,
        brake_est_stds[1],
        prop_mag_std / 2,
        prop_est_stds[0],
        prop_imp_std / 2,
        prop_est_stds[1],
    ]

    x = np.arange(len(metrics))
    width = 0.25
    multiplier = 0
    i = 0

    fig, ax = plt.subplots(layout="constrained")

    for measure, result in heights.items():

        offset = width * multiplier
        rects = ax.bar(
            x + offset,
            result,
            width,
            label=measure,
            yerr=std_bars[i],  # / 2,
            error_kw={"elinewidth": 1, "capsize": 3, "capthick":
1},
        )
        ax.bar_label(rects, padding=3)
        multiplier += 1
        i += 1

    ax.set_ylim(0, np.max(np.abs(brake_mag_mean)) + np.max(
std_bars) + 2)
    ax.set_title(f"{participant.name} Metrics")
    ax.set_xticks(x + (width / 2), metrics)
    ax.legend(loc="upper center", ncols=2)
```

```
662
663     # plt.show()
664
665     filename = f"figures/{participant.name}_metrics.pdf"
666     plt.savefig(filename, bbox_inches="tight")
667
668     return fig
669
670 def get_APGRF(participant):
671
672     y = np.array(participant.apgrf)
673     X = np.array(
674         [
675             participant.thigh_accX,
676             participant.thigh_accY,
677             participant.thigh_accZ,
678             participant.thigh_gyroX,
679             participant.thigh_gyroY,
680             participant.thigh_gyroZ,
681             participant.shank_accX,
682             participant.shank_accY,
683             participant.shank_accZ,
684             participant.shank_gyroX,
685             participant.shank_gyroY,
686             participant.shank_gyroZ,
687             participant.foot_accX,
688             participant.foot_accY,
689             participant.foot_accZ,
690             participant.foot_gyroX,
691             participant.foot_gyroY,
692             participant.foot_gyroZ,
693         ]
694     )
695
696     strides = np.shape(y)[1]
697     metrics = np.shape(X)[0]
698     X_hold = np.empty([strides, 100, metrics]) * 0
699     for i in range(strides):
700
701         X_hold[i, :, :] = X[:, :, i].T
702
703     return X_hold, y
```

```
704
705
706 def features_apgrf(X, y, feature_count):
707     labels = [
708         "Thigh Acc X",
709         "Thigh Acc Y",
710         "Thigh Acc Z",
711         "Thigh Gyro X",
712         "Thigh Gyro Y",
713         "Thigh Gyro Z",
714         "Shank Acc X",
715         "Shank Acc Y",
716         "Shank Acc Z",
717         "Shank Gyro X",
718         "Shank Gyro Y",
719         "Shank Gyro Z",
720         "Foot Acc X",
721         "Foot Acc Y",
722         "Foot Acc Z",
723         "Foot Gyro X",
724         "Foot Gyro Y",
725         "Foot Gyro Z",
726     ]
727
728     rank = LOOF(X, y, data="apgrf")
729     print("-----------")
730     print("Features selected: ")
731
732     for n in range(feature_count):
733         print(labels[rank[n]])
734
735     return rank[0:feature_count]
736
737
738 def split_apgrf(X, y, train_size):
739
740     # train_size = 0.75
741     cut = np.int32(np.round(np.shape(y)[1] * train_size, decimals
    =0))
742     cut / np.shape(y)[1]
743     train = cut / np.shape(y)[1]
744
```

```
    X_hold = np.reshape(X, (-1, X.shape[2]), order="F")
    y_hold = y.flatten(order="F")

    X_train, X_test, y_train, y_test = train_test_split(
        X_hold, y_hold, train_size=train, random_state=305
    )

    X_train = np.reshape(X_train, (-1, 100, 18))
    y_train = np.reshape(y_train, (100, -1))

    return X_train, X_test, y_train, y_test


def train_apgrf(X, y, metrics):

    strides = y.shape[1]

    reg = linear_model.SGDRegressor()

    for i in range(strides - 1):

        X_hold = X[i, :, metrics].T
        y_hold = y[:, i]
        reg.partial_fit(X_hold, y_hold)

    y_pred = reg.predict(X[i + 1, :, metrics].T)
    y_test = y[:, i + 1]
    print(f"MAPE: {mean_absolute_percentage_error(y_test, y_pred)
    :.2f}")
    print(f"RMSE: {root_mean_squared_error(y_test, y_pred):.2f}")
    print(f"R2: {r2_score(y_test, y_pred):.2f}")

    return reg


def plot_apgrf(y_test, y_pred):

    fig, ax = plt.subplots(figsize=(10, 5))
    ax.plot(range(len(y_test)), y_test, color=color_pal[0], label
    ="Measured")
    ax.plot(range(len(y_pred)), y_pred, color=color_pal[1], label
    ="Estimated")
```

```
    ax.legend(loc="upper center", ncols=2)
    ax.set_ylabel("%BW")
    ax.set_xlabel("% Stance")

    return fig, ax


def save_apgrf(participant, y_test, y_pred):
    y_test_hold = np.reshape(y_test, (100, -1))
    y_test_avg = np.mean(y_test_hold, axis=1)
    y_pred_hold = np.reshape(y_pred, (100, -1))
    y_pred_avg = np.mean(y_pred_hold, axis=1)
    df = pd.DataFrame([y_test_avg, y_pred_avg]).T
    df.to_csv(f"{participant.name}.txt", header=None, index=None,
    sep=",", mode="w")


def run_apgrf(participant, features):

    X, y = get_APGRF(participant)
    metrics = features_apgrf(X, y, features)

    X_train, X_test, y_train, y_test = split_apgrf(X, y, 0.7)

    model = train_apgrf(X_train, y_train, metrics)

    y_pred = model.predict(X_test)

    save_apgrf(participant, y_test, y_pred)

    fig, ax = plot_apgrf(y_test, y_pred)
    ax.set_title(f"{participant.name} APGRF")
    filename = f"figures/{participant.name}_apgrf.pdf"
    plt.savefig(filename, bbox_inches="tight")
    plt.close(fig)
```

References

Abaid, N., Cappa, P., Palermo, E., Petrarca, M., and Porfiri, M. (2013). Gait Detection in Children with and without Hemiplegia Using Single-Axis Wearable Gyroscopes. *PLoS ONE*, 8(9).

Agresta, C. and Brown, A. (2015). Gait retraining for injured and healthy runners using augmented feedback: A systematic literature review. *Journal of Orthopaedic & Sports Physical Therapy*, 45(8):576–584.

Alaqtash, M., Sarkodie-Gyan, T., Yu, H., Fuentes, O., Brower, R., and Abdelgawad, A. (2011). Automatic classification of pathological gait patterns using ground reaction forces and machine learning algorithms. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 453–457.

Alvarez, A. M., Collimore, A. N., Aiello, A. J. M., Binder-Macelod, S. A., and Awad, L. N. (2020). Propulsion timing affects the relationship between paretic propulsion and long-distance walking function after stroke [Poster session]. In *American Society of Biomechanics 2020*.

Aminian, K., Najafi, B., Büla, C., Leyvraz, P.-F., and Robert, Ph. (2002). Spatio-temporal parameters of gait measured by an ambulatory system using miniature gyroscopes. *Journal of Biomechanics*, 35(5):689–699.

Ancillao, A., Tedesco, S., Barton, J., and O'flynn, B. (2018). Indirect measurement of ground reaction forces and moments by means of wearable inertial sensors: A systematic review. *Sensors*, 18(8).

Aurbach, M., Wagner, K., Süß, F., and Dendorfer, S. (2017). Implementation and validation of human kinematics measured using IMUs for musculoskeletal simulations by the evaluation of joint reaction forces. In *CMBEBIH 2017: Proceedings of the International Conference on Medical and Biological Engineering 2017*, pages 205–211. Springer.

Awad, L. N., Bae, J., Kudzia, P., Long, A., Hendron, K., Holt, K. G., O'Donnell, K., Ellis, T. D., and Walsh, C. J. (2017a). Reducing Circumduction and Hip Hiking During Hemiparetic Walking Through Targeted Assistance of the Paretic Limb Using a Soft Robotic Exosuit. *American Journal of Physical Medicine & Rehabilitation*, 96(10):S157–S164.

Awad, L. N., Bae, J., O'Donnell, K., Hendron, K. L., Sloot, L., Siviy, C., Kudzia, P., Ellis, T. D., and Walsh, C. J. (2017b). Soft exosuits increase walking speed and distance after stroke. In *2017 International Symposium on Wearable Robotics and Rehabilitation (WeRob)*, pages 1–2. IEEE.

Awad, L. N., Binder-Macleod, S. A., Pohlig, R. T., and Reisman, D. S. (2015a). Paretic Propulsion and Trailing Limb Angle Are Key Determinants of Long-Distance Walking Function After Stroke. *Neurorehabilitation and Neural Repair*, 29(6):499–508.

Awad, L. N., Palmer, J. A., Pohlig, R. T., Binder-Macleod, S. A., and Reisman, D. S. (2015b). Walking speed and step length asymmetry modify the energy cost of walking after stroke. *Neurorehabilitation and Neural Repair*, 29(5):416–423.

Awad, L. N., Reisman, D. S., Pohlig, R. T., and Binder-Macleod, S. A. (2016). Reducing the Cost of Transport and Increasing Walking Distance After Stroke: A Randomized Controlled Trial on Fast Locomotor Training Combined With Functional Electrical Stimulation. *Neurorehabilitation and Neural Repair*, 30(7):661–670.

Balaban, B. and Tok, F. (2014). Gait Disturbances in Patients With Stroke. *American Academy of Physical Medicine and Rehabilitation*, 6(7):635–642.

Begg, R. and Kamruzzaman, J. (2005). A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data. *Journal of Biomechanics*, 38(3):401–408.

Begg, R., Palaniswami, M., and Owen, B. (2005). Support vector machines for automated gait classification. *IEEE Transactions on Biomedical Engineering*, 52(5):828–838.

Bejarano, N. C., Ambrosini, E., Pedrocchi, A., Ferrigno, G., Monticone, M., and Ferrante, S. (2015). A Novel Adaptive, Real-Time Algorithm to Detect Gait Events From Wearable Sensors. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(3):413–422.

Berrigan, D., Carroll, D. D., Fulton, J. E., Galuska, D. A., Brown, D. R., Dorn, J. M., Armour, B., and Paul, P. (2012). Vital signs: Walking among adults-United States, 2005 and 2010. *Morbidity and Mortality Weekly Report*, 61(31):595–601.

Bethoux, F. and Bennett, S. (2011). Evaluating Walking in Patients with Multiple Sclerosis: Which Assessment Tools Are Useful in Clinical Practice? *International Journal of MS Care*, 13(1):4–14.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern Recognition and Machine Learning*, volume 4. Springer.

Boutaayamou, M., Schwartz, C., Stamatakis, J., Denoël, V., Maquet, D., Forthomme, B., Croisier, J.-L., Macq, B., Verly, J. G., Garraux, G., and Brüls, O. (2015). Development and validation of an accelerometer-based method for quantifying gait events. *Medical Engineering & Physics*, 37(2):226–232.

Bowden, M. G., Balasubramanian, C. K., Behrman, A. L., and Kautz, S. A. (2008). Validation of a Speed-Based Classification System Using Quantitative Measures of Walking Performance Poststroke. *Neurorehabilitation and Neural Repair*, 22(6):672–675.

Bowden, M. G., Balasubramanian, C. K., Neptune, R. R., and Kautz, S. A. (2006). Anterior-posterior ground reaction forces as a measure of paretic leg contribution in hemiparetic walking. *Stroke*, 37(3):872–876.

Bowden, M. G., Behrman, A. L., Neptune, R. R., Gregory, C. M., and Kautz, S. A. (2013). Locomotor rehabilitation of individuals with chronic stroke: Difference between responders and nonresponders. *Archives of Physical Medicine and Rehabilitation*, 94(5):856–862.

Browne, M. G. and Franz, J. R. (2017). Does dynamic stability govern propulsive force generation in human walking? *Royal Society Open Science*, 4(11):171673.

Browne, M. G. and Franz, J. R. (2019). Ankle power biofeedback attenuates the distal-to-proximal redistribution in older adults. *Gait & Posture*, 71:44–49.

Brunton, S. L. and Kutz, J. N. (2021). *Data-Driven Science and Engineering*. Cambridge University Press.

Caldas, R., Mundt, M., Potthast, W., Buarque de Lima Neto, F., and Markert, B. (2017). A systematic review of gait analysis methods based on inertial sensors and adaptive algorithms. *Gait & Posture*, 57(February):204–210.

Cappozzo, A., Figura, F., Marchetti, M., and Pedotti, A. (1976). The interplay of muscular and external forces in human ambulation. *Journal of Biomechanics*, 9(1):35–43.

Catalfamo, P., Ghoussayni, S., and Ewins, D. (2010). Gait Event Detection on Level Ground and Incline Walking Using a Rate Gyroscope. *Sensors*, 10(6):5683–5702.

Catalfamo, P., Moser, D., Ghoussayni, S., and Ewins, D. (2008). Detection of gait events using an F-Scan in-shoe pressure measurement system. *Gait & Posture*, 28(3):420–426.

Chen, G. and Patten, C. (2008). Joint moment work during the stance-to-swing transition in hemiparetic subjects. *Journal of Biomechanics*, 41(4):877–883.

Chen, G., Patten, C., Kothari, D. H., and Zajac, F. E. (2005). Gait differences between individuals with post-stroke hemiparesis and non-disabled controls at matched speeds. *Gait & Posture*, 22(1):51–56.

Clark, D. J., Ting, L. H., Zajac, F. E., Neptune, R. R., and Kautz, S. A. (2010). Merging of Healthy Motor Modules Predicts Reduced Locomotor Performance and Muscle Coordination Complexity Post-Stroke. *Journal of Neurophysiology*, 103(2):844–857.

Collins, S. (2005). Efficient Bipedal Robots Based on Passive-Dynamic Walkers. *Science*, 307(5712):1082–1085.

Combs, S. A., Van Puymbroeck, M., Altenburger, P. A., Miller, K. K., Dierks, T. A., and Schmid, A. A. (2013). Is walking faster or walking farther more important to persons with chronic stroke? *Disability and Rehabilitation*, 35(10):860–867.

Crea, S., De Rossi, S. M., Donati, M., Reberšek, P., Novak, D., Vitiello, N., Lenzi, T., Podobnik, J., Munih, M., and Carrozza, M. C. (2012). Development of gait segmentation methods for wearable foot pressure sensors. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pages 5018–5021.

Cutting, J. E. and Kozlowski, L. T. (1977). Recognizing friends by their walk: Gait perception without familiarity cues. *Bulletin of the Psychonomic Society*, 9:353–356.

Cuzzolin, F., Sapienza, M., Esser, P., Saha, S., Franssen, M. M., Collett, J., and Dawes, H. (2017). Metric learning for Parkinsonian identification from IMU gait measurements. *Gait & Posture*, 54:127–132.

Danks, K. A., Roos, M. A., McCoy, D., and Reisman, D. S. (2014). A step activity monitoring program improves real world walking activity post stroke. *Disability and Rehabilitation*, 36(26):2233–2236.

Dean, C. M., Rissel, C., Sherrington, C., Sharkey, M., Cumming, R. G., Lord, S. R., Barker, R. N., Kirkham, C., and O'Rourke, S. (2012). Exercise to Enhance Mobility and Prevent Falls After Stroke : The Community Stroke Club Randomized Trial. *Neurorehabilitation and Neural Repair*, 26(9):1046–1057.

Dejnabadi, H., Jolles, B. M., and Aminian, K. (2005). A new approach to accurate measurement of uniaxial joint angles based on a combination of accelerometers and gyroscopes. *IEEE Transactions on Biomedical Engineering*, 52(8):1478–1484.

Della Croce, U., Riley, P. O., Lelas, J. L., and Kerrigan, D. C. (2001). A refined view of the determinants of gait. *Gait & Posture*, 14(2):79–84.

Detrembleur, C., Dierick, F., Stoquart, G., Chantraine, F., and Lejeune, T. (2003). Energy cost, mechanical work, and efficiency of hemiparetic walking. *Gait & Posture*, 18(2):47–55.

Dickstein, R. (2008). Rehabilitation of Gait Speed After Stroke: A Critical Review of Intervention Approaches. *Neurorehabilitation and Neural Repair*, 22(6):649–660.

Donath, L., Faude, O., Lichtenstein, E., Pagenstert, G., Nüesch, C., and Mündermann, A. (2016). Mobile inertial sensor based gait analysis: Validity and reliability of spatiotemporal gait characteristics in healthy seniors. *Gait & Posture*, 49:371–374.

Donelan, J., Kram, R., and Kuo, A. D. (2002a). Simultaneous positive and negative external mechanical work in human walking. *Journal of Biomechanics*, 35(1):117–124.

Donelan, J. M., Kram, R., and Kuo, A. D. (2002b). Mechanical work for step-to-step transitions is a major determinant of the metabolic cost of human walking. *Journal of Experimental Biology*, 205(23):3717–3727.

Duncan, P. W., Sullivan, K. J., Behrman, A. L., Azen, S. P., Wu, S. S., Nadeau, S. E., Dobkin, B. H., Rose, D. K., Tilson, J. K., Cen, S., and Hayden, S. K. (2011). Body-Weight–Supported Treadmill Rehabilitation after Stroke. *New England Journal of Medicine*, 364(21):2026–2036.

Ellis, R. G., Howard, K. C., and Kram, R. (2013). The metabolic and mechanical costs of step time asymmetry in walking. *Proceedings of the Royal Society B: Biological Sciences*, 280:20122784–20122784.

English, C., Manns, P. J., Tucak, C., and Bernhardt, J. (2014). Physical Activity and Sedentary Behaviors in People With Stroke Living in the Community: A Systematic Review. *Physical Therapy*, 94(2):185–196.

Evans, R. L. and Arvind, D. K. (2014). Detection of gait phases using orient specks for mobile clinical gait analysis. In *2014 11th International Conference on Wearable and Implantable Body Sensor Networks*, pages 149–154. IEEE.

Farah, J. D., Baddour, N., and Lemaire, E. D. (2017). Gait phase detection from thigh kinematics using machine learning techniques. In *2017 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pages 263–268, Rochester, MN, USA. IEEE.

Farris, D. J., Hampton, A., Lewek, M. D., and Sawicki, G. S. (2015). Revisiting the mechanics and energetics of walking in individuals with chronic hemiparesis following stroke: From individual limbs to lower limb joints. *Journal of Neuro-Engineering and Rehabilitation*, 12(1):1–12.

Figueiredo, J., Santos, C. P., and Moreno, J. C. (2018). Automatic recognition of gait patterns in human motor disorders using machine learning: A review. *Medical Engineering & Physics*, 53:1–12.

Franceschini, M., Rampello, A., Agosti, M., Massucci, M., Bovolenta, F., and Sale, P. (2013). Walking Performance: Correlation between Energy Cost of Walking and Walking Participation. New Statistical Approach Concerning Outcome Measurement. *PLoS ONE*, 8(2):e56669.

Franz, J. R. (2016). The age-associated reduction in propulsive power generation in walking. *Exercise and Sport Sciences Reviews*, 44(4):129–136.

Franz, J. R. and Kram, R. (2013a). Advanced age affects the individual leg mechanics of level, uphill, and downhill walking. *Journal of Biomechanics*, 46(3):535–540.

Franz, J. R. and Kram, R. (2013b). How does age affect leg muscle activity/coactivity during uphill and downhill walking? *Gait & Posture*, 37(3):378–384.

Franz, J. R. and Kram, R. (2014). Advanced age and the mechanics of uphill walking: A joint-level , inverse dynamic analysis. *Gait & Posture*, 39(1):135–140.

Franz, J. R., Maletis, M., and Kram, R. (2014). Real-time feedback enhances forward propulsion during walking in old adults. *Clinical Biomechanics*, 29(1):68–74.

Fulk, G. D., He, Y., Boyne, P., and Dunning, K. (2017). Predicting Home and Community Walking Activity Poststroke. *Stroke*, 48(2):406–411.

Gafurov, D. (2007). A survey of biometric gait recognition: Approaches, security and challenges. In *Annual Norwegian Computer Science Conference*, pages 19–21. Annual Norwegian Computer Science Conference Norway.

Gamito, P., Oliveira, J., Coelho, C., Morais, D., Lopes, P., Pacheco, J., Brito, R., Soares, F., Santos, N., and Barata, A. F. (2017). Cognitive training on stroke patients via virtual reality-based serious games. *Disability and Rehabilitation*, 39(4):385–388.

Genthe, K., Schenck, C., Eicholtz, S., Zajac-Cox, L., Wolf, S., and Kesar, T. M. (2018). Effects of real-time gait biofeedback on paretic propulsion and gait biomechanics in individuals post-stroke. *Topics in Stroke Rehabilitation*, 25(3):186–193.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT press.

Gordon, K. E., Ferris, D. P., and Kuo, A. D. (2009). Metabolic and mechanical energy costs of reducing vertical center of mass movement during gait. *Archives of Physical Medicine and Rehabilitation*, 90(1):136–144.

Guo, Y., Storm, F., Zhao, Y., Billings, S., Pavic, A., Mazzà, C., and Guo, L.-Z. (2017). A New Proxy Measurement Algorithm with Application to the Estimation of Vertical Ground Reaction Forces Using Wearable Sensors. *Sensors*, 17(10):2181.

Halliday, S. E., Winter, D. A., Frank, J. S., Patla, A. E., and Prince, F. (1998). The initiation of gait in young, elderly, and Parkinson's disease subjects. *Gait & Posture*, 8(1):8–14.

Hass, C. J., Buckley, T. A., Pitsikoulis, C., and Barthelemy, E. J. (2012). Progressive resistance training improves gait initiation in individuals with Parkinson's disease. *Gait & Posture*, 35(4):669–673.

Hass, C. J., Waddell, D. E., Fleming, R. P., Juncos, J. L., and Gregor, R. J. (2005). Gait Initiation and Dynamic Balance Control in Parkinson 's Disease. *Archives of Physical Medicine and Rehabilitation*, 86(11):2172–2176.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

Hof, A., Elzinga, H., Grimmius, W., and Halbertsma, J. (2002). Speed dependence of averaged EMG profiles in walking. *Gait & Posture*, 16(1):78–86.

Hof, A. L., Nauta, J., van der Knaap, E. R., Schallig, M. A., and Struwe, D. (1992). Calf muscle work and segment energy changes in human treadmill walking. *Journal of Electromyography and Kinesiology*, 2(4):203–216.

Horak, F., King, L., and Mancini, M. (2015). Role of Body-Worn Movement Monitor Technology for Balance and Gait Rehabilitation. *Physical Therapy*, 95(3):461–470.

Hsiao, H., Awad, L. N., Palmer, J. A., Higginson, J. S., and Binder-Macleod, S. A. (2016a). Contribution of Paretic and Nonparetic Limb Peak Propulsive Forces to Changes in Walking Speed in Individuals Poststroke. *Neurorehabilitation and Neural Repair*, 30(8):743–752.

Hsiao, H., Knarr, B. A., Higginson, J., and Binder-Macleod, S. A. (2015a). The Relative Contribution of Ankle Moment and Trailing Limb Angle to Propulsive Force during Gait. *Human Movement Science*, pages 212–221.

Hsiao, H., Knarr, B. A., Higginson, J. S., and Binder-Macleod, S. A. (2015b). Mechanisms to increase propulsive force for individuals poststroke. *Journal of Neuro-Engineering and Rehabilitation*, 12(1).

Hsiao, H., Zabielski, T. M., Palmer, J. A., Higginson, J. S., and Binder-Macleod, S. A. (2016b). Evaluation of measurements of propulsion used to reflect changes in walking speed in individuals poststroke. *Journal of Biomechanics*, 49(16):4107–4112.

Jacobs, D. A. and Ferris, D. P. (2015). Estimation of ground reaction forces and ankle moment with multiple , low-cost sensors. *Journal of NeuroEngineering and Rehabilitation*, 12(1).

Jonkers, I., Delp, S., and Patten, C. (2009). Capacity to increase walking speed is limited by impaired hip and ankle power generation in lower functioning persons post-stroke. *Gait & Posture*, 29(1):129–137.

Jonsdottir, J., Cattaneo, D., Recalcati, M., Regola, A., Rabuffetti, M., Ferrarin, M., and Casiraghi, A. (2010). Task-Oriented Biofeedback to Improve Gait in Individuals With Chronic Stroke: Motor Learning Approach. *Neurorehabilitation and Neural Repair*, 24(5):478–485.

Kaczmarczyk, K., Wit, A., Krawczyk, M., and Zaborski, J. (2009). Gait classification in post-stroke patients using artificial neural networks. *Gait & Posture*, 30(2):207–210.

Karatsidis, A., Bellusci, G., Schepers, H., de Zee, M., Andersen, M., and Veltink, P. (2016). Estimation of Ground Reaction Forces and Moments During Gait Using Only Inertial Motion Capture. *Sensors*, 17(1):75.

Karatsidis, A., Jung, M., Schepers, H. M., Bellusci, G., de Zee, M., Veltink, P. H., and Andersen, M. S. (2019). Musculoskeletal model-based inverse dynamic analysis under ambulatory conditions using inertial motion capture. *Medical Engineering & Physics*, 65:68–77.

Karatsidis, A., Richards, R. E., Konrath, J. M., van den Noort, J. C., Schepers, H. M., Bellusci, G., Harlaar, J., and Veltink, P. H. (2018). Validation of wearable visual feedback for retraining foot progression angle using inertial sensors and an augmented reality headset. *Journal of NeuroEngineering and Rehabilitation*, 15(1).

Kavanagh, J. J. and Menz, H. B. (2008). Accelerometry: A technique for quantifying movement patterns during walking. *Gait & Posture*, 28(1):1–15.

Kerrigan, D., Riley, P. O., Lelas, J. L., and Croce, U. D. (2001). Quantification of pelvic rotation as a determinant of gait. *Archives of Physical Medicine and Rehabilitation*, 82(2):217–220.

Kesar, T. M., Perumal, R., Reisman, D. S., Jancosko, A., Rudolph, K. S., Higginson, J. S., and Binder-Macleod, S. A. (2009). Functional Electrical Stimulation of Ankle Plantarflexor and Dorsiflexor Muscles: Effects on Poststroke Gait. *Stroke*, 40(12):3821–3827.

Kieron Jie-Han, N., Gouwanda, D., Gopalai, A. A., and Yu Zheng, C. (2018). Estimation of Vertical Ground Reaction Force during Running using Neural Network Model and Uniaxial Accelerometer. *Journal of Biomechanics*, 76:269–273.

Kitago, T. and Krakauer, J. W. (2013). Motor learning principles for neurorehabilitation. In *Handbook of Clinical Neurology*, volume 110, pages 93–103. Elsevier.

Kramer, S., Johnson, L., Bernhardt, J., and Cumming, T. (2016). Energy Expenditure and Cost During Walking After Stroke: A Systematic Review. *Archives of Physical Medicine and Rehabilitation*, 97(4):619–632.e1.

Kruger, J., Ham, S. A., Berrigan, D., and Ballard-Barbash, R. (2008). Prevalence of transportation and leisure walking among U.S. adults. *Preventive Medicine*, 47(3):329–334.

Kuhman, D. and Hurt, C. P. (2019). The timing of locomotor propulsion in healthy adults walking at multiple speeds. *Human Movement Science*, 68:102524.

Kuo, A. D. (2002). Energetics of Actively Powered Locomotion Using the Simplest Walking Model. *Journal of Biomechanical Engineering*, 124(1):113–120.

Kuo, A. D. and Donelan, J. M. (2010). Dynamic Principles of Gait and Their Clinical Implications. In *Physical Therapy*, volume 90, pages 157–174. Oxford University Press.

Kuo, A. D., Donelan, J. M., and Ruina, A. (2005). Energetic Consequences of Walking Like an Inverted Pendulum: Step-to-Step Transitions:. *Exercise and Sport Sciences Reviews*, 33(2):88–97.

Lapointe, R., Lajoie, Y., Serresse, O., and Barbeau, H. (2001). Functional community ambulation requirements in incomplete spinal cord injured subjects. *Spinal Cord*, 39(6):327–335.

Laver, K. E., Lange, B., George, S., Deutsch, J. E., Saposnik, G., and Crotty, M. (2017). Virtual reality for stroke rehabilitation. *Cochrane Database of Systematic Reviews*, (11).

Lee, I.-M. and Buchner, D. M. (2008). The Importance of Walking to Public Health. *Medicine & Science in Sports & Exercise*, 40(7):S512–S518.

Li, Z., Han, X.-G., Sheng, J., and Ma, S.-J. (2016). Virtual reality for improving balance in patients after stroke: A systematic review and meta-analysis. *Clinical Rehabilitation*, 30(5):432–440.

Lim, H., Kim, B., and Park, S. (2019). Prediction of Lower Limb Kinetics and Kinematics during Walking by a Single IMU on the Lower Back Using Machine Learning. *Sensors*, 20(1):130.

Lim, S. B., Horslen, B. C., Davis, J. R., Allum, J. H., and Carpenter, M. G. (2016). Benefits of multi-session balance and gait training with multi-modal biofeedback in healthy older adults. *Gait & Posture*, 47:10–17.

Lipfert, S., Gunther, M., Renjewski, D., and Seyfarth, A. (2014). Impulsive ankle push-off powers leg swing in human walking. *Journal of Experimental Biology*, 217(10):1831–1831.

Mahon, C. E., Farris, D. J., Sawicki, G. S., and Lewek, M. D. (2015). Individual limb mechanical analysis of gait following stroke. *Journal of Biomechanics*, 48(6):984–989.

Mannini, A., Genovese, V., and Sabatini, A. M. (2014). Online Decoding of Hidden Markov Models for Gait Event Detection Using Foot-Mounted Gyroscopes. *IEEE Journal of Biomedical and Health Informatics*, 18(4):1122–1130.

Mannini, A. and Sabatini, A. M. (2011). A hidden Markov model-based technique for gait segmentation using a foot-mounted gyroscope. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4369–4373.

Mannini, A. and Sabatini, A. M. (2012). Gait phase detection and discrimination between walking–jogging activities using hidden Markov models applied to foot motion data from a gyroscope. *Gait & Posture*, 36(4):657–661.

Mannini, A., Trojaniello, D., Cereatti, A., and Sabatini, A. (2016). A Machine Learning Framework for Gait Classification Using Inertial Sensors: Application to Elderly, Post-Stroke and Huntington's Disease Patients. *Sensors*, 16(1):134.

Marsland, S. (2011). *Machine Learning: An Algorithmic Perspective*. Chapman and Hall/CRC, 1st edition.

Martin, M., Shinberg, M., Kuchibhatla, M., Ray, L., Carollo, J. J., and Schenkman, M. L. (2002). Gait Initiation in Community-Dwelling Adults With Parkinson Disease: Comparison With Older and Younger Adults Without the Disease. *Physical Therapy*, 82(6):566–577.

Massaad, F., Lejeune, T. M., and Detrembleur, C. (2010). Reducing the Energy Cost of Hemiparetic Gait Using Center of Mass Feedback: A Pilot Study. *Neurorehabilitation and Neural Repair*, 24(4):338–347.

Mayagoitia, R. E., Nene, A. V., and Veltink, P. H. (2002). Accelerometer and rate gyroscope measurement of kinematics: An inexpensive alternative to optical motion analysis systems. *Journal of Biomechanics*, 35(4):537–542.

McCain, E. M., Dick, T. J. M., Giest, T. N., Nuckols, R. W., Lewek, M. D., Saul, K. R., and Sawicki, G. S. (2019). Mechanics and energetics of post-stroke walking aided by a powered ankle exoskeleton with speed-adaptive myoelectric control. *Journal of NeuroEngineering and Rehabilitation*, 16(1).

McGinnis, R. S., Mahadevan, N., Moon, Y., Seagers, K., Sheth, N., Wright, J. A., DiCristofaro, S., Silva, I., Jortberg, E., Ceruolo, M., Pindado, J. A., Sosnoff, J., Ghaffari, R., and Patel, S. (2017). A machine learning approach for gait speed estimation using skin-mounted wearable sensors: From healthy controls to individuals with multiple sclerosis. *PLOS ONE*, 12(6):e0178366.

McGreer, T. (1990). Passive Dynamic Walking. *The International Journal of Robotic Research*, 9(2):62–82.

Mian, O. S., Thom, J. M., Ardigò, L. P., Narici, M. V., and Minetti, A. E. (2006). Metabolic cost, mechanical work, and efficiency during walking in young and older men. *Acta Physiologica*, 186(2):127–139.

Michael, K. M., Allen, J. K., and Macko, R. F. (2005). Reduced Ambulatory Activity After Stroke: The Role of Balance, Gait, and Cardiovascular Fitness. *Archives of Physical Medicine and Rehabilitation*, 86(8):1552–1556.

Mijailoviü, N., Gavriloviü, M., and Rafajlovi ü, S. (2009). Gait Phases Recognition from Accelerations and Ground Reaction Forces: Application of Neural Networks. *Telfor Journal*, 1(1):3.

Miyazaki, T., Kawada, M., Nakai, Y., Kiyama, R., and Yone, K. (2019). Validity of Measurement for Trailing Limb Angle and Propulsion Force during Gait Using a Magnetic Inertial Measurement Unit. *BioMed Research International*, 2019:1–8.

Moore, J. L., Roth, E. J., Killian, C., and Hornby, T. G. (2010). Locomotor Training Improves Daily Stepping Activity and Gait Efficiency in Individuals Poststroke Who Have Reached a " Plateau" in Recovery. *Stroke*, 41(1):129–135.

Moore, S. A., Hickey, A., Lord, S., Del Din, S., Godfrey, A., and Rochester, L. (2017). Comprehensive measurement of stroke gait characteristics with a single accelerometer in the laboratory and community: A feasibility, validity and reliability study. *Journal of NeuroEngineering and Rehabilitation*, 14(1).

Neptune, R., Kautz, S., and Zajac, F. (2001). Contributions of the individual ankle plantar flexors to support, forward progression and swing initiation during walking. *Journal of Biomechanics*, 34(11):1387–1398.

Novak, D., Reberšek, P., De Rossi, S. M. M., Donati, M., Podobnik, J., Beravs, T., Lenzi, T., Vitiello, N., Carrozza, M. C., and Munih, M. (2013). Automated detection of gait initiation and termination using wearable sensors. *Medical Engineering and Physics*, 35(12):1713–1720.

Ohtaki, Y., Sagawa, K., and Inooka, H. (2001). A method for gait analysis in a daily living environment by body-mounted instruments. *JSME International Journal*

Series C Mechanical Systems, Machine Elements and Manufacturing, 44(4):1125–1132.

Ortega, J. D. and Farley, C. T. (2005). Minimizing center of mass vertical movement increases metabolic cost in walking. *Journal of Applied Physiology*, 99:9.

Pappas, I. P., Keller, T., Mangold, S., Popovic, M. R., Dietz, V., and Morari, M. (2004). A reliable gyroscope-based gait-phase detection sensor embedded in a shoe insole. *Sensors*, 4(2):268–274.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., and Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830.

Penke, K., Scott, K., Sinskey, Y., and Lewek, M. D. (2019). Propulsive Forces Applied to the Body's Center of Mass Affect Metabolic Energetics Poststroke. *Archives of Physical Medicine and Rehabilitation*, 100(6):1068–1075.

Perez-Marcos, D., Chevalley, O., Schmidlin, T., Garipelli, G., Serino, A., Vuadens, P., Tadi, T., Blanke, O., and Millán, J. d. R. (2017). Increasing upper limb training intensity in chronic stroke using embodied virtual reality: A pilot study. *Journal of NeuroEngineering and Rehabilitation*, 14(1).

Perry, J. and Davids, J. R. (1992). Gait analysis: Normal and pathological function. *Journal of Pediatric Orthopaedics*, 12(6):815.

Peruzzi, A., Della Croce, U., and Cereatti, A. (2011). Estimation of stride length in level walking using an inertial measurement unit attached to the foot: A validation of the zero velocity assumption during stance. *Journal of Biomechanics*, 44(10):1991–1994.

Phadke, C. P. (2012). Immediate Effects of a Single Inclined Treadmill Walking Session on Level Ground Walking in Individuals After Stroke :. *American Journal of Physical Medicine & Rehabilitation*, 91(4):337–345.

Pieper, N. L., Lewek, M. D., and Franz, J. R. (2019). Can shank acceleration provide a clinically feasible surrogate for individual limb propulsion during walking? *Journal of Biomechanics*, 98:109449.

Potluri, S., Chandran, A. B., Diedrich, C., and Schega, L. (2019). Machine Learning based Human Gait Segmentation with Wearable Sensor Platform. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 588–594.

Rand, D., Eng, J. J., Tang, P.-F., Hung, C., and Jeng, J.-S. (2010). Daily physical activity and its contribution to the health-related quality of life of ambulatory individuals with chronic stroke. *Health and Quality of Life Outcomes*, 8(1):80.

Reisman, D. S., Rudolph, K. S., and Farquhar, W. B. (2009). Influence of speed on walking economy poststroke. *Neurorehabilitation and Neural Repair*, 23(6):529–34.

Revi, D. A., Alvarez, A. M., Walsh, C. J., De Rossi, S. M., and Awad, L. N. (2020). Indirect measurement of anterior-posterior ground reaction forces using a minimal set of wearable inertial sensors: From healthy to hemiparetic walking. *Journal of neuroengineering and rehabilitation*, 17:1–13.

Rueterbories, J., Spaich, E. G., and Andersen, O. K. (2014). Gait event detection for use in FES rehabilitation by radial and tangential foot accelerations. *Medical Engineering & Physics*, 36(4):502–508.

Ryu, H. X. and Park, S. (2018). Estimation of unmeasured ground reaction force data based on the oscillatory characteristics of the center of mass during human walking. *Journal of Biomechanics*, 71:135–143.

Saunders, M., Inman, VT., and Eberhart, HD. (1953). The major determinant in normal and pathological gait. *Journal of Bones and Joint Surgery*, 35(3):543–558.

Schenck, C., Bakke, D., and Besier, T. (2019). Haptic biofeedback induces changes in ankle push-off during walking. *Gait & Posture*, 74:76–82.

Schenck, C. and Kesar, T. M. (2017). Effects of unilateral real-time biofeedback on propulsive forces during gait. *Journal of NeuroEngineering and Rehabilitation*, 14(1):1–10.

Seel, T., Raisch, J., and Schauer, T. (2014). IMU-based joint angle measurement for gait analysis. *Sensors*, 14(4):6891–6909.

Seel, T., Schauer, T., and Raisch, J. (2012). Joint axis and position estimation from inertial measurement data by exploiting kinematic constraints. In *2012 IEEE International Conference on Control Applications*, pages 45–49.

Selles, R. W., Formanoy, M. A. G., Bussmann, J. B. J., Janssens, P. J., and Stam, H. J. (2005). Automated estimation of initial and terminal contact timing using accelerometers; development and validation in transtibial amputees and controls. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13(1):81–88.

Shahabpoor, E. and Pavic, A. (2018). Estimation of vertical walking ground reaction force in real-life environments using single IMU sensor. *Journal of Biomechanics*, 79:181–190.

Shetty, S. and Rao, Y. S. (2016). SVM based machine learning approach to identify Parkinson's disease using gait analysis. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 2, pages 1–5.

Sigrist, R., Rauter, G., Riener, R., and Wolf, P. (2013). Augmented visual, auditory, haptic, and multimodal feedback in motor learning: A review. *Psychonomic Bulletin & Review*, 20(1):21–53.

Sockol, M. D., Raichlen, D. A., and Pontzer, H. (2007). Chimpanzee locomotor energetics and the origin of human bipedalism. *Proceedings of the National Academy of Sciences*, 104:12265–12269.

Sprager, S. and Juric, M. (2015). Inertial Sensor-Based Gait Recognition: A Review. *Sensors*, 15(9):22089–22127.

Stanton, R., Ada, L., Dean, C. M., and Preston, E. (2017). Biofeedback improves performance in lower limb activities more than usual therapy in people following stroke: A systematic review. *Journal of Physiotherapy*, 63(1):11–16.

Steele, K. M., Rozumalski, A., and Schwartz, M. H. (2015). Muscle synergies and complexity of neuromuscular control during gait in cerebral palsy. *Developmental Medicine and Child Neurology*, 57(12):1176–1182.

Sullivan, J. E., Espe, L. E., Kelly, A. M., Veilbig, L. E., and Kwasny, M. J. (2014). Feasibility and Outcomes of a Community-Based, Pedometer-Monitored Walking Program in Chronic Stroke: A Pilot Study. *Topics in Stroke Rehabilitation*, 21(2):101–110.

Taborri, J., Palermo, E., Rossi, S., and Cappa, P. (2016). Gait Partitioning Methods: A Systematic Review. *Sensors*, 16(1):66.

Taborri, J., Rossi, S., Palermo, E., and Cappa, P. (2015a). A HMM distributed classifier to control robotic knee module of an active orthosis. In *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*, pages 277–282, Singapore, Singapore. IEEE.

Taborri, J., Rossi, S., Palermo, E., Patanè, F., and Cappa, P. (2014). A Novel HMM Distributed Classifier for the Detection of Gait Phases by Means of a Wearable Inertial Sensor Network. *Sensors*, 14(9):16212–16234.

Taborri, J., Scalona, E., Palermo, E., Rossi, S., and Cappa, P. (2015b). Validation of Inter-Subject Training for Hidden Markov Models Applied to Gait Phase Detection in Children with Cerebral Palsy. *Sensors*, 15(9):24514–24529.

Tahir, N. M. and Manap, H. H. (2012). Parkinson Disease Gait Classification based on Machine Learning Approach. *Journal of Applied Sciences*, 12:180–185.

Takahashi, K. Z., Lewek, M. D., and Sawicki, G. S. (2015). A neuromechanics-based powered ankle exoskeleton to assist walking post-stroke: A feasibility study. *Journal of NeuroEngineering and Rehabilitation*, 12(1):23.

Tate, J. J. and Milner, C. E. (2010). Real-Time Kinematic, Temporospatial, and Kinetic Biofeedback During Gait Retraining in Patients: A Systematic Review. *Physical Therapy*, 90(8):1123–1134.

Trojaniello, D., Cereatti, A., and Della Croce, U. (2014). Accuracy, sensitivity and robustness of five different methods for the estimation of gait temporal parameters using a single inertial sensor mounted on the lower trunk. *Gait & Posture*, 40(4):487–492.

Turns, L. J., Neptune, R. R., and Kautz, S. A. (2007). Relationships Between Muscle Activity and Anteroposterior Ground Reaction Forces in Hemiparetic Walking. *Archives of Physical Medicine and Rehabilitation*, 88(9):1127–1135.

van Gelder, L. M., Barnes, A., Wheat, J. S., and Heller, B. W. (2018). The use of biofeedback for gait retraining: A mapping review. *Clinical Biomechanics*, 59:159–166.

van Swigchem, R., Roerdink, M., Weerdesteyn, V., Geurts, A. C., and Daffertshofer, A. (2014). The Capacity to Restore Steady Gait After a Step Modification Is Reduced in People With Poststroke Foot Drop Using an Ankle-Foot Orthosis. *Physical Therapy*, 94(5):654–663.

Vera-Rodriguez, R., Mason, J. S., Fierrez, J., and Ortega-Garcia, J. (2012). Comparative analysis and fusion of spatiotemporal information for footstep recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):823–834.

Virani, S. S., Alonso, A., Benjamin, E. J., Bittencourt, M. S., Callaway, C. W., Carson, A. P., Chamberlain, A. M., Chang, A. R., Cheng, S., and Delling, F. N. (2020). Heart disease and stroke statistics—2020 update: A report from the American Heart Association. *Circulation*, 141(9):e139–e596.

Vistamehr, A., Kautz, S. A., and Neptune, R. R. (2014). The influence of solid ankle-foot-orthoses on forward propulsion and dynamic balance in healthy adults during walking. *Clinical Biomechanics*, 29(5):583–589.

Wang, L., Tan, T., Ning, H., and Hu, W. (2003). Silhouette analysis-based gait recognition for human identification. *IEEE transactions on pattern analysis and machine intelligence*, 25(12):1505–1518.

Wert, D. M., Brach, J. S., Perera, S., and VanSwearingen, J. (2013). The association between energy cost of walking and physical function in older adults. *Archives of Gerontology and Geriatrics*, 57(2):198–203.

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

Winter, D. A. and Robertson, D. G. E. (1978). Joit torque and energy patterns in normal gait. *Biological Cybernetics*, 29(3):137–142.

Wouda, F. J., Giuberti, M., Bellusci, G., Maartens, E., Reenalda, J., van Beijnum, B.-J. F., and Veltink, P. H. (2018). Estimation of Vertical Ground Reaction Forces and Sagittal Knee Kinematics During Running Using Three Inertial Sensors. *Frontiers in Physiology*, 9.

Wutzke, C. J., Sawicki, G. S., and Lewek, M. D. (2012). The influence of a unilateral fixed ankle on metabolic and mechanical demands during walking in unimpaired young adults. *Journal of Biomechanics*, 45(14):2405–2410.

Yang, S., Zhang, J.-T., Novak, A. C., Brouwer, B., and Li, Q. (2013). Estimation of spatio-temporal parameters for post-stroke hemiparetic gait using inertial sensors. *Gait & Posture*, 37(3):354–358.

Yen, S.-C., Landry, J. M., and Wu, M. (2014). Augmented multisensory feedback enhances locomotor adaptation in humans with incomplete spinal cord injury. *Human Movement Science*, 35:80–93.

Zelik, K. E. and Adamczyk, P. G. (2016). A unified perspective on ankle push-off in human walking. *The Journal of Experimental Biology*, 219(23):3676–3683.

Zelik, K. E., Takahashi, K. Z., and Sawicki, G. S. (2015). Six degree-of-freedom analysis of hip, knee, ankle and foot provides updated understanding of biomechanical work during human walking. *Journal of Experimental Biology*, 218(6):876–886.

Zhang, W., Smuck, M., Legault, C., Ith, M. A., Muaremi, A., and Aminian, K. (2018). Gait Symmetry Assessment with a Low Back 3D Accelerometer in Post-Stroke Patients. *Sensors*, 18(10):3322.

Zijlstra, A., Mancini, M., Chiari, L., and Zijlstra, W. (2010). Biofeedback for training balance and mobility tasks in older populations: A systematic review. *Journal of NeuroEngineering and Rehabilitation*, 7(1).

Zollikofer, C. P., de León, M. S. P., Lieberman, D. E., Guy, F., Pilbeam, D., Likius, A., Mackaye, H. T., Vignaud, P., and Brunet, M. (2005). Virtual cranial reconstruction of Sahelanthropus tchadensis. *Nature*, 434(7034):755–759.

Andre Alvarez

Email | GitHub | LinkedIn

Experience

Neuromotor Recovery Lab — Boston, MA
Doctoral Researcher — Sep 2017 to Present

- **Automated data extraction and aggregation** by building and implementing programming pipelines.
- **Streamlined data analysis** by developing predictive models based on machine learning algorithms, including regression, classification and time series analysis.
- **Trained, onboarded and managed performance** of undergrad and graduate student laboratory staff.

Biomechanics and Gait Lab — Coral Gables, FL
Master's Researcher — Aug 2014 to May 2017

- **Contributed to design, development and implementation** of data collection strategies that laid foundation strategies for multiple motion capture technologies.
- **Determined effectiveness of emerging technologies** at aiding rate of rehabilitation, providing benefit to both physicians and patients.

Education

Boston University — Boston, MA
PhD in Rehabilitation Sciences — May 2025

- Dissertation: "What Moves Us - Studies of Propulsion Measurement in Post-stroke Walking."

University of Miami — Coral Gables, FL
Master of Science in Biomedical Engineering — May 2017

- Thesis: "Validation of Microsoft Kinect for Use in Detecting Balance Impairment in ACL Repaired Patients."

University of Miami — Coral Gables, FL
Bachelor of Science in Biomedical Engineering — May 2014

- Senior Design Project: Design of a cell phone-based diagnostic and biofeedback device for balance training and prevention of falls.

Technical Skills

- **Python:** Numpy | pandas | PyTorch | TensorFlow | Scikit-Learn | matplotlib
- **MATLAB:** Statistics | Machine Learning | Signal Processing | Simulink
- **R Programming:** shiny | tidyr | ggplot | dplve
- **Julia:** DataFrames.jl | SciML | JuliaStats | Lux | Makie
- **SQL:** SQLite | MySQL | PostegreSQL
- **Excel / Google Sheets:** PivotTables | formulas | functions | VLOOKUP
- **Motion Analysis:** Vicon | Qualisys | Bertec | Xsens | Delsys | V3D