# A Lower-Body Exoskeleton Simulation and Control Framework based on Deep Reinforcement Learning

by

Lowell Rose

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Department of Mechanical and Industrial Engineering
University of Toronto

# Abstract

A Lower-Body Exoskeleton Simulation and Control Framework based on Deep Reinforcement Learning

Lowell Rose
Master of Applied Science
Department of Mechanical and Industrial Engineering
University of Toronto
2020

Stroke is a leading cause of disability, and post-stroke individuals often experience hemiparesis, which can result in compensatory movements that affect their overall gait pattern. Lower-body exoskeletons allow for repetitive, intense, and task-based practice of movement patterns that can promote gait recovery. Validating exoskeleton gait patterns in simulation is vital for ensuring patient safety and comfort; however, current simulation tools possess challenges in controlling exoskeleton hardware. Additionally, present model-based adaptive controllers rely on complex dynamics models for the user and exoskeleton that are difficult to accurately define. In this thesis, a centralized exoskeleton simulation and control framework is presented, where desired gait patterns can be validated and customized on-the-fly in physiotherapy sessions. A model-free, end-to-end deep reinforcement learning controller is also presented, to allow for exoskeleton control that is robust to desired gait patterns and a user's interaction with the device. Several experiments were conducted to validate these approaches.

## Acknowledgements

I would first like to thank my supervisor Professor Goldie Nejat for her guidance and support throughout my master's degree. I would also like to thank my committee members for their valuable feedback and time. Thank you to Michael Bazzocchi for his assistance and support in completing the projects discussed in this thesis, Kara Patterson, Julie Vaughan-Graham and Dina Brooks for their collaboration in defining the rehabilitation goals of this project, and Guillermo Asín Prieto for his advice on exoskeleton hardware related matters. I also would like to thank my fellow labmates as well as colleagues at the Rehabilitation Sciences Institute for their additional support, and for creating an enjoyable work environment and experience. Finally, I must express my sincere gratitude to my parents, sister, roommate, and friends for their constant support and encouragement throughout my program, without whose support this accomplishment would not have been possible.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Stroke is a leading cause of disability in Canada, with between 654,000 and 726,000 Canadians expected to experience a stroke by 2038 [1]. After a stroke has occurred, post-stroke individuals often exhibit reduced motor control, typically on one side of their body. They often develop compensatory and atypical motor behaviour that affects their gait pattern as a result [2,3], where it will differ from that of a healthy individual [4]. A post-stroke individual affected by hemiparesis, or weakness, on one side of their body, will typically experience changes to the spatio-temporal and kinematic parameters of their gait [4]. A full cycle of a gait pattern is composed of distinct segments that can be grouped into phases consisting of toe off, hip swing, heel strike, and flat foot/stance [5]. These gait phases can then be more generally grouped into stance and swing phases [6], where the stance phase will begin with heel contact and continue until toe off, and where the swing phase is comprised of the hip swing movement. Effects of hemiparesis in post-stroke individuals commonly manifest as a longer period of stance on the non-affected leg, [7] and also results in insufficient foot clearance during the swing phase in the affected leg [2], in addition to a longer duration of this phase [7–10]. The asymmetric gait and movement patterns that are found as a result [11], are often due to the post-stroke individual attempting to compensate for these characteristics. In general, it has been found that post-stroke individuals have reduced range of motion and flexion in the sagittal plane of their joints [3, 12]. Therefore, the improvement of function, motor performance, and gait recovery of these users is a main goal of post-stroke gait rehabilitation.

More specifically, to improve the gait performance of a post-stroke individual, improving the symmetry of step length, swing time, and foot clearance of their gait are common goals, as well as increasing hip and knee flexion during their swing phase [2, 11]. In particular it is recommended that gait rehabilitation for post-stroke individuals should involve repetitive, intense, and task-based movements [13]. Traditionally, a common method of therapy to accomplish this involves gait training on a treadmill [13], typically with body-weight support, where a patient is supported from above to relieve pressure on their limbs, and where a physiotherapist may also assist in providing correct motions to a patient [14, 15]. However, a therapist providing manual assistance during treadmill training can result in limited outcomes such as poor coordination and synchronization of movements in both legs [16], where therapist discomfort, and the number of therapists required for this task are also limiting factors [17, 18].

### 1.1.1 Exoskeletons for Post-Stroke Gait Rehabilitation

In recent years, lower body exoskeletons have been introduced as tools to replicate traditional physiotherapy and deliver the required task-specific, repetitive movements. They have been investigated as potential tools for gait rehabilitation of people affected by stroke and spinal cord injury [14, 17, 19–22], where they have been shown to be effective intervention tools for these patients [23]. The use of these exoskeletons can assist in increasing walking speed in post-stroke individuals, recovering gait symmetry, and improving range of motion, as well as improving mobility and function overall [14, 23, 24].

The effectiveness of these devices in overground gait training is thought to arise from their ability to provide greater user participation and practice of typical movement patterns found in overground walking [16], such as requiring the user be responsible for maintaining balance and navigation [14]. This can enhance movement recovery and motor learning, and also therefore reduce the atypical and compensatory motor behavior. More specifically, using an exoskeleton for gait rehabilitation provides the opportunity for greater intensity and repetition of specific gait-based tasks, while providing cognitive engagement [16], which are main elements of gait rehabilitation understood to increase positive neuroplastic change [25–27]. In effect this can optimize motor learning and recovery, especially in over-ground walking environments. In general, post-stroke gait rehabilitation with exoskeleton devices have been found to improve mobility and function [24, 25], especially when used in tandem with traditional physiotherapy [25].

A majority of exoskeleton devices intended for use in rehabilitation, either available commercially or for research, are limited to active actuation in the hip and knee joints, where a passive or spring activated ankle joint is commonly included [28]. Post-stroke rehabilitation studies and physiotherapist recommendations however have found that an actively actuated ankle joint is a key element of gait rehabilitation, as it addresses the propulsion and foot-drop issues that are common in post-stroke individuals [29, 30], and allows the patient to achieve a more natural gait pattern. The H2 exoskeleton, developed by Technaid S.L. [31], is one model of exoskeleton device that possesses an actuated ankle joint along with actuated hip and knee joints. The H2 can therefore deploy a fully actuated gait pattern during rehabilitation.

### 1.1.2 Simulation and Control of Exoskeletons

Typically in lower-body exoskeleton research, simulations of these devices are generated before they are implemented on the physical hardware, in order to develop and verify control schemes [32, 33]. Furthermore, validating appropriate exoskeleton gait patterns in simulation for use in a physiotherapy session is vital for ensuring patient safety and comfort. A variety of simulation platforms such as Adams [34], OpenSim [35], and Simulink [36] have been used for these simulation tasks, but often cannot directly be used to control the physical hardware, or have potential drawbacks such as limited real-time performance. In these cases, new software tools to generate or transfer a model to hardware are often necessary for implementing the simulated control schemes on an exoskeleton. If this is not done, the software is often duplicated on the hardware platform, where this can delay validation or testing of gait patterns required for rehabilitation. This can also hinder the on-the-fly customization of gait patterns that is desired for individualized gait rehabilitation.

In the development of control strategies for lower-body exoskeletons, common areas of research include increasing the effectiveness, accuracy, and adaptability of an exoskeleton between differing patients. Improving the trajectory tracking ability of these controllers has often been a main focus, where a specific gait pattern is imposed on a patient [37,38]. While this has been found to aid in post-stroke recovery and allow for some improvement in gait rehabilitation, it can result in less patient participation and therefore motor learning. This can in turn lead to insufficient improvement in gait recovery for the individuals using the exoskeleton, as well as lead to user discomfort [37, 39, 40]. When gait training that is personalized for users is provided, more effective gait recovery has often been found as a result [5, 41–44]. Furthermore, exoskeleton control schemes that provide assist-as-needed control, where the only the amount of torque or power required to assist a user in reaching a goal is provided, have been a popular area of research [14, 17, 37, 45, 46]. To do so, there have been a variety of adaptive model-based control techniques proposed, where a predefined dynamics model of both the exoskeleton and user is often required [47]. Defining this model accurately is a key challenge however, as due to the complex nature of the human-exoskeleton interaction, and the resulting dynamic uncertainties, nonlinearities, and perturbations, these defined dynamics models are often not able to fully and accurately describe the interaction between the exoskeleton and user [48–52]. Control actions taken using inaccurate dynamics models could then result in tracking and control errors, potentially leading to user discomfort or injury.

## 1.2 Problem Definition

Limiting factors in the gait rehabilitation of post-stroke individuals include time spent in physiotherapy, the frequency and intensity of task-based gait training, as well as physiotherapist effort required to provide this. Physiotherapists typically have to continually work with a patient to adapt and adjust desired gait patterns, and will often have to provide these movements manually in gait training, depending on the level of recovery of the patient. Therefore lower-body exoskeleton devices which are able to assist in gait rehabilitation by providing consistent and repetitive gait training are a valuable asset to physiotherapists and can optimize time spent in rehabilitation. To further optimize time in clinical sessions it is vital to be able to simulate and verify gait patterns prior to implementation on a physical device. While many simulation tools currently exist for this purpose, they possess limitations in their ability for hardware implementation and control. Therefore a centralized framework for simulating and controlling exoskeleton gait patterns is desired. Furthermore, in the control of these exoskeleton devices, subject-specific and adaptive control is important for obtaining the best results through gait rehabilitation. Designing these subject-specific and adaptive control schemes are difficult however due to the complex nature of the human-exoskeleton interaction, and the dynamics models that are typically required are difficult to accurately define. Therefore control schemes which use deep learning methods to autonomously adapt to patients in a model-free nature is a promising area of research.

The objectives of this thesis are to first present the development of a simulation and control platform for the H2 exoskeleton, where exoskeleton gait patterns can be simulated and controlled through a centralized software structure, and can then also be adapted and verified on-the-fly in physiotherapy sessions. Then the development of a deep reinforcement learning (DRL) based method for model-free, subject-specific, adaptive exoskeleton control is discussed. This DRL-based approach can enact control that is robust to a post-stroke individual's baseline gait pattern and the gait pattern goals that are desired throughout physiotherapy, without the use of a predefined dynamics model. Experiments are then conducted to validate these exoskeleton control approaches.

## 1.3 Proposed Thesis Methodology

The proposed software architecture for the simulation and control of a lower-body exoskeleton first includes the use of the Robot Operating System (ROS) and the Gazebo simulation environment to provide a centralized platform for storing and updating desired gait patterns on-the-fly during physiotherapy sessions, and where the execution of the controller is not reliant on the performance of the simulation. This approach allows for gait patterns to be first verified in simulation before deployment on the exoskeleton hardware, and can therefore optimize physiotherapist time in clinical sessions, while providing a platform to safely evaluate these gait patterns.

To allow for subject-specific and adaptive control on an exoskeleton, a DRL method for exoskeleton control was developed using the OpenSim-RL platform, where an exoskeleton can accurately and robustly be controlled for a wide range of desired gait patterns while incorporating a user's existing baseline gait pattern and interaction with the exoskeleton. This is accomplished through observing the state information of both the user and exoskeleton and training the DRL agent to enact the appropriate control actions.

The proposed contributions of this thesis are discussed in the following sections:

### 1.3.1 Literature Review

Chapter 2 will provide a literature review on the existing simulation and control platforms for lower-limb exoskeletons, adaptive control techniques, and reinforcement learning and deep reinforcement learning based techniques for exoskeleton control and locomotion. The limitations of the existing techniques are discussed, and the potential solutions are identified.

### 1.3.2 A Simulation and Control Framework for Exoskeletons

Chapter 3 will present the development of a centralized software framework for both the simulation and control of an exoskeleton, based on the ROS and Gazebo platforms. Each of the modules necessary to do so are discussed in detail, and experiments are presented to validate the approach.

### 1.3.3 End-to-end Deep Reinforcement Learning For Exoskeleton Control

In Chapter 4, an end-to-end, model-free, DRL-based exoskeleton control scheme is presented. This control method incorporates a user's existing baseline gait pattern, as well as the desired gait pattern into the DRL training. The DRL environment includes a simulated exoskeleton paired to a musculoskeletal model in OpenSim, and receives the user's joint angles, velocities, and accelerations as observations, and produces the exoskeleton joint torque actions necessary to follow a desired gait pattern, while being robust to a user's movement and perturbations. Initial experiments of this method are discussed in this chapter.

### 1.3.4 Generalized DRL Control

Chapter 5 will discuss the development of a generalized DRL controller to allow for robust and generalized control for various post-stroke individuals and desired gait patterns. This is accomplished through updates to the DRL scheme presented in Chapter 4, and includes training with a clinical dataset of post-stroke individuals' baseline gait patterns, as well as a dataset of desired gait patterns gathered from healthy users. Additional experiments are presented that validate this approach both on the training dataset of desired gait patterns, and a set of unseen desired gait patterns.

### 1.3.5 Conclusions

Chapter 6 will discuss the contributions of this thesis on the development of the centralized exoskeleton simulation and control framework, and the model-free, DRL-based exoskeleton control approach. Future recommendations for both of these approaches will also be discussed.

# Chapter 2

# Literature Review

This chapter will focus on the existing literature in the simulation and control platforms used for exoskeletons for rehabilitation, and the existing adaptive and learning-based control methods that have been developed. The simulation software platforms that are used for exoskeleton modeling, those used for control, and the potential drawbacks of some of these platforms will be first discussed, as well as the availability of developed solutions. Then a more extensive discussion into the existing control methods used in prior exoskeleton research is presented. These are grouped into those based on traditional model-based adaptive control methods, and those that use learning based methods, including reinforcement learning (RL) and deep reinforcement learning (DRL).

## 2.1 Simulation Platforms for Exoskeleton Design and Control

Lower body exoskeleton modeling and simulation has generally focused on determining the dynamics and forces present when using an exoskeleton [34, 35, 53–55], or for evaluating a specific controller design [32, 36, 56, 57]. Simulation software packages that have been used include the AnyBody Modeling System (AMS) [53], SimMechanics [32, 55], Adams [34], Simulink [36, 56], and OpenSim [35, 57].

For example, in [53], the Anybody Modeling System (AMS) software was used to simulate a human musculoskeletal model in conjunction with the ARKE exoskeleton, in order to develop a control method based on ground reaction forces. In [32, 55], simulated dynamics models of exoskeletons and their controllers were generated in SimMechanics, in order to test a neural network based controller. In [34], an exoskeleton was modeled in the multibody dynamics simulation software, Adams. This platform was used to verify the exoskeleton's proposed design through comparing simulated and desired joint angles, as well as interaction forces between the exoskeleton and a musculoskeletal human model.

In [35], a model consisting of the kinematic and dynamic properties of an existing exoskeleton was generated in OpenSim to verify the kinematic design and actuator performance. A musculoskeletal human model was added to investigate the interaction between the user and the device, and appropriate actuator torque values were found. A modification to this exoskeleton design was proposed, and a new dynamic model was tested in simulation. In [57], the robot operating system (ROS) was used with the H2 exoskeleton to obtain the online estimation of a patient's muscle forces through electromyography (EMG). A musculoskeletal model was then simulated in OpenSim using this data.

The aforementioned methods focused on the modeling and control of these exoskeletons solely in simulation, or used the simulation platform to analyze or view experimentally collected data. Alternatively in [36], a model of an exoskeleton was simulated using Simulink and converted with the QUARC software into a representation that could be transferred onto a physical exoskeleton. In [56], a similar approach was conceptually presented. However, the use of a Simulink based control method has potential drawbacks such as limited real-time performance when using QUARC [58], and the potential for dropped commands when used in conjunction with ROS [59]. When used on an exoskeleton with a patient, these limitations could alter the effectiveness of the device, and potentially lead to user discomfort.

Additionally, in many of these simulation techniques, replication of software in separate simulation and hardware control architectures would be required [34, 35, 53]. Furthermore, in the control approaches using simulation platforms such as Simulink [36, 56], the output of the simulation is required for enacting control on the hardware. The simulation's speed often depends on the model complexity and computational power [59], and therefore if the deployed model is too slow to meet execution frequency, control commands (i.e., joint angle positions) can be missed, resulting in incorrect hardware execution [59]. Such processing delays and synchronization issues could further be problematic in a clinical setting, where incorrect exoskeleton movements can result in patient injury [21, 60].

The combination of ROS and Gazebo for control and simulation is one that has been used for other hardware platforms such as mobile robots and manipulator arms [61, 62]. For manipulators, ROS and Gazebo have been used to perform trajectory planning, manipulation and grasping control, and for mobile robots to simulate autonomous navigation and 3D-mapping, as well as enable experiments in a physical environment [61]. This combination allows for the ease of prototyping, validation and testing by consolidating the simulation and hardware control software, and is a useful tool to also apply to the development of exoskeletons. The use of exoskeleton devices presents challenges related to the interaction of a patient and the device, where safety and comfort are primary concerns, and simulations are vital to validating an appropriate gait pattern on-the-fly in a physiotherapy session.

An exoskeleton control framework using these tools would address the aforementioned drawbacks of other simulation platforms, and its development will be discussed in Chapter 3. In order to further increase the accuracy and realism of the exoskeleton simulation however, it is important to include a model of the user, as was done in some of the above techniques. To do so, OpenSim, a musculoskeletal modelling platform, is used to develop the proposed DRL-based control approach, where a human musculoskeletal model is incorporated into the the exoskeleton simulation environment. This is further discussed in Chapters 4 and 5.

## 2.2 Exoskeleton Control Methods

Subject-specific control of exoskeletons has been a growing area of research, as it has been found that subject-specific gait training can often lead to greater recovery in a post-stroke individual's gait pattern [5, 42–44]. Previous work that has focused on the development of exoskeleton control with patient-specific adaptation can be sorted into categories such as traditional model-based adaptive control methods [63–69], and reinforcement learning methods [70–74]. Deep reinforcement learning (DRL) techniques have also been proposed for learning bipedal locomotion [50, 75–80], as well as learning upper and lower limb exoskeleton-based control tasks [81–83], but have not yet been applied to end-to-end exoskeleton control for gait training.

### 2.2.1 Model-Based Adaptive Control

Typical adaptive control techniques use a dynamics models to represent the equations of motion of the exoskeleton and user, and determine the actuator torques necessary to control an exoskeleton in the presence of human interaction [39, 45, 47, 48, 63–65, 84]. To-date adaptive controllers allow exoskeletons to adjust and synchronize to users online [39, 45, 47], and try to account for some of the dynamic uncertainties and perturbations resulting from the human-robot interaction and variations in the behavior of different users [48, 63–65, 84]. Existing adaptive control techniques attempt to personalize to various users through adaptation of subject-specific parameters such as step length or walking speed [63, 67], or synchronize with a user's movements through online adaptation and feedback [64, 65, 68, 69].

In [63], a dual unscented Kalman filter was used to predict and generate exoskeleton trajectories based on the spatiotemporal features of a user's gait. This was accomplished with a partial coupled dynamics model of a human and exoskeleton, where the exoskeleton and user's limb were modeled as one body. An impedance supervisory controller was then used to follow a trajectory and synchronize with the user during their locomotion.

In [64], a compliance controller was used to provide motion control to an exoskeleton based on tracking a user's joint angles, foot pressure, and trunk inclination. A parameterized trajectory was created and deployed on the exoskeleton. The controller used the generated trajectory as equilibrium points, while allowing the exoskeleton to deviate slightly from these points when perturbed by the user in order to achieve compliance. In [67], an exoskeleton was controlled using end point model predictive control (MPC) to create joint trajectories online. Step-length, swing duration and walking speed were used as inputs to create end point references for the MPC controller. This allowed for user-specific gait assistance within the swing portion of a gait pattern.

In [65], an assist-as-needed method was presented, where an exoskeleton only enacts force on a user when they cannot reach a desired goal on their own. An impedance-based force-field controller was designed to assist users by tracking the interaction forces between the user and the exoskeleton. The level of impedance was then adjusted based on whether the user was within the defined force-field. In [66], an impedance controller was also implemented to enable assist-as-needed control, by adjusting the end-point stiffness of the exoskeleton depending on the user's kinematic deviations in their gait phases.

In [69], a bipedal robot control design was adapted for an exoskeleton for spinal cord injury patients in order to allow them to walk without additional balance support. This method used a feedback controller, based on virtual constraints, to output a velocity regulating parameter comprised of the forward hip velocity of the exoskeleton, and posture regulating terms to synchronize the other joints. This method took advantage of offline trajectory optimization techniques to design gaits that could be tracked efficiently online.

In [68], a linear quadratic regulator (LQR) was used to improve trajectory tracking control on a lower limb exoskeleton, and to account for disturbances resulting from human-exoskeleton interactions. The LQR control inputs consisted of a feedforward reference torque, and feedback proportional, integral, and derivative actions. The parameters of the exoskeleton dynamics model were then derived experimentally through least squares estimation.

The aforementioned control methods require defined dynamics models of the exoskeleton and user. However, accurate models can be difficult to obtain due to the human-exoskeleton interaction and the resulting non-linear characteristics of this interaction [32,85]. Therefore, typically, to facilitate the use of these adaptive control schemes, these models have often been simplified by considering either only the interaction forces between the exoskeleton and user [65], or representing the human and exoskeleton as one body [63, 64, 67, 68].

### 2.2.2 Reinforcement Learning In Exoskeleton Control

Reinforcement learning (RL) for exoskeleton control is an emerging area of research. Due to its nature of exploring an action space and finding optimal actions, it has been used in a variety of tasks where an optimized or subject-specific controller is desired. Thus far it has generally been used in exoskeleton control to optimize parameters of adaptive controllers [70, 71], to account for user-exoskeleton interactions [73, 74], and also as a means to provide assist-as-needed control [72].

In [70], dynamic movement primitives (DMPs) were used with an exoskeleton to model human movement trajectories and adapt to user motions. A coupled cooperative primitive (CCP) was defined, which incorporated an interaction term into a conventional DMP, using an impedance-based spring-damper model to describe the interaction. The CCPs were first learned through imitation learning using motion trajectories of users, then RL was used to update the stiffness, damping and scaling parameters of the CCPs. Using RL to learn these parameters resulted in the reduction of human and exoskeleton interactive forces and dynamic uncertainties.

In [71], RL was used for learning personalized parameters for an admittance controller, in order to reduce interaction forces between a user and a lower limb exoskeleton, while following a reference trajectory. The optimal parameters of stiffness and damping for the admittance controller were learned and tuned based on the performance of the user by using joint angle error and user-exoskeleton contact forces as observations. In [86], a similar method was employed for learning optimal impedance control parameters for an upper limb exoskeleton.

In [74], RL was used to design a model-reference compliance controller and to track a reference trajectory for a lower limb exoskeleton. Using Q-learning, the exoskeleton tracked a reference trajectory by observing discretized exoskeleton joint angles, velocities, and accelerations, and outputted joint torque values to a joint-based compliance controller. The compliance controller was added to incorporate safety features into the exoskeleton to allow for perturbations from the user, by modeling the exoskeleton joints as a second order mass-spring-damper system. The mass-spring-damper model was then used to account for interaction torque from the user and adjust joint torques accordingly.

In [73], assistive strategies for an upper limb exoskeleton were learned from interactions between an exoskeleton and user. Model-based RL was used to learn a dynamics model and control policies from limited data. The user moved a simulated arm with their muscle-activations recorded with electromyography (EMG), and a model-based policy search method found the optimal torque to assist the user in completing a reaching task and reduce the observed EMG signals.

In [72], an actor-critic based approach was used to model an impedance controller for assist-as-needed control of an ankle exoskeleton. The RL agent adjusted the level of assistance by altering the stiffness parameter of the force-field provided by the impedance controller, based on the user's performance. Performance was gauged by gathering observations of the tracking errors from a reference trajectory. This allowed for the controller to assist users in tracking a sinusoidal reference trajectory shown on a screen.

The aforementioned RL approaches were used for tuning or finding optimal parameters of a controller [70–73] or for tracking a reference trajectory through modeled joints [74]. However, they also required a predefined dynamics model or had to learn the dynamics model by learning the transition probabilities of state-action pairs. RL is also generally constrained to discrete low-dimension observation and action spaces, and it is not able to handle high dimensional or continuous systems [87]. However, exoskeleton control requires continuous high-dimensional observation and continuous action spaces to represent the joint angles, velocities, accelerations and actuator torques, and discretization these spaces is difficult as there are too many state-action pairs to store [88], also known as the curse of dimensionality.

### 2.2.3 Deep Reinforcement Learning for Exoskeletons and Locomotion

Deep reinforcement learning (DRL), has the ability to learn in these high-dimensional continuous observation and continuous action spaces by mapping the states to actions through the use of deep neural networks as function approximators [88]. Therefore, it can overcome the limitations of standard reinforcement learning methods, while not requiring a predefined or learned dynamics model. The ability to learn in these spaces allows for more information to be provided to the controller, which can therefore output more accurate control torques [82]. To-date, model-free DRL has been used to learn tasks such as human locomotion [75–77], control of bipedal robots [50, 78–80], and control of upper and lower limb exoskeleton joints [81–83].

In [75–77], DRL was used to learn locomotion skills for human musculoskeletal models. This was done with observations of joint angles, velocities, and accelerations, and joint torques or muscle activations as actions. DRL has also been successful in learning locomotion and motor skills for biped or legged robots, with stability, speed, and distance traversed as learning goals [50, 78–80].

With respect to exoskeletons, in [82], DRL was used to learn an optimal controller for enabling functional electrical stimulation (FES) of an upper limb while wearing a passive elbow exoskeleton. This was accomplished using a Proximal Policy Optimization (PPO) method. The learning agent took angular position, velocity and acceleration of the elbow, and the deviation from the expected goal as observations. This allowed for learning continuous actions of electrical stimulations to the arm muscles in order to enact elbow extension movements and for the user to reach specified elbow angles.

In [81], DRL was used to learn a fall predictor and recovery policy for an exoskeleton that was designed to assist in fall prevention of elderly users, by providing small torque adjustments to a user's locomotion when a potential fall was detected. PPO was used to learn a policy for human locomotion, and the simulated locomotion data was then used to train the fall recovery policy. This also was accomplished using PPO, with hip joint angular position and velocity as observations, and hip joint torques as actions. A support vector machine (SVM) classifier was trained to predict states that could lead to potential falling actions, and a hip exoskeleton was then added to validate the fall recovery policy and provided actuator torques to assist with maintaining stability.

In [83], a rehabilitation training game was presented wherein a character in the game was directed by a user wearing an EMG-controlled knee exoskeleton. The user's muscle activity was recorded from a thigh worn EMG sensor and translated into exoskeleton movements which controlled the game character. To enable the DRL-based assistance, a deep-Q network algorithm (DQN) was used to gather image-based observations from the game interpreted through a convolutional neural network (CNN), and produced outputs of discrete movement actions to control the character. To assist the user in playing the game, the DRL agent was able to augment the user's EMG-based control by converting the game movements into joint angles on the exoskeleton.

In general, the aforementioned DRL techniques have shown how DRL can be used to learn locomotion skills and help to control actions such as elbow movements or maintaining balance. However, they have not focused on the specific task of learning gait patterns for exoskeletons to be used in over-ground gait rehabilitation. End-to-end model-free DRL control methods, where actions are learned directly from raw sensor observations [89], can be specifically useful for such continuous and high-dimensional control tasks, as they do not require a predefined dynamics model and can learn from only the exoskeleton directly interacting with the environment. A novel DRL-based exoskeleton control framework will thus be presented in Chapters 4 and 5.

## 2.3 Chapter Summary

A number of existing simulation and control platforms for use with exoskeletons were first discussed. This included those used solely for simulation, and those that could be extended for hardware control. The limitations of these approaches, and potential centralized simulation and control solutions for this problem were presented. Then, existing exoskeleton control schemes for subject-specific control were discussed. These included adaptive model-based control techniques where a predefined dynamics model is required, and learning based control techniques based on reinforcement learning and deep reinforcement learning. The benefits of model-free DRL specifically for exoskeleton control were then also discussed.

# Chapter 3

# A Simulation and Control Framework for Exoskeletons

In this chapter, the development of a centralized control architecture for the simulation and control of an exoskeleton is presented. The proposed approach allows for a desired gait pattern to be controlled and adjusted on-the-fly both in simulation and on a real exoskeleton. This can ultimately allow for customization and validation of gait patterns directly during gait rehabilitation sessions, thereby optimizing the time spent in rehabilitation. This is primarily achieved through the use of ROS and the Gazebo simulation platform, the details of which are discussed in this chapter.

The main advantages of this method are first that it provides a centralized software structure to store and control gait patterns for both the simulated model and the exoskeleton hardware, where only a single software structure needs to be adapted for gait pattern customization (i.e., no replication of software is required), and secondly that the exoskeleton hardware control is not reliant on the execution and output of the simulation, therefore avoiding computational interruptions and performance issues, where incorrect movements could result in patient discomfort or injury.

## 3.1 Control Architecture

The overall control architecture that was developed is presented in Figure 3.1. This control architecture is based on the Robot Operating System (ROS), a popular middleware software for robotics that provides a framework for the communication of messages between control processes, low-level hardware, and high-level controllers. The input to the control architecture is the desired gait pattern to be controlled by the exoskeleton. In this chapter, this is provided by OpenSim's 2354 gait model [90], Figure 3.2. The 2354 Gait Model is musculoskeletal model of a human that is approximately 1.8 m tall with a mass of 75 kg. The experimental data provided with this model was obtained from [91].

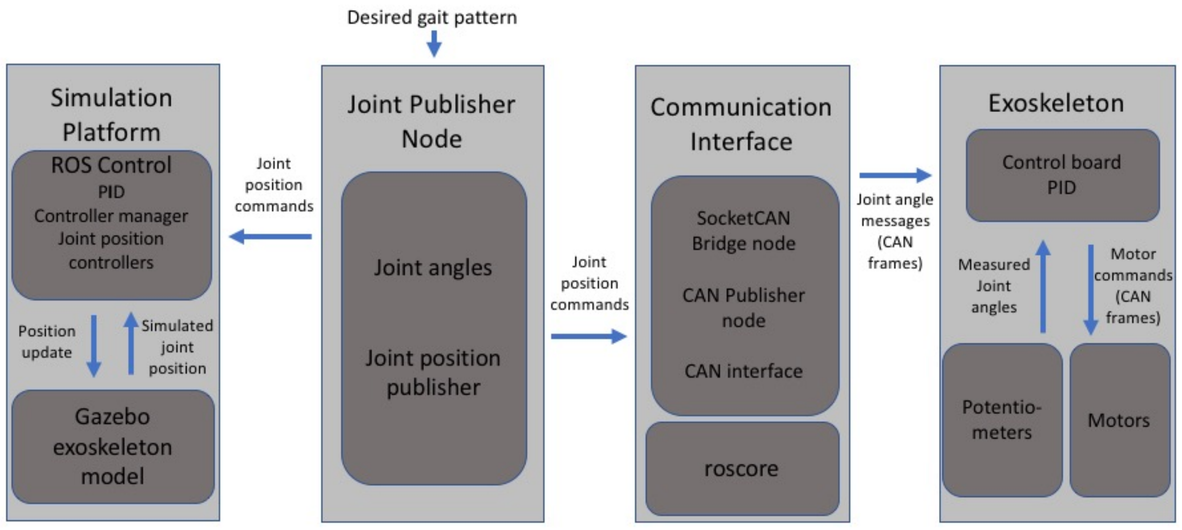

Figure 3.1: The overall proposed architecture for the control of the simulation and exoskeleton hardware devices.

The control and communication architecture seen in Figure 3.1 consists of the following main modules: 1) the Joint Publisher Node, which is responsible for sending joint position commands to the entire architecture, 2) the Simulation Platform, which models and controls the exoskeleton in a simulated environment, 3) the Communication Interface, which consists of the development board used to communicate joint angle messages to the exoskeleton, and 4) the Exoskeleton module, which executes control commands on the physical device. These modules are discussed in further detail below. The H2

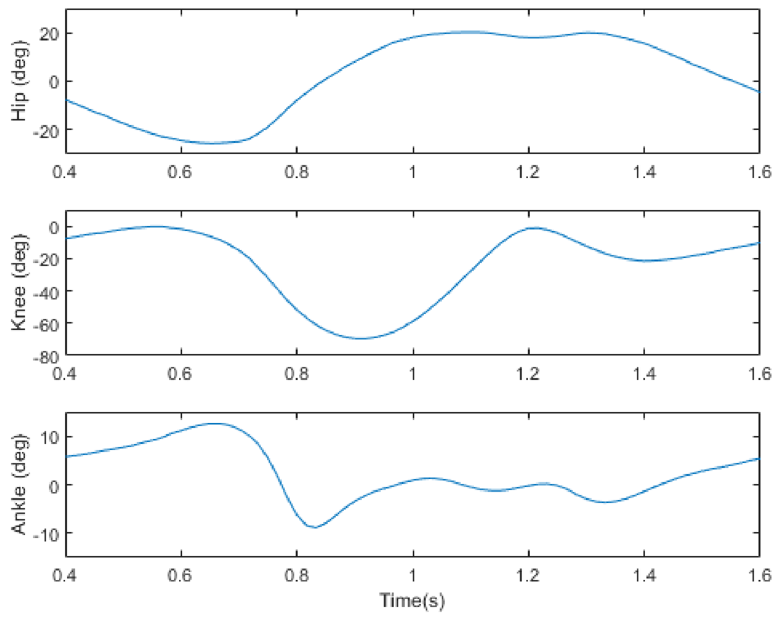

Figure 3.2: Desired gait pattern based on OpenSim's 2354 model.

exoskeleton from Technaid S.L. [31], is then used for implementation and testing of the control architecture.

### 3.1.1 Joint Publisher Node

The Joint Publisher Node is responsible for providing the control commands to the entire architecture. This module contains the aforementioned gait pattern from OpenSim's 2354 gait model [90], which is represented as a series of joint angles in the sagittal plane. This is the same format used to store gait patterns as on the H2's existing control architecture, and was designed to be similar. The Joint Publisher Node reads the provided joint angles and also establishes each of the joint position topics. In ROS, *topics* are the communication channels that are established to send messages throughout the network and to the nodes. The joint publisher node then publishes joint position commands for each joint to the Simulation Platform and the Communication Interface modules, over the joint position topics. The speed of publishing is also set in this node, where messages are sent over topics to the network at a speed of 20 Hz.

### 3.1.2 Simulation Platform

To simulate the exoskeleton, a combination of a ROS Control node [92] and Gazebo [93] is used. ROS Control is an open source package developed for ROS, which allows for common controllers to be initialized both in Gazebo and on a hardware platform. The ROS Control framework also has the capability to both establish and store high level control schemes, as well as the ability to communicate with lower level hardware interfaces such as motor control boards, drivers, and sensors. In this application, ROS Control is used to establish a controller manager to subscribe to each joint position topic, a joint position interface to store the simulated joint positions, and also contains a PID-based position controller. The PID controller is based on the H2 exoskeleton's built-in controller [94] with similar PID gains, which is used to obtain and hold joint positions through torque output from the motors. The joint position commands published from the Joint Publisher Node are sent to the established position controller of each joint in Gazebo, to simulate the motion of the exoskeleton. The simulated joint position data can then be saved with the ROS command, *rosbag*, for analysis and comparison. This tool provides a timestamped recording of messages that were sent and received over selected topics.

As an initial step in simulating the form and dynamics of the exoskeleton, the March IV exoskeleton model [95] in the Unified Robot Description Format (URDF) was extended to represent the physical configuration, dimensions and specifications of the H2, Table 3.1. The URDF framework is a widely used robotics description format based in XML. It is common across varying simulation platforms, and allows for various physical and dynamic characteristics of robot models to be set, such as size, mass, and mesh characteristics of bodies, joints between these bodies, and their moment of inertias. The 3D simulated model of the exoskeleton is presented in Figure 3.3.

| **Link** | **Thigh** | **Shank** | **Foot** |
|---|---|---|---|
| Length [m] | 0.40 | 0.40 | 0.23 |
| Mass [kg] | 1.00 | 1.00 | 0.5 |
| $I_{xx}$ [kg m$^2$]* | 0.034 | 0.034 | 0.013 |
| $I_{yy}$ [kg m$^2$] | 0.034 | 0.034 | 0.013 |
| $I_{zz}$ [kg m$^2$] | 0.042 | 0.042 | 0.021 |
| **Joint Limits** | **Hip** | **Knee** | **Ankle** |
| Maximum [°] | 105 | 105 | 15 |
| Minimum [°] | -30 | -5 | -15 |

*The $I_{xx}$, $I_{yy}$ , $I_{zz}$, are the moment of inertia tensor's diagonal values.

Table 3.1: Specifications of the simulated exoskeleton.

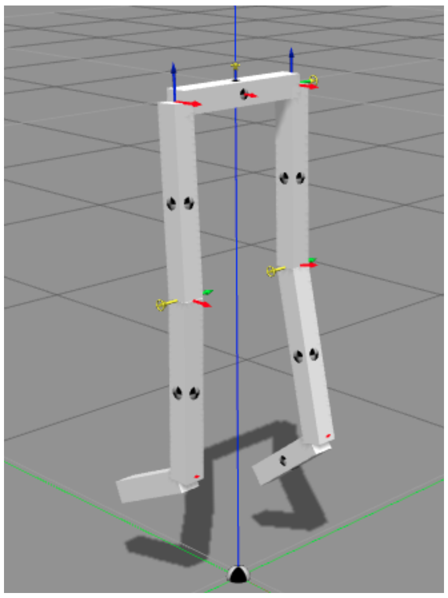

Figure 3.3: Simulated 3D model of the exoskeleton in Gazebo.

### 3.1.3 Communication Interface

To enable control on the exoskeleton, a CAN (Controller Area Network) Bus interface is necessary to send the joint position commands from the Joint Publisher Node to the on-board microcontroller, in the form of joint angle messages. The CAN bus is a message-based protocol originally developed as a communications bus for vehicles, that allows messages to be sent and received simultaneously to and from all nodes in a system [96]. A CAN node in the network, in this case the exoskeleton's motor drivers, will either accept or pass along a message based on its included identifier. This allows for consistent, high speed, and prioritized transmission of packets [96]. In a standard CAN bus architecture these messages take the form of CAN frames, where information including the ID of the intended recipient, the priority of the message, and up to 8 bytes of data are included [96].

The control board on the H2 has the built-in capability to receive CAN frames from an external source and send them to the motors. The CAN protocol and message types that can be sent and received were developed by Technaid S.L. and are outlined in [94], which forms the basis for the developed communication interface. To enable CAN communication, a BeagleBone Black (BBB) wireless development board [97] was used. The BBB has 512MB DDR3 RAM, 4GB 8-bit eMMC on-board flash storage with expandable micro SD support, 802.11b/g/n and Bluetooth 4.1 plus BLE. The use of this board's wireless capabilities with ROS allows for control processes to either be split between devices, or for all processes to be run on-board, which allows for wireless, stand-alone operation of the exoskeleton. The BBB also hosts the roscore, which executes the master program used for communication between the ROS nodes in Figure 3.1.

To create the CAN interface necessary for sending messages to the exoskeleton control board, a package called SocketCAN [98] is used. SocketCAN is an open source collection of drivers for Linux, used for establishing a CAN socket. This software allows a CAN port to be enabled by creating a socket similar to the TCP/IP protocol, and binds the socket to an interface in the software [98]. To send CAN frames over the network, a CAN transceiver is required to interface between the socket and the exoskeleton CAN input [96]. An add-on board for the BBB from Waveshare containing a CAN transceiver [99] was used for this purpose. To send the published joint angles to the CAN network, the CAN Publisher node is established. This node receives the joint position commands from the Joint Publisher Node and translates them into a CAN frame message format. On the BBB, a ROS node, SocketCAN Bridge [100], is used to establish a socket and ROS message format to represent and forward the CAN frames. The CAN Publisher node then sends joint angle messages in the form of CAN frames to the exoskeleton control board.

### 3.1.4 H2 Exoskeleton

The H2 exoskeleton, Figure 3.4, is a lower-limb 6 degree of freedom (DOF) device, for use in over-ground gait training with post-stroke individuals [16]. This device has active hip, knee, and ankle actuation controlled by brushless DC motors with attached harmonic drive gear boxes. The device also has sensors for joint position (potentiometers), joint interaction torque, and foot/ground contact (heel and toe) [94].

The associated kinematics model for the exoskeleton is also presented in Figure 3.4. Each leg is composed of 3 revolute joints, linked by equally sized linkages. In its current configuration these have a length $l_1, l_2$ of 0.4 m and represent the thigh and shank linkages. The foot linkage has a length $l_3$ equal to 0.23 m. The three joint angles, i.e., $\theta_1, \theta_2, \theta_3$, are the active hip, knee, and ankle joints, respectively; where $\theta_1$ is the angle from $y_1$ to $z_1$, $\theta_2$ is the angle from $y_2$ to $z_2$, and $\theta_3$ is the angle from $y_3$ to $z_3$. The zero-degree state for these motors is aligned with the z-plane for the thigh and shank, and aligned with the y-plane for the foot.

The on-board microcontroller on the H2 is based on an STMicroelectronics ARM architecture, as outlined in [16]. This control board receives the CAN frames and sends them to the motor control boards as motor commands for execution, at a speed of 1 Mbps, and communication frequency of 1 kHz [16]. The board also contains a PID controller to regulate the position control on the exoskeleton. The H2 includes on-board potentiometers for each motor to measure joint angular position, as well as hall effect sensors in each motor. As the measured joint angle data is being sent back over the CAN network, it is read by the CAN transceiver on the BBB and is stored with SocketCAN's *candump* function.

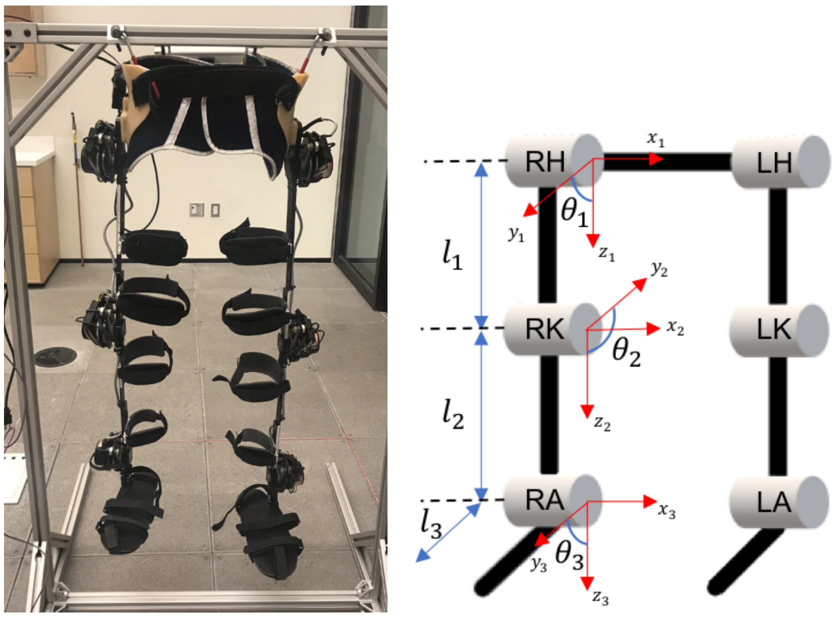

Figure 3.4: The H2 exoskeleton and associated kinematics model.

## 3.2 Experiments

To validate the effectiveness of the control architecture, preliminary experiments were conducted to obtain and compare joint angles for both the simulated and physical exoskeleton. A participant wearing the exoskeleton, as seen in Figure 3.5, repeatedly walked along a straight line for at least 5 gait cycles. The gait cycles were recorded both on the exoskeleton and from the Gazebo simulation to evaluate the real-time performance of the exoskeleton within the control architecture. Then, a new gait pattern provided by a physiotherapist was implemented on-the-fly during the experiment, and gait cycles were again recorded.

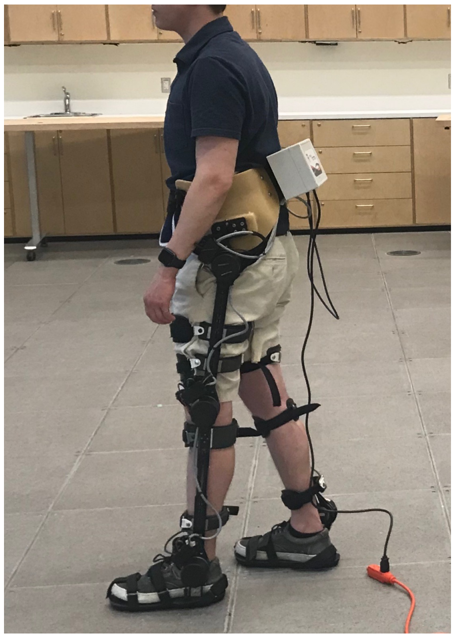

Figure 3.5: The experimental setup for the exoskeleton trial with a user wearing the device.

The joint position data, with timestamps, from the potentiometers on the exoskeleton captured using the *candump* command, was then compared to the input joint position commands and the simulated joint angles, which are obtained from Gazebo using the *rosbag* function.

### 3.2.1 Results

The results from the exoskeleton test with a participant wearing the exoskeleton for both the left and right legs are presented in Figures 3.6 and 3.7. When compared to the input commands, both the simulated joint angles and exoskeleton joint angles followed the desired gait pattern closely, where this is most evident with the hip and knee joints.

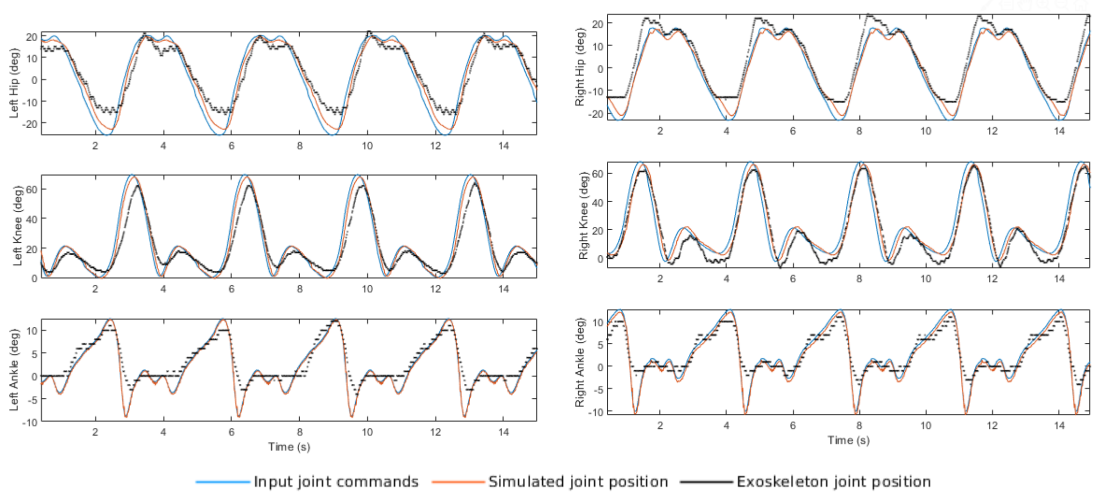

Figure 3.6: Results for the experiments with the exoskeleton worn by a user with the gait pattern derived from OpenSim.

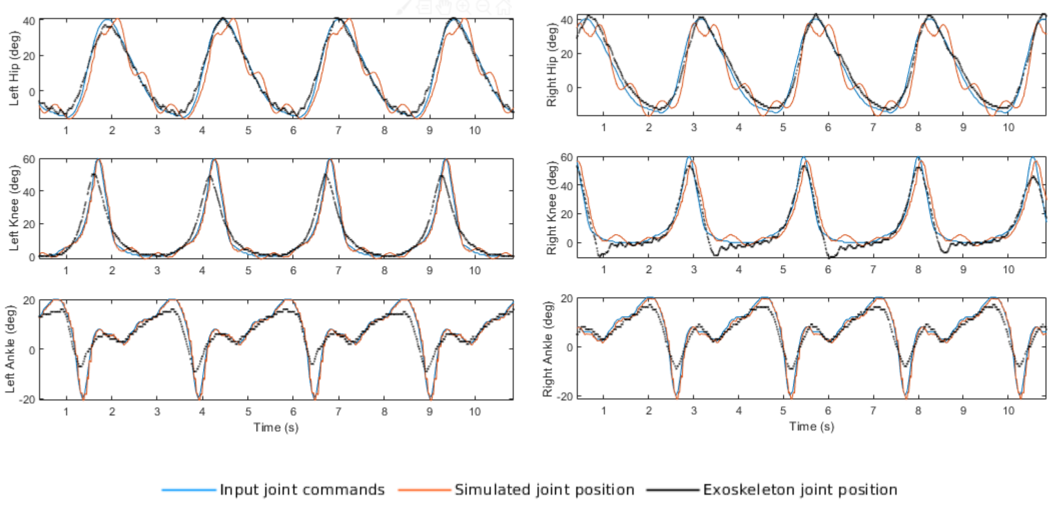

Figure 3.7: Results for the experiments with the exoskeleton worn by a user with the gait pattern recommended by a physiotherapist.

However, the exoskeleton did not reach the minimum joint angles for both the ankle and hip for the gait pattern in Figure 3.6 and for the ankle in the gait pattern in Figure 3.7. This was due in part to the physical joint limitations that are inherent in the design of the exoskeleton hardware, as well as the human-exoskeleton interaction. Some of the discrepancies observed (i.e., Figure 3.7 right knee) are also due to noisy potentiometer readings and slight differences in the simulation and hardware execution speeds. Errors due to the presence of the human-exoskeleton interaction were also observed, such as the left and right hip in Figure 3.6, where the recorded exoskeleton joint angles were found to overshoot the desired gait pattern in hip extension, and where the minimum desired joint angle in hip flexion was also not reached. The mean absolute error in the Gazebo simulation of the OpenSim gait pattern in Figure 3.6 was found to range between approximately 0.7° in the ankle to 3.1° in the knee. Furthermore, without additional controller tuning between desired gait patterns, the tracking accuracy of the simulation was found to be inconsistent when the gait pattern was changed, as evident in the simulated hip joint angles in Figure 3.7.

Additionally, due to this simulation not containing a human model, the performance of the exoskeleton in response to the human-exoskeleton interaction was not accounted for in this specific simulation. While this could be addressed by more substantial PID tuning, or approximated through adjusting the characteristics of the Gazebo model, the model in this application was primarily used to visualize and demonstrate a chosen gait pattern. For a higher fidelity simulation, a platform such as OpenSim, where the user is included in the simulation as a musculoskeletal model must be used. Furthermore, the controller must be able to accurately and robustly react to and compensate for the movements of the user wearing the exoskeleton, in order to address the observed effects of the human-exoskeleton interaction, as well as changes to the desired gait pattern, both in the simulation and on the physical device. The development of these simulation and control approaches are discussed in the following chapters.

## 3.3 Chapter Summary

In this chapter, a centralized framework for the simulation and control of a lower-body exoskeleton was discussed. This architecture was based on the open source Robot Operating System (ROS), and the Gazebo simulator. The input data to the architecture, represented as desired joint angle patterns obtained from OpenSim was first presented. Each of the individual modules comprising the full architecture were then described in detail, including the CAN interface for communication, and the hardware interface for integration on a lower body exoskeleton device. The proposed architecture was then validated through experiments on the H2 exoskeleton. This framework can then form the basis of the control and communication architecture necessary to conduct further clinical studies with the exoskeleton.

# Chapter 4

# End-to-end Deep Reinforcement Learning For Exoskeleton Control

In exoskeleton control, a previously defined dynamics model is commonly required to determine the control torque necessary to follow a desired gait pattern. However, it is often difficult to accurately define this dynamics model for both the user and exoskeleton to use with existing model-based control methods, due to the complex interaction between the user and exoskeleton, non-linear characteristics of this interaction, and additional dynamic uncertainties and perturbations. The limitations of the existing adaptive and learning based controllers discussed in Chapter 2 can therefore be addressed by model-free, deep learning-based control techniques. Deep reinforcement learning (DRL) more specifically allows for control in high dimensional observation and actions spaces, and possesses the ability to learn in these spaces by allowing for more information to be provided to the controller, where more accurate control torque actions can be provided as a result [82].

In this chapter, the development of a novel end-to-end, model-free DRL exoskeleton control method for user personalization of gait training is discussed. The presented approach allows for model-free learning of a desired gait pattern, where the torque values of hip, knee, and ankle actuators of an exoskeleton are learned from scratch to achieve a desired gait pattern, while considering a user's body characteristics, their existing gait pattern, and potential perturbations during their gait. A main benefit of this control method is its ability adapt the level of assistance needed as patients are wearing the device, and also as they progress in their individual physiotherapy goals. This therefore allows the exoskeleton to be trained and controlled for individual users. In this reinforcement learning based control scheme, a user's movements, and the exoskeleton's torque actions are included in the observation space, and torque actions for the exoskeleton motors are learned accordingly.

## 4.1 DRL Framework

The general reinforcement learning framework is represented as a process where an action $a$, is taken in a state $s$, and where a new state $s'$ as well as a reward for this action $r$ is then collected. In a value-based reinforcement learning approach, an agent attempts to find a policy $\pi$ that will lead to the greatest value. This value is comprised of the future expected reward, known as the return, the agent will receive after taking an action from the policy in the current state. In DRL specifically, these policy and value functions are represented by deep neural networks, which are defined as neural networks with two or more hidden layers, and in this case are formulated as fully connected multi-layer perceptrons [101].

As discussed in the Chapter 2, a DRL algorithm that is suitable for use in high dimension, continuous observation and action spaces is required. The DRL approach presented in this chapter utilizes the Deep Deterministic Policy Gradient (DDPG) algorithm [88] for end-to-end learning of a patient-specific torque-based controller for a lower body exoskeleton. DDPG has been used in environments with continuous high dimensional observation and action spaces, and it has been found to produce good results in various tasks, including those which pertain to the locomotion of simulated animals and bipeds [80, 88]. The algorithm discussed in this chapter is based on the one included in the Keras-RL library [101], and the specifics of this algorithm will be discussed in detail. The overall architecture of the DRL approach is presented in Figure 4.1.

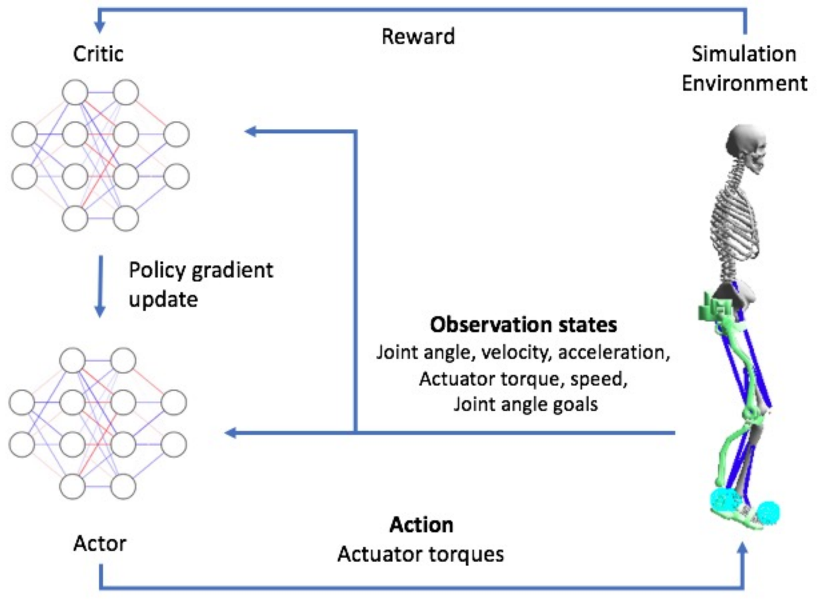

Figure 4.1: Overview of the DRL architecture for exoskeleton control.

The DDPG agent is primarily composed of an Actor and a Critic deep neural network to represent the aforementioned policy and value functions of the RL formulation, respectively [88]. It is primarily based on the advances made in the DQN algorithm [102], such as using a neural network to approximate the target Q-value. However DQN is limited by requiring lower-dimensional and discrete action spaces [88]. DDPG extends this to allow for high-dimensional, continuous actions. In DDPG the actor network takes actions based on its current policy and observations from the environment, and the critic evaluates these actions based on the observations and received reward, to determine their value. The simulation environment is composed of a simulated exoskeleton model paired to a musculoskeletal model, implemented in OpenSim-RL [103], a DRL platform originally developed for learning locomotion skills in musculoskeletal models. While a 3D model of the user, exoskeleton, and environment is required in this DRL formulation, in this model-free approach the dynamics model is not learned by the policy, with learning only conducted through the simulated model interacting with the environment. The details of the proposed DRL approach are discussed below.

### 4.1.1 Deep Deterministic Policy Gradient

DDPG is a model-free, off-policy, actor-critic deep reinforcement learning method [88]. In this RL framework, the learning agent gathers an observation state $s_t$ from an environment, takes an action $a_t$ based on the actor's policy, and receives a scalar reward, $r_t$, and the next state $s_{t+1}$, in each timestep, while maximizing the expected future discounted cumulative reward [88]:

$$R = \sum_{i=t}^{T} \gamma^{(i-t)} r_i(s_i, a_i) \tag{4.1}$$

where $\gamma \in [0\ 1]$ is a discounting factor used to emphasize the influence of future rewards rather than those immediately gathered. The critic, $Q(s, a)$, uses the Bellman equation to represent the value function, which describes the future expected return after taking actions in a particular state. The actor policy, $\mu(s|\theta^{\mu})$, is parameterized by mapping states to specific actions, and is updated by following the policy gradient. The policy gradient approach works by updating the policy through gradient descent based on the Q-value estimated by the critic network.

During training, the critic network, $Q(s, a|\theta^{Q})$, and the actor network, $\mu(s|\theta^{\mu})$, are first randomly initialized with weights $\theta^{Q}$ and $\theta^{\mu}$. In the DDPG framework, target networks are also established. This is based on the framework developed in the DQN algorithm [102], where target networks using copies of the original network weights were found to aid in learning. The copied actor and critic target networks are used to encourage a stable learning process and discourage divergence when calculating the target Q-value

[88], due to the potential issues arising from estimating the Q-value based on the same critic network that is being updated. These networks, $Q'$ and $\mu'$, have the following weights:

$$\theta^Q \to \theta^{Q'} \tag{4.2}$$

$$\theta^\mu \to \theta^{\mu'} \tag{4.3}$$

A replay buffer, $B$, used to store transitions of experience $(s_t, a_t, r_t, s_{t+1})$, is also initialized. While training, a random process $\mathcal{N}$ is initialized to encourage action exploration, based on an Ornstein Uhlenbeck (OU) Process [104], and a current observation state is received. This random process adds noise to the action space in a temporally correlated manner, which is thought to be helpful in environments containing inertial bodies [88].

The observations that make up the state in this environment consist of the real-time joint angles, velocities, and accelerations for each hip, knee and ankle of the human model, the exoskeleton's hip, knee and ankle actuator torque and velocity values, and the current goals. The goals at each timestep are comprised of the desired joint angles to be reached for the hip, knee, and ankle joints of the musculoskeletal model. Over a full episode of training, these joint angle goals represent one gait cycle of the desired gait pattern.

For each timestep, an action, $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$, is selected and executed based on the current actor policy $\pi$ and noise for exploration, and a reward, $r_t$, and new state, $s_{t+1}$ are obtained. In this framework, the actions are composed of torque actions on each actuator of the exoskeleton. The state transitions tuples $(s_t, a_t, r_t, s_{t+1})$ are stored in the replay buffer, and a target Q-value, $y_t$, is then obtained from the target networks $Q'$ and $\mu'$, and reward, $r_t$, using the Bellman equation [88]:

$$y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'}) \tag{4.4}$$

with the discount factor $\gamma$.

The critic is then updated by minimizing the Mean Squared Error (MSE) loss between the target Q-value and the current Q-value estimated by the critic network, over a random batch of state transition samples, $N$, from the replay buffer [88]:

$$L = \frac{1}{N}\sum_t (y_t - Q(s_t, a_t|\theta^Q))^2 \tag{4.5}$$

The actor policy network is then updated based on the sampled policy gradient [88]. The parameter optimizer for the actor and critic neural networks is based on the Adam optimizer [105], a commonly used method for stochastic gradient-based optimization, where a learning rate of 0.001 is used for the actor and critic networks.

The weights of target actor and critic networks are then updated based on the main actor and critic networks, but with a soft update, $\tau << 1$, so that they slowly track the main network weights with a time delay to encourage learning stability [88]:

$$\tau\theta^Q + (1-\tau)\theta^{Q'} \rightarrow \theta^{Q'} \tag{4.6}$$

$$\tau\theta^\mu + (1-\tau)\theta^{\mu'} \rightarrow \theta^{\mu'} \tag{4.7}$$

As each timestep in an episode progresses, the desired joint angle goals are updated to the next joint angle in the desired gait patterns. Eventually the agent learns to reach all joint angles in sequence in order to maximize its reward and control the desired gait pattern.

In this implementation of DDPG, the actor network is comprised of the state observations as inputs, three hidden layers of 32 neurons with rectified linear unit (ReLU) activation functions, and an output layer of actions with a sigmoid activation function. The critic network takes actions and state observations as inputs, and is comprised of three hidden layers of 64 neurons with ReLU activation functions, and an output layer with a linear activation function. This network architecture is based on the Keras-RL library's multi-layer perceptron policy for DDPG [101].

### 4.1.2 Reward Function

To allow the agent to find optimal actions, a reward function is defined to reward good actions and penalize poor actions. In this implementation the reward function is comprised of the following components: 1) a reward based on the offset between the current and desired joint angles, and 2) a penalty for exceeding a maximum or minimum joint angle. This reward function allows the exoskeleton to follow a desired gait pattern by reducing the overall error between the human model's current joint angles and the desired joint angles, while providing additional penalties when an undesirable joint angle state is reached. In this framework, a desired hip, knee and ankle joint angle is obtained from the desired gait pattern at each timestep, where the desired gait pattern is updated at the beginning of each episode.

The total cumulative reward, $R_E$, for all joints, for each episode, $E$, over a number of timesteps, $n$, is defined as:

$$R_E = \sum_{i=0}^{n}(w_h r_{i_h} + w_k r_{i_k} + w_a r_{i_a}) \tag{4.8}$$

where $r_i$ is the reward function for a particular joint, and $h, k, a$ represent the hip, knee, and ankle joints, respectively, with $w$ the weight for each joint. The reward for each timestep is defined as:

$$r_i = w_F \mathcal{F}(q_i) + w_G \mathcal{G}(q_i) \tag{4.9}$$

where $\mathcal{F}$ and $\mathcal{G}$ are the components of the reward function, and $w_F$ and $w_G$ their weights. $\mathcal{F}$ represents a reward based on the current joint angle offset, and $\mathcal{G}$ represents a penalty for exceeding a maximum or minimum defined joint angle.

$\mathcal{F}$ is defined as a Gaussian function:

$$\mathcal{F}(q_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{d-\mu}{\sigma}\right)^2} \tag{4.10}$$

where $\mu$ and $\sigma$ are the mean and standard deviation respectively, and $d$ is defined as the absolute difference between the current joint angle, $q_i$, and the current desired joint angle, $q_{d_i}$:

$$d = |q_i - q_{d_i}| \tag{4.11}$$

To assist in the learning process, a penalty is added if a defined maximum or minimum joint angle is exceeded during exploration, $\mathcal{G}(q_i)$:

$$\mathcal{G}(q_i) = -M(y_{max}) - M(y_{min}) \tag{4.12}$$

where,

$$M(y) = \begin{cases} 0 & y \leq 0 \\ y & y > 0 \end{cases} \tag{4.13}$$

$$y_{max} = q_i - q_{max} \tag{4.14}$$

and,

$$y_{min} = q_{min} - q_i \tag{4.15}$$

Above, $q_{max}$ and $q_{min}$ are the defined maximum and minimum joint angles, respectively, the values of which are set as 20 degrees beyond the maximum and minimum desired joint angles. With the use of a ramp function $M(y)$, the penalty will only be applied if the maximum or minimum joint angles are exceeded.

### 4.1.3 User-Exoskeleton Simulation Environment

The simulation environment used for training the controller incorporates both a 3D musculoskeletal human model and a 3D exoskeleton model. OpenSim-RL [103] is used to provide the reinforcement learning environment and agent. This RL environment is based on OpenAI Gym [106], a popular platform for deep reinforcement learning research and benchmarking.

OpenSim-RL uses the OpenSim API platform to provide the simulation of the musculoskeletal model. This musculoskeletal model is derived from OpenSim's Gait2392 model, which is anatomically accurate and scaled to the proportions of a human with a height of 1.8 m and mass of 76.16 kg [90]. The OpenSim platform has previously been used to perform dynamic simulations of locomotion and muscle control analyses [107–109]. The muscles on this musculoskeletal model are based on a Hill-type muscle model [110], each containing a muscle-tendon unit with specific properties such as muscle fiber length, tendon slack length and maximum isometric force. Each muscle is able to provide both active and passive forces. The Simbody physics engine [111] is used to provide the multibody dynamics for the motion and interaction of the musculoskeletal bones, joints, muscles, and exoskeleton linkages in the OpenSim-RL environment. The simulation environment is presented in Figure 4.2.

The exoskeleton model used in this simulation is a 3D hip-knee-ankle exoskeleton model adapted from [109] that contains a thigh, shank, and foot linkage, as well as a hip attachment, where one leg is modelled in the present method to increase the simulation speed. This is added to the OpenSim environment and secured to the musculoskeletal model, with torque actuators added to each joint, as would be found on a hip-knee-

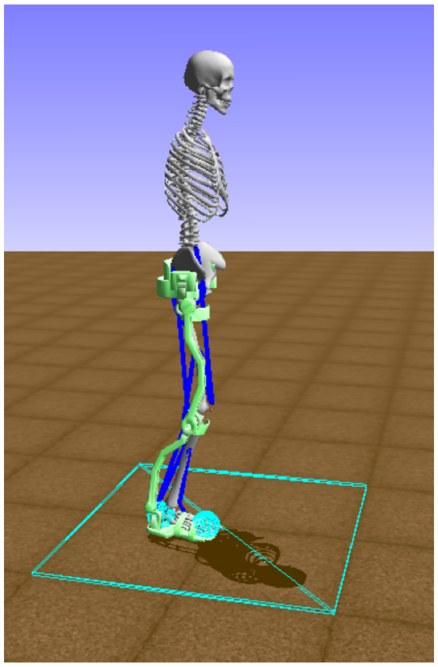

Figure 4.2: Simulation environment in OpenSim-RL, composed of a musculoskeletal model with a hip-knee-ankle exoskeleton.

ankle exoskeleton. In this simulation these actuators are modeled as idealized torque actuators. Previous studies have used similar actuators to model orthoses and exoskeletons [108,109,112,113]. Additionally, each exoskeleton joint is constrained to a specified range of motion, in order to simulate a range similar to real exoskeleton joints [114], and prevent unrealistic and dangerous movements during exploration or execution, such as hyperextension in the knee. The forces incorporated in the simulation environment consist of ground reaction forces on the bottom of the exoskeleton foot plates, acceleration due to gravity, and joint limit constraints. The ground reaction forces are modeled as Hunt Crossley Forces in the OpenSim environment. This force model creates force interactions between contact spheres that are applied at the exoskeleton's foot plate, as seen in Figure 4.2, and a half sphere established in the floor plane. The exoskeleton parameters are summarized in Table 4.1.

A hip-knee-ankle joint torque pattern, also obtained from the Gait2392 model, is applied to the joints of the musculoskeletal model in order to simulate a post-stroke

| **Link** | **Thigh** | **Shank** | **Foot** |
|---|---|---|---|
| Length [m] | 0.47 | 0.42 | 0.26 |
| Mass [kg] | 1.5 | 1.5 | 0.5 |
| $I_{xx}$ [kg m$^2$]* | 0.015 | 0.059 | 0.337 |
| $I_{yy}$ [kg m$^2$] | 0.005 | 0.003 | 0.009 |
| $I_{zz}$ [kg m$^2$] | 0.014 | 0.057 | 0.338 |
| **Joint Limits** | **Hip** | **Knee** | **Ankle** |
| Maximum [°] | 80 | 10 | 40 |
| Minimum [°] | -80 | -100 | -40 |

*The $I_{xx}$, $I_{yy}$ , $I_{zz}$, are the moment of inertia tensor's diagonal values.

Table 4.1: Exoskeleton parameters in OpenSim-RL.

individual's baseline gait pattern. This is done in place of active muscle forces, in a similar fashion to [113]. This is accomplished through a Coordinate Actuator added to each musculoskeletal joint, which provides force around a defined joint coordinate in OpenSim. Simulating the baseline motion through joint torque rather than active muscle forces allowed for a further increased simulation speed. The corresponding joint angle pattern in the sagittal plane of the model can be seen in Figure 4.3. To obtain the desired gait pattern, also seen in Figure 4.3, an inverse kinematic process is performed from motion capture data provided by OpenSim [90], to generate joint angle patterns for the hip, knee, and ankle. This desired gait pattern is modeled after healthy user data provided by OpenSim in the Gait2392 model [91]. In this chapter, this desired gait pattern is used to validate the DRL approach, however additional user data and corresponding gait patterns from clinical studies of healthy individuals will be used in the following chapter.

## 4.2 Experiments

In order to validate the DRL method for exoskeleton control of gait patterns, the DDPG agent is first trained with the included baseline and desired gait patterns. The trained policy is then tested for its ability to control the exoskeleton joints to follow the desired gait pattern, and slight unseen variations of this desired gait pattern.

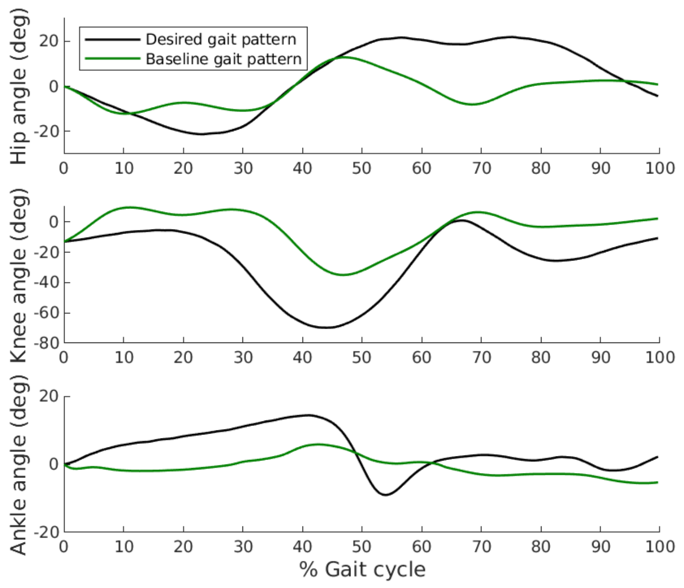

Figure 4.3: Desired gait pattern and the user baseline gait pattern used in training.

### 4.2.1 Training

Training was conducted in the OpenSim-RL framework, with the user baseline gait pattern and a desired gait pattern as shown in Figure 4.3. Training was implemented on an Intel Core i7-8700 CPU for over 2,200 episodes, with 308 timesteps per episode, which represents one full gait cycle of the desired gait pattern. After 1,300 episodes, the cumulative reward reached an average of approximately 63, as seen in Figure 4.4. The parameters and weights for the reward function were defined as $\sigma = 5, \mu = 0, w_F = 1, w_G = 0.002, w_h, w_k, w_a = 1$. The hyperparameters for the training and networks are as follows: $\gamma = 0.99, \tau = 0.001$, and a learning rate of 0.001.

### 4.2.2 Testing

Testing to validate this approach was conducted in two stages: Stage 1, with the baseline and desired gait pattern used in training, and Stage 2, with adjusted baseline and desired gait patterns from training. Stage 2 testing was used to investigate robustness to slight changes in the gait patterns. To do so, the baseline joint torque and desired joint angle

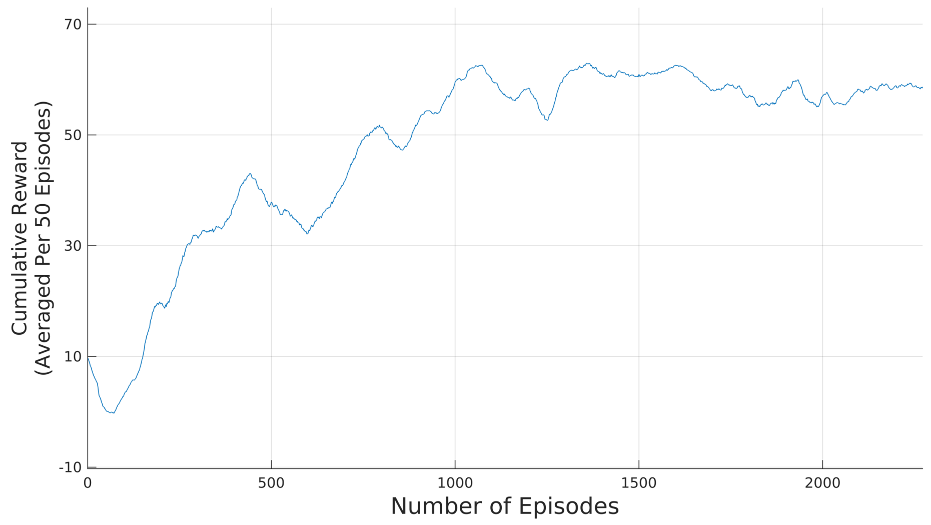

Figure 4.4: Cumulative reward per episode during training, averaged per 50 episodes.

values for each joint were adjusted by a scale factor. For both stages, the exoskeleton performed 10 full gait cycles of the learned joint torque pattern, with the resulting hip, knee, and ankle joint angles of the musculoskeletal model averaged over the cycles.

### 4.2.3 Results

The results of the tests in Stage 1 are presented in Figure 4.5. The descriptive statistics for the error are also presented in Table 4.2. It can be seen that the trained exoskeleton was able to follow the desired gait pattern closely throughout the gait cycle. The mean absolute error ranged from 1.06 degrees for the ankle to 2.63 degrees for the knee. The knee joint exhibited the largest error as it also has the largest range of motion of the three joints.

| Joint | Mean absolute error (degrees) | Standard deviation (degrees) |
|---|---|---|
| Hip | 1.82 | 1.01 |
| Knee | 2.63 | 1.72 |
| Ankle | 1.06 | 1.05 |

Table 4.2: Stage 1 Gait Pattern Error.

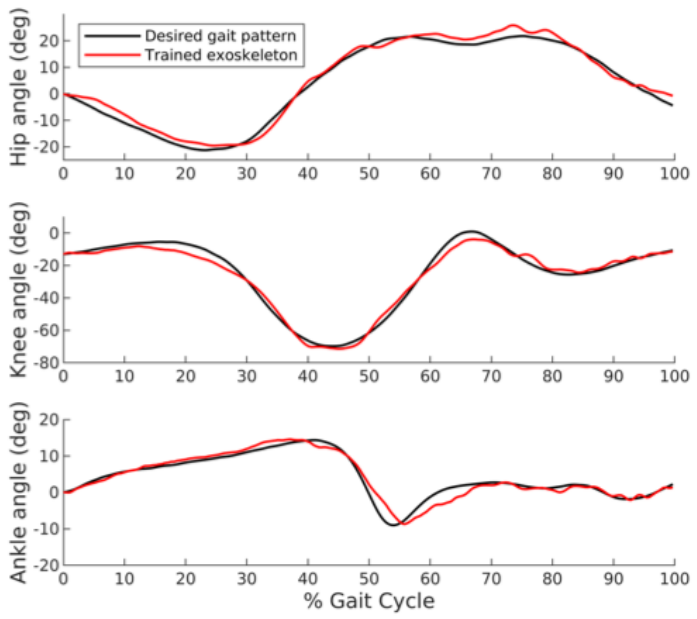

Figure 4.5: Stage 1 desired gait pattern and the trained exoskeleton joint angle patterns for the hip, knee, and ankle joints.

The Stage 2 results for the scaled desired joint angle adjustments are presented in Figure 4.6, with the descriptive statistics presented in Table 4.3. Scaling factors of 0.5 and 0.8 applied to the desired joint angles were used to demonstrate deviations in the desired gait pattern that would be applicable during different stages of gait rehabilitation, and demonstrate initial robustness in this DRL approach. It can be seen that the adjusted desired gait patterns were followed closely even though they were not used in training, with the mean absolute error ranging between 0.73 degrees for the ankle to 1.72 degrees for the knee. As the range of motion of the desired gait patterns in the Stage 2 tests was lower, the errors were also found to be lower across all joints. In general, the error ranges from the results are comparable to existing model-based adaptive and RL-based joint control methods, which have ranged from 1 degree to 5 degrees, i.e., [68], [71]. The results validate the effectiveness of this DRL method to learn and follow a desired gait pattern while accounting for a user's baseline gait pattern, as well as the ability to handle small deviations from the trained desired pattern.

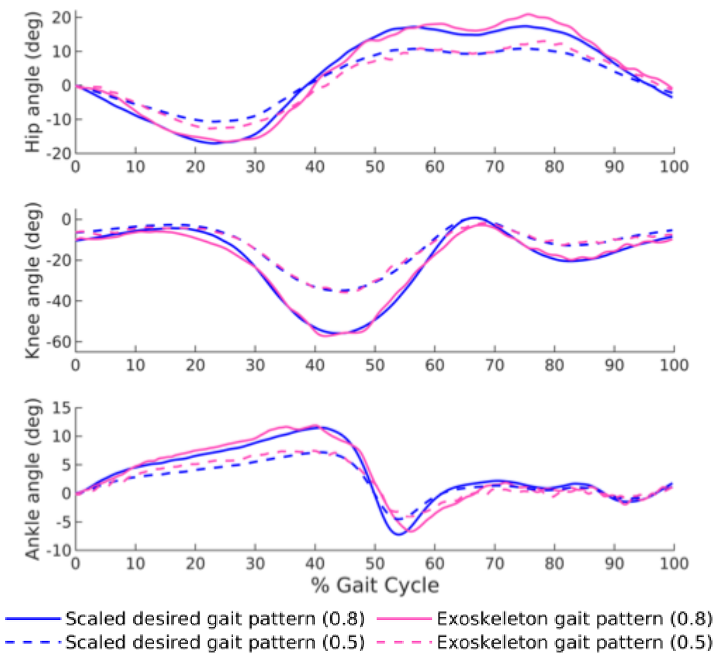

Figure 4.6: Stage 2 desired and trained exoskeleton gait patterns with the desired gait patterns scaled by a factor of 0.5 and 0.8

| | Joint | Mean absolute error (degrees) | Standard deviation (degrees) |
|---|---|---|---|
| Scaled desired gait pattern (0.8) | Hip | 1.58 | 0.98 |
| | Knee | 1.72 | 1.3 |
| | Ankle | 0.92 | 0.77 |
| Scaled desired gait pattern (0.5) | Hip | 1.5 | 0.92 |
| | Knee | 1.08 | 0.85 |
| | Ankle | 0.73 | 0.51 |

Table 4.3: Stage 2 Gait Pattern Error.

In Chapter 5, the presented DRL method will be expanded to include training on a wide variety of desired gait patterns gathered from clinical data of a range of healthy individuals. As well, data from post-stroke individuals will be used as the baseline gait pattern data to further validate this approach for gait rehabilitation. These additions are beneficial as they will allow the developed DRL controller to be robust to varying users and exoskeleton gait patterns that would be desired in physiotherapy sessions. Additionally, updates to the DRL method will be discussed which further increases the accuracy of the controller.

## 4.3 Chapter Summary

In this chapter, the development and testing of an end-to-end, model-free deep reinforcement learning method for exoskeleton control of gait patterns with a musculoskeletal model was presented. The use of DRL allows for model-free control of gait patterns in high-dimensional, continuous observation and action spaces, where the user's baseline gait pattern is included in the environment. This DRL solution used the Deep Deterministic Policy Gradient algorithm, and was trained and tested using the OpenSim-RL platform. The details of the DRL agent, as well as the training environment and reward function were presented. Results to validate this approach were then discussed, where low error was obtained for all joints, both in testing with the desired gait pattern used in training, and with slight adjustments to the baseline and desired gait patterns.

# Chapter 5

# Generalized DRL Control

In this chapter, a generalized deep reinforcement learning control approach is presented to allow for adaptive exoskeleton control of any desired gait pattern, that is robust to varying post-stroke exoskeleton users. This is accomplished through extending the DRL method presented in Chapter 4. To do so, a set of existing user data for both the baseline gait patterns and the desired gait patterns are included in training. These datasets are collected from clinical studies of post-stroke individuals and healthy users. Testing sets of desired gait patterns both seen and unseen in training are then used to validate the controller's ability for on-the-fly adaptation of desired gait patterns. Additionally, updates to components of the DRL method presented in the previous chapter are discussed, as well as experiments and demonstrated results for this updated approach.

## 5.1 User Data for Training

### 5.1.1 Desired gait patterns

To obtain the desired gait patterns for the exoskeleton to learn to follow during training, joint angle data is obtained from [115], a clinical study that observed the gait of 20 healthy adult subjects ranging in age from 22 to 72. In this chapter, this desired gait pattern set will be referred to as Desired-1. The joint angles from these gait patterns are used as the goals for each joint, for each episode timestep during training, as discussed in Chapter 4. Each episode consists of one full gait cycle of the desired gait pattern. In [115] the subjects were first recorded with motion capture during 5 trials of walking at a self-selected natural walking speed. They were then instructed to walk at a self-selected slower speed, and an increased speed. The subjects in this study had the following mean gait characteristics: a step length of 0.7 m, stride length of 1.4 m, stride time of 1.1 s, a double support time of 13% of their gait cycle, and a swing time of 38% of their gait cycle. After the data was captured, the mean of these subjects's gait patterns were grouped into five categories of walking speeds, normalized to their body heights ($v/h$). These categories, including their speed were: very slow ($v/h < 0.6$), slow ($0.6 \leq v/h < 0.8$), medium ($0.8 \leq v/h < 1$), and fast ($v/h > 1$), with a mean natural walking speed also collected [115].

These gait patterns are represented as one gait cycle stride from heel-strike to heel-strike. In this chapter the episode length, and thus duration of the gait cycle, is based on the various gait pattern speeds. Joint angle patterns of $\pm 1$ standard deviation from each speed's mean gait pattern were also provided, and are also used in training to provide additional variation in the desired gait pattern goals. This corresponded to an average maximum range of motion in the sagittal plane of 45.6°, 61.5°, and 34.4° for the hip, knee, and ankle joints respectively. These gait patterns represent a wide range of possible healthy gait pattern data, as seen in [116, 117], with similar joint angle ranges. The mean of these gait patterns for each speed can be seen in Figure 5.1.

To add further robustness to the desired gait pattern data, the amplitude of these gait patterns are scaled between a factor of 0.5 and 1.2 in order to simulate various stages of gait recovery towards a typical gait pattern, for a range of users with varying motor function. As a post-stroke individual would have limited gait function in their affected leg [3, 118], and thus a more limited amplitude of flexion [3], the range of motion for each joint in these patterns is scaled down, in order to represent varying levels of difficulty

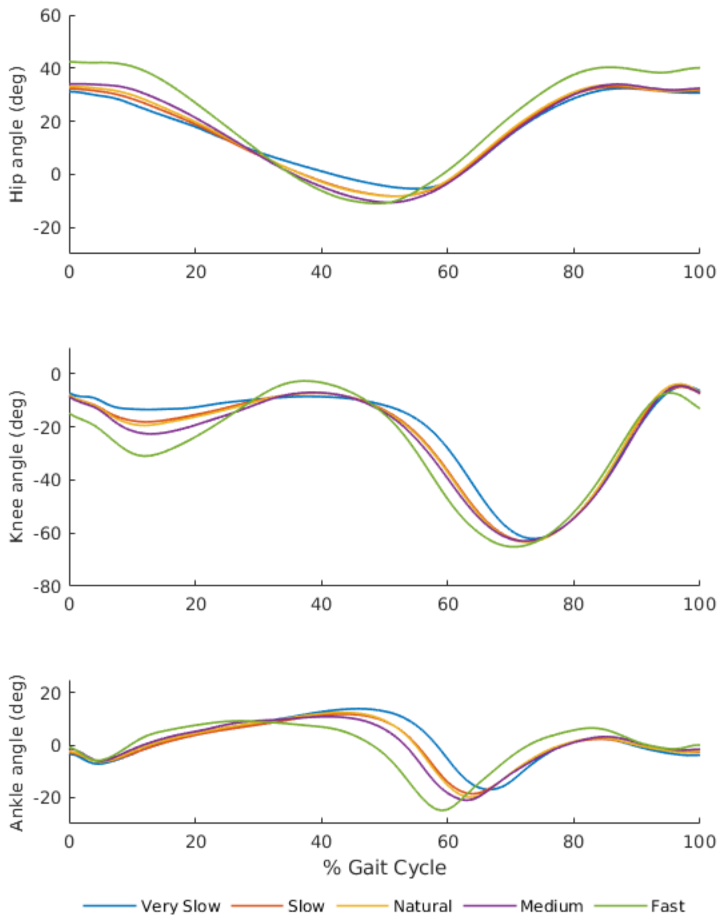

Figure 5.1: Mean desired joint angle patterns from healthy individuals for each gait speed [115].

in the desired gait pattern. A more typical gait pattern involves increased flexion in the knee [119], increased stride length, and a general increased range of motion [118]. Therefore this scaling allows for a training set that represents a range of desired gait patterns to match the current performance of the user, and provide more typical and challenging gait patterns as they progress in their recovery. The gait patterns are also scaled up slightly to provide a larger range of motion for desired gait patterns that can further challenge the user if necessary, and also add additional variations in the training set. In a clinical scenario, a physiotherapist would choose the scale of desired gait pattern that would best fit the current motor performance of the patient, and progressively increase this as their gait and range of motion recovered. The range of scaled gait patterns for each mean gait pattern speed is shown in Figure 5.2, where the plotted line represents the provided mean gait pattern from [115], with the shaded region representing the scaled gait patterns from 0.5 to 1.2.

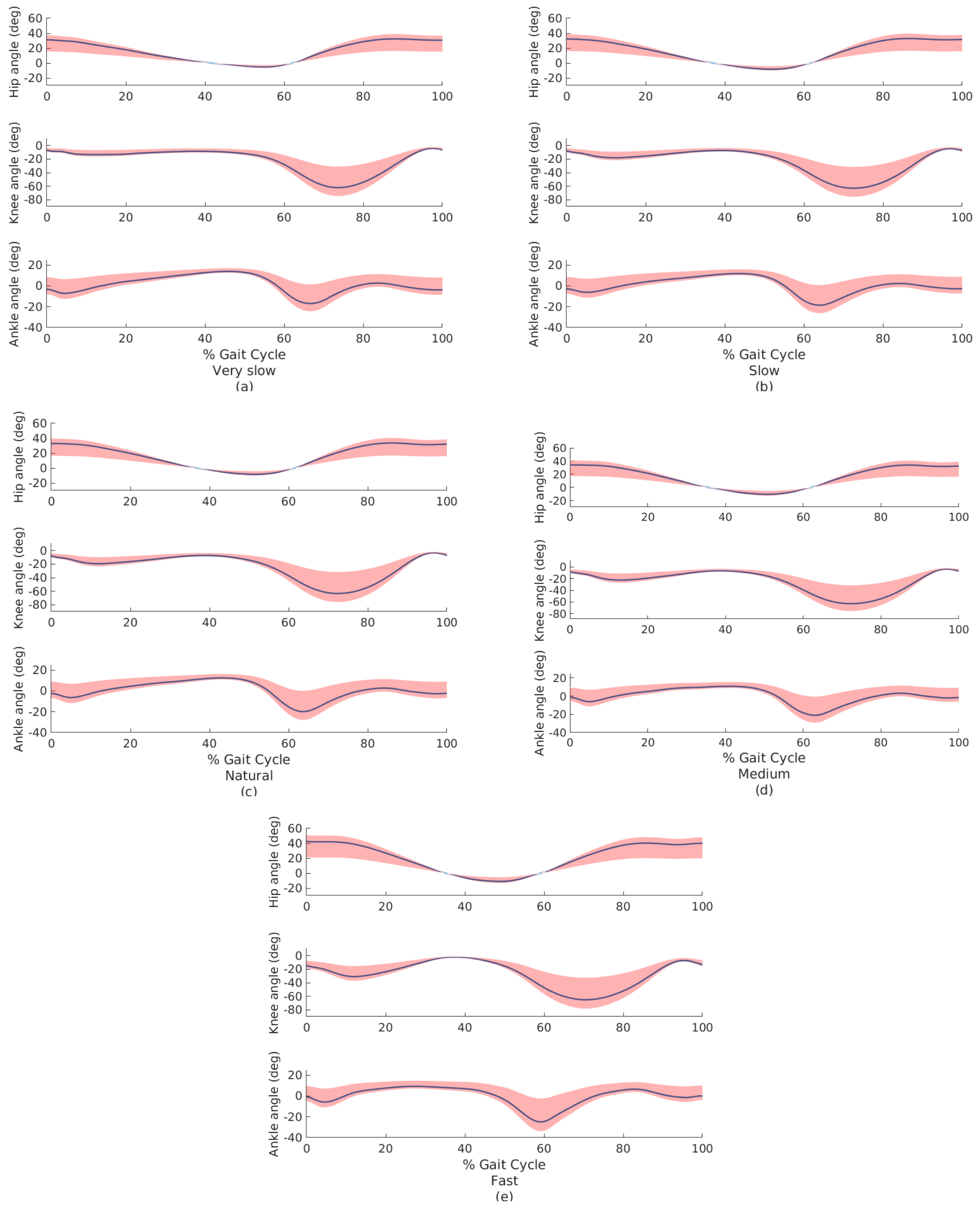

Figure 5.2: Mean desired joint angle patterns for each gait speed provided in [115], with the shaded region indicating the scaling from 0.5 to 1.2.

### 5.1.2 Baseline gait patterns

To allow for a greater degree of subject-specific control and gait training of post-stroke individuals, a set of baseline gait patterns representing multiple post-stroke individuals is also included in training. To obtain the baseline motions for the musculoskeletal model, gait patterns in the sagittal plane from 6 chronic post-stroke individuals were obtained from [120], a clinical study evaluating the gait and motor recovery of stroke survivors. These gait patterns were captured in [120] by using the Vicon Motion Capture system as subjects walked at a self-selected walking speed on a split-belt treadmill, and was then converted to an OpenSim motion file format [120]. The subjects ranged in age from 54 to 70 and were between 12 months and 55 months past the date of their stroke. Selection criteria was defined as at least 5 months post-stroke and the capability to walk for 5 minutes at a self selected speed in the presence of gait deficits. The post-stroke individuals included in this study had hemiparesis on one side of their body, which is indicated in the data, and these individuals had a mean self-selected gait speed of approximately 0.4 m/s. Information for each subject, as named in [120], can be seen in Table 5.1.

| **Subject** | **Gender** | **Age (yrs)** | **Side of Hemiparesis (L/R)** | **Time since Stroke (months)** | **Self-selected gait speed (m/s)** |
|---|---|---|---|---|---|
| **S98** | M | 66 | R | 19 | 0.3 |
| **S108** | M | 70 | L | 21 | 0.5 |
| **S110** | F | 65 | R | 15 | 0.3 |
| **S128** | F | 65 | R | 18 | 0.5 |
| **S129** | F | 54 | R | 55 | 0.5 |
| **S136** | F | 58 | R | 12 | 0.3 |

Table 5.1: Characteristics of post-stroke subjects used for baseline gait patterns [120].

For the presented DRL scheme, the joint angle data from the affected leg was collected for one gait cycle from the OpenSim motion file provided for each subject, and scaled in a similar manner as was done for the desired gait patterns in Section 5.1.1, to simulate various levels of function and recovery for each post-stroke individual. Joint torque data was then obtained for each of these gait patterns, and the torque patterns were then applied to the musculoskeletal model through torque actuators added to the model's joints in OpenSim, to simulate the user's baseline motion as done in Chapter 4. The baseline joint angle patterns for one cycle of each subject's gait as provided in [120], can be seen in Figure 5.3.

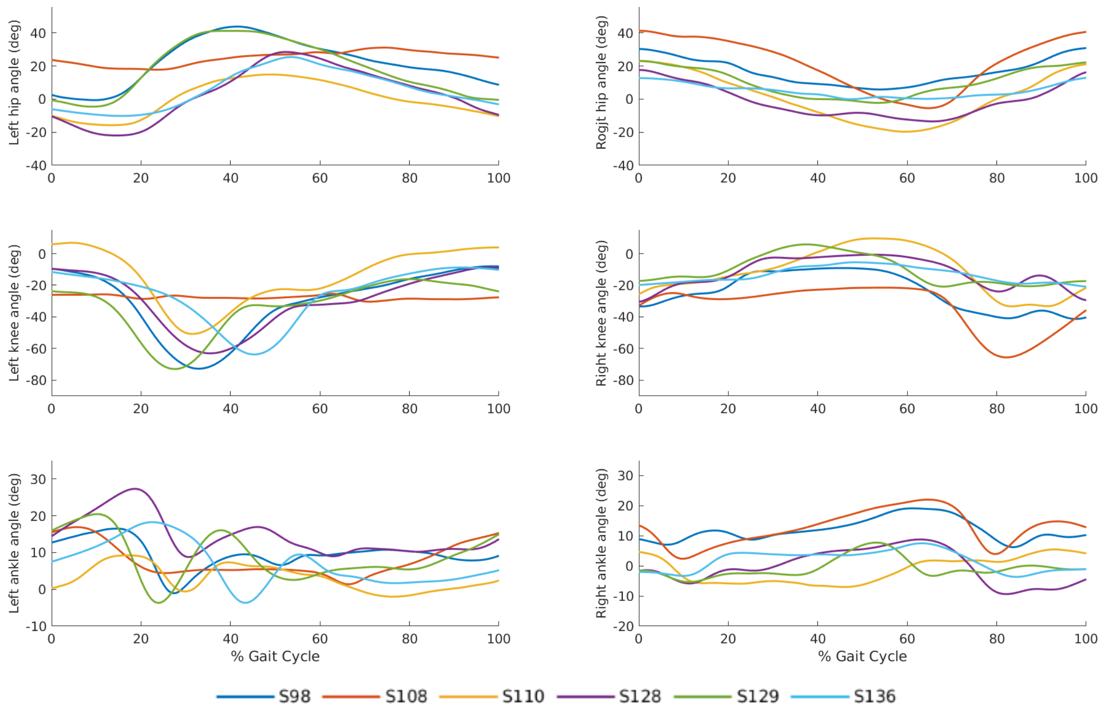

Figure 5.3: Baseline joint angle patterns for post-stroke individuals [120].

## 5.2 Updates to the DRL Method

In order to adapt to the various desired gait patterns used in this chapter, modifications were made to the DRL agent described in Chapter 4. This included alterations to the noise used for exploration, components of the reward function, and the deep neural network architecture. With the exception of the maximum torque values for the exoskeleton actuators, the OpenSim-RL environment, along with the 3D exoskeleton and musculoskeletal model were unchanged from Chapter 4.

### 5.2.1 Exploration

A Gaussian white noise process with $\sigma = 0.3$ is used to add noise to the action space during training. This choice of action noise is supported in a more recent update to the DDPG algorithm [121], and this allowed for greater exploration than the Ornstein-Uhlenbeck process used in Chapter 4 provided. Higher maximum torque values of 220 Nm were used for each exoskeleton actuator to account for desired gait patterns with a larger range of motion and speed included in the clinical data from [115], as well as the torque required to achieve higher accuracy during the toe off motion of the gait cycle. This also corresponds to a similar maximum torque the motors on the H2 exoskeleton with attached harmonic drive gear boxes can provide [16]. The DRL agent using the Ornstein-Uhlenbeck process with these higher torque values had difficulty exploring the expanded action space sufficiently, and would often become stuck in local minima at the beginning of training. The use of Gaussian white noise on the actions addressed this and allowed for improved exploration.

### 5.2.2 Reward Function

In the present reward function, a linear function is used to penalize the joint angle error. Through training the DRL agent it became clear that while the Gaussian function, $\mathcal{F}(q_i)$, used in Chapter 4 allowed the agent to find the range of highly rewarded actions, and thus low-error, achieving further minimal error was more challenging. This was due in part to the shape of the Gaussian function curve, where the slope decreases close to its mean, therefore not rewarding the agent sufficiently for progressively lower error. Due to this sparse reward, the agent had difficulty reducing the error further.

In place of the Gaussian function, $\mathcal{F}(q_i)$ is updated to a linear function:

$$\mathcal{F}(q_i) = -5d \tag{5.1}$$

where $d$ is defined as the absolute difference between the actual joint angle, $q_i$, and the desired joint angle, $q_{d_i}$, at the current timestep:

$$d = |q_i - q_{d_i}| \tag{5.2}$$

The reward function component as described above applies a penalty based on the joint angle error, and reduces the penalty to the agent in a linear nature as the error decreases. A steep slope was chosen for this function to more positively influence lower error. This was combined with the previously defined penalty, $\mathcal{G}(q_i)$, in Chapter 4, for exceeding a defined maximum or minimum joint angle (4.12).

The full reward function at each timestep, $r_i$, for a given joint is therefore:

$$r_i = w_F\mathcal{F}(q_i) + w_G\mathcal{G}(q_i) \tag{5.3}$$

with weights $w_F$ and $w_G$.

This ultimately led to improved performance and increased accuracy across all joints and gait patterns, as will be discussed below.

### 5.2.3 Actor and Critic Networks

In the Actor neural network architecture, a hyperbolic tangent (tanh) activation function is used in the output layer of the network in place of a sigmoid activation function. This allows for a range of normalized actions within -1 to 1, whereas the sigmoid function constrains this from 0 to 1 with added noise, which affected the range of the actuator torque outputs. It has been discussed in [122] that the use of a tanh activation function also provides greater stability and performance in training than the sigmoid activation function. Furthermore, action clipping was added to the action outputs at each timestep, so that the actions with added exploration noise from the actor network did not exceed the maximum torque of the actuators. This ensured that the observations of these actions were within the same range as well. Additionally, the absolute error, $d$, at each timestep of an episode for each joint was added to the observation space for input to the actor and critic networks. These additions were found to further aid in improving the accuracy of the controller.

## 5.3 Experiments

### 5.3.1 Training

During training, a desired gait pattern from the Desired-1 gait set, a baseline gait pattern, and a scale factor for each was randomly chosen each episode. The duration of each episode was also set based on the speed of the selected desired gait pattern. Throughout the training process, the agent learns to adapt to these desired and baseline gait patterns.

The agent was trained for 2 million timesteps, corresponding to over 10000 episodes, on an Intel Core i7-8700 CPU, with each episode consisting of between 147 and 247 timesteps, depending on the speed and duration of the desired gait pattern. The reward plot for the training run can be seen in Figure 5.4. The parameters and weights for the reward function were defined as $\sigma = 0.3, \mu = 0, w_F = 1, w_G = 1, w_h = w_k = w_a = 1$, and the hyperparameters for training and the DDPG algorithm were as follows: $\gamma = 0.99, \tau = 0.001$, and a learning rate of 0.001, based on the parameters included in [88, 101].

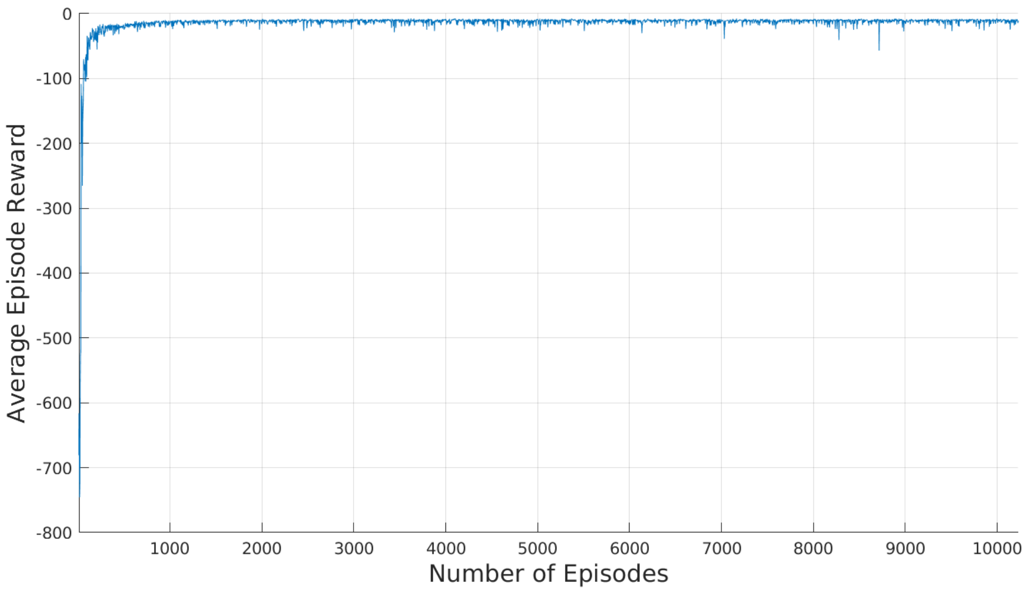

Figure 5.4: Average reward per episode during training.

### 5.3.2 Testing

In order to validate this approach, testing was divided into two stages. In Stage 1, the trained agent was first tested on the set of mean desired gait patterns that were used in training, Desired-1, as described in section 5.1.1. The agent is tested for 400 episodes, corresponding to 400 gait cycles, with each desired gait pattern and scale factor tested 10 times, and the baseline gait pattern and scale randomly chosen each episode. In Stage 2, the agent is then tested on a set of unseen desired gait patterns, obtained from [123], a study that contains gait information for healthy adult subjects of a similar age range to Desired-1, walking at self-selected natural, slow and fast speeds. Gait data from this set was collected in a similar manner to [115], and joint angles for one gait cycle from heel-strike to heel-strike, for each subject, walking trial, and joint are provided. This set of unseen desired gait patterns will be referred to in this chapter as Desired-2. The agent was tested for another 400 episodes with 5 mean desired gait patterns from this set at varying speeds, with each desired gait pattern and scale tested 10 times, and the baseline gait pattern and scale randomly chosen each episode. Finally, an additional test was then performed to validate the assist-as-needed nature of this controller.

### 5.3.3 Results

#### Stage 1: Desired gait patterns used in training

The results of the testing in Stage 1 with the desired gait pattern set used in training, Desired-1, can be seen in Table 5.2, where the descriptive statistics are shown for the full range of testing. In this table, the mean and standard deviation for each joint is calculated from the mean absolute error of each episode in testing. It can be seen that the overall mean absolute error was low for the tests from Desired-1, with the error ranging from 0.23° for the ankle to 0.47° for the knee. The overall mean error from these tests was found to be substantially lower than the experiments from both stages in Chapter 4. In Figure 5.5, the tracking performance for the desired gait patterns at various scales can be seen for each desired gait pattern speed. From these plots it is evident that all desired gait patterns tested were followed closely by the trained exoskeleton, where the full range of motion was reached.

| Joint | Mean absolute error (degrees) | Standard deviation (degrees) |
|---|---|---|
| Hip | 0.36 | 0.14 |
| Knee | 0.47 | 0.17 |
| Ankle | 0.23 | 0.097 |

Table 5.2: Gait pattern error for the Stage 1 tests, with the Desired-1 testing set seen in training.

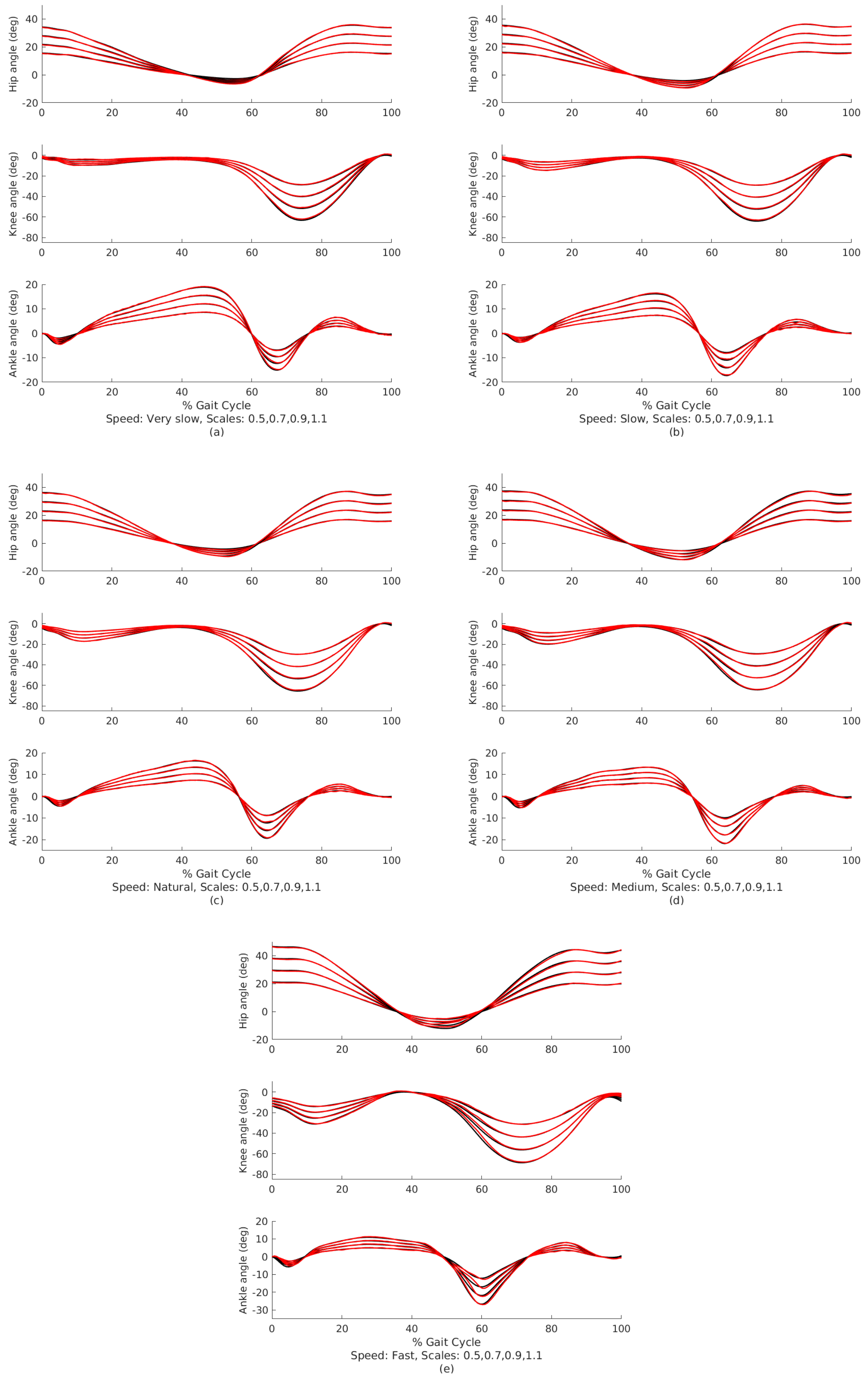

Figure 5.5: Stage 1 desired gait patterns (black) and the trained exoskeleton gait patterns (red) for each speed, with desired gait patterns scales between 0.5 and 1.1.

In Figure 5.6 a boxplot of the the error across all tested gait pattern speeds in Stage 1, for each joint can be seen. The error was found to be the low overall for all speeds, while slightly higher for the fastest gait speed. This is due in part to the larger range of motion of the fastest gait pattern, as seen in Figure 5.1. However, as shown in Figure 5.5, the desired gait patterns were still followed closely at all speeds and reached the desired range of motion. In Table 5.3, the mean error for each speed is shown, where they were found to be within a range of ± 0.15°, 0.16°, and 0.11°, for the hip, knee, and ankle joints respectively. Additionally, the mean absolute torque that was required to track the gait patterns can be seen in Table 5.4, calculated over a moving average of 10 episode steps to account for the noise in individual torque actions. A higher average torque can also be seen for the fastest gait pattern.

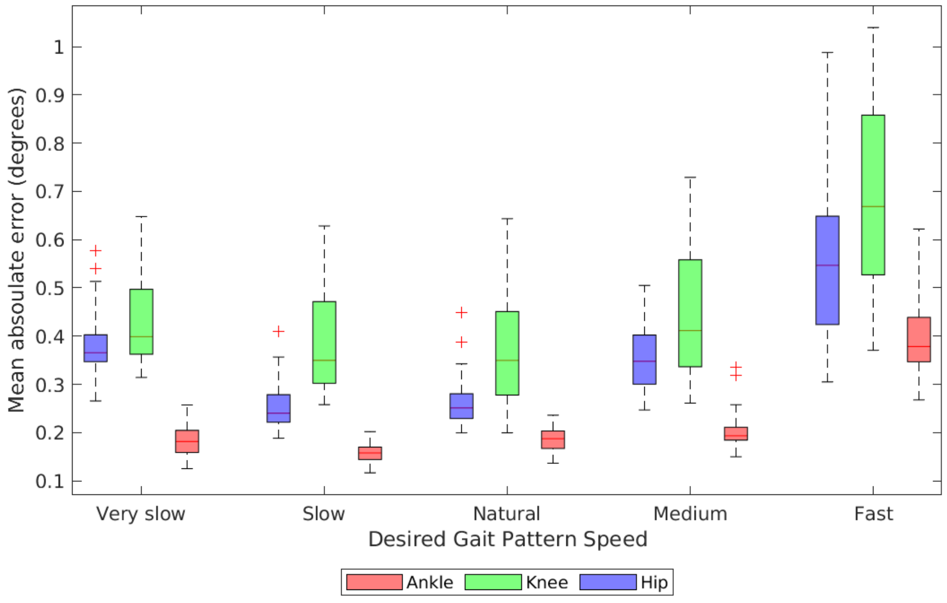

Figure 5.6: Stage 1 mean absolute error of each desired gait pattern speed for each hip, knee, and ankle joint.

| | **Mean absolute error (deg)** | | | | |
|---|---|---|---|---|---|
| **Joint** \ **Speed** | Very slow | Slow | Natural | Medium | Fast |
| Hip | 0.38 | 0.25 | 0.26 | 0.36 | 0.55 |
| Knee | 0.43 | 0.39 | 0.37 | 0.45 | 0.69 |
| Ankle | 0.18 | 0.16 | 0.19 | 0.2 | 0.4 |

Table 5.3: Stage 1 mean absolute error for each desired gait pattern speed.

| | **Mean absolute torque (Nm)** | | | | | **Maximum Torque (Nm)** |
|---|---|---|---|---|---|---|
| **Joint** \ **Speed** | Very slow | Slow | Natural | Medium | Fast | |
| Hip | 50.4 | 53.7 | 56.3 | 63.2 | 70.7 | 220 |
| Knee | 34.8 | 36 | 38.5 | 42.9 | 49.7 | 220 |
| Ankle | 25.8 | 26.2 | 26.9 | 30.4 | 38.1 | 220 |

Table 5.4: Stage 1 mean absolute torque for each gait speed tested.

In Table 5.5 the mean error across all the tested baseline gait patterns for each joint can be seen. A corresponding box plot is shown in Figure 5.7, and in Figure 5.8, the error for each baseline gait pattern scale is shown. Although it can be seen that the overall mean error varied between baseline subjects and scales, and was slightly higher for the subjects and joints with a larger range of motion, such as S110 in the knee, there did not appear to be a direct correlation between the baseline gait pattern or its scale and the error. Furthermore, the mean error for each baseline subject, as seen in Table 5.5, is low overall, with mean absolute errors within a range of $\pm$ 0.04°, 0.045°, and 0.025° between subjects for the hip, knee, and ankle joints, respectively. This supports the robustness of this controller for use with multiple post-stroke individuals and their individual level of motor function and gait recovery.

| | **Mean absolute error (deg)** | | | | | |
|---|---|---|---|---|---|---|
| **Joint** \ **Subject** | S98 | S108 | S110 | S128 | S129 | S136 |
| Hip | 0.35 | 0.34 | 0.41 | 0.33 | 0.4 | 0.35 |
| Knee | 0.46 | 0.42 | 0.51 | 0.48 | 0.47 | 0.47 |
| Ankle | 0.21 | 0.2 | 0.23 | 0.22 | 0.25 | 0.24 |

Table 5.5: Stage 1 mean absolute error for each baseline gait pattern tested.

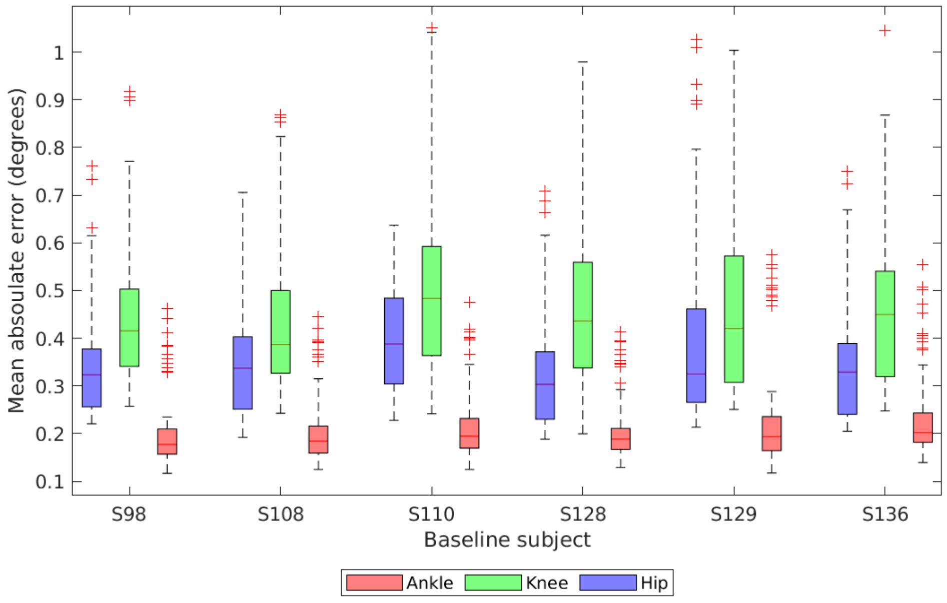

Figure 5.7: Stage 1 mean absolute error for each baseline gait pattern tested.

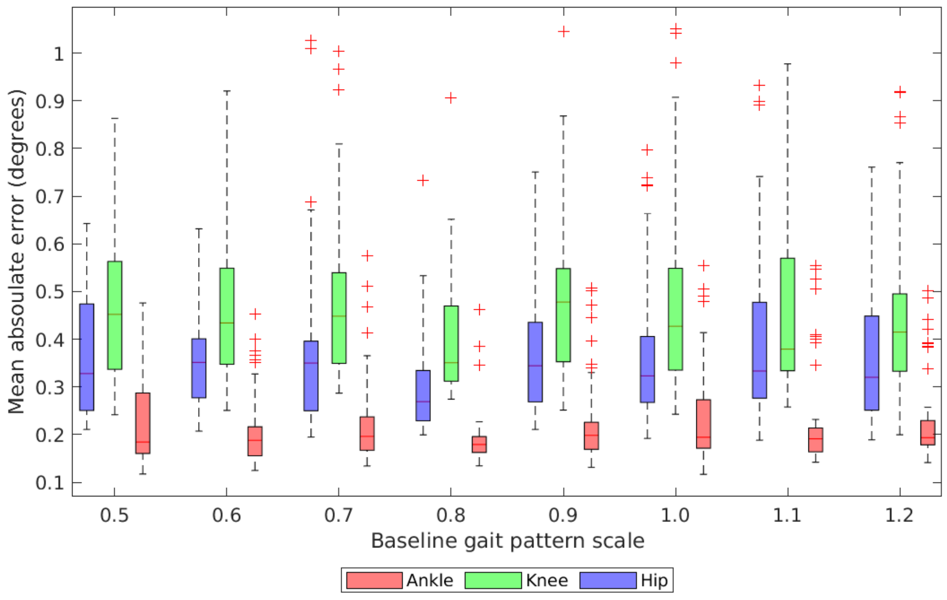

Figure 5.8: Stage 1 mean absolute error for the baseline gait pattern scales.

### Stage 2: Unseen desired gait patterns

Through training on the full set of scaled desired gait patterns from Desired-1, the DRL agent should be able to adapt to unseen desired gait patterns. In Stage 2, joint angle patterns from the unseen testing set, Desired-2 [123], are used as the desired gait patterns. In this test, the mean gait patterns for 5 subjects in a similar age range to the subjects in [120] are tested, each at a different gait speed. The error statistics for these Stage 2 tests are shown in Table 5.6.

| Joint | Mean absolute error (degrees) | Standard deviation (degrees) |
|---|---|---|
| Hip | 0.44 | 0.12 |
| Knee | 0.53 | 0.15 |
| Ankle | 0.37 | 0.14 |

Table 5.6: Gait pattern error for the Stage 2 tests with the unseen desired gait patterns from Desired-2.

As expected, the overall mean error was slightly higher than in Stage 1 with the set of desired gait patterns used in training. However, the error was still low for all joints for these unseen gait patterns, with an average error ranging from 0.37° for the ankle to 0.53° for the knee. This provides evidence for the robustness of this controller for tracking additional desired gait patterns a physiotherapist may wish the exoskeleton and subject to perform in a clinical session. Joint angle plots of the agent tested with the unseen desired gait patterns are shown in Figure 5.9, with various desired gait pattern scales displayed for each speed. From these figures it can be seen that the exoskeleton also closely followed these unseen desired gait patterns, and reached the expected range of motion for all gait patterns and speeds. In Figure 5.10 a boxplot of the error for the hip, knee, and ankle over the full test episodes in Stage 2 is shown.

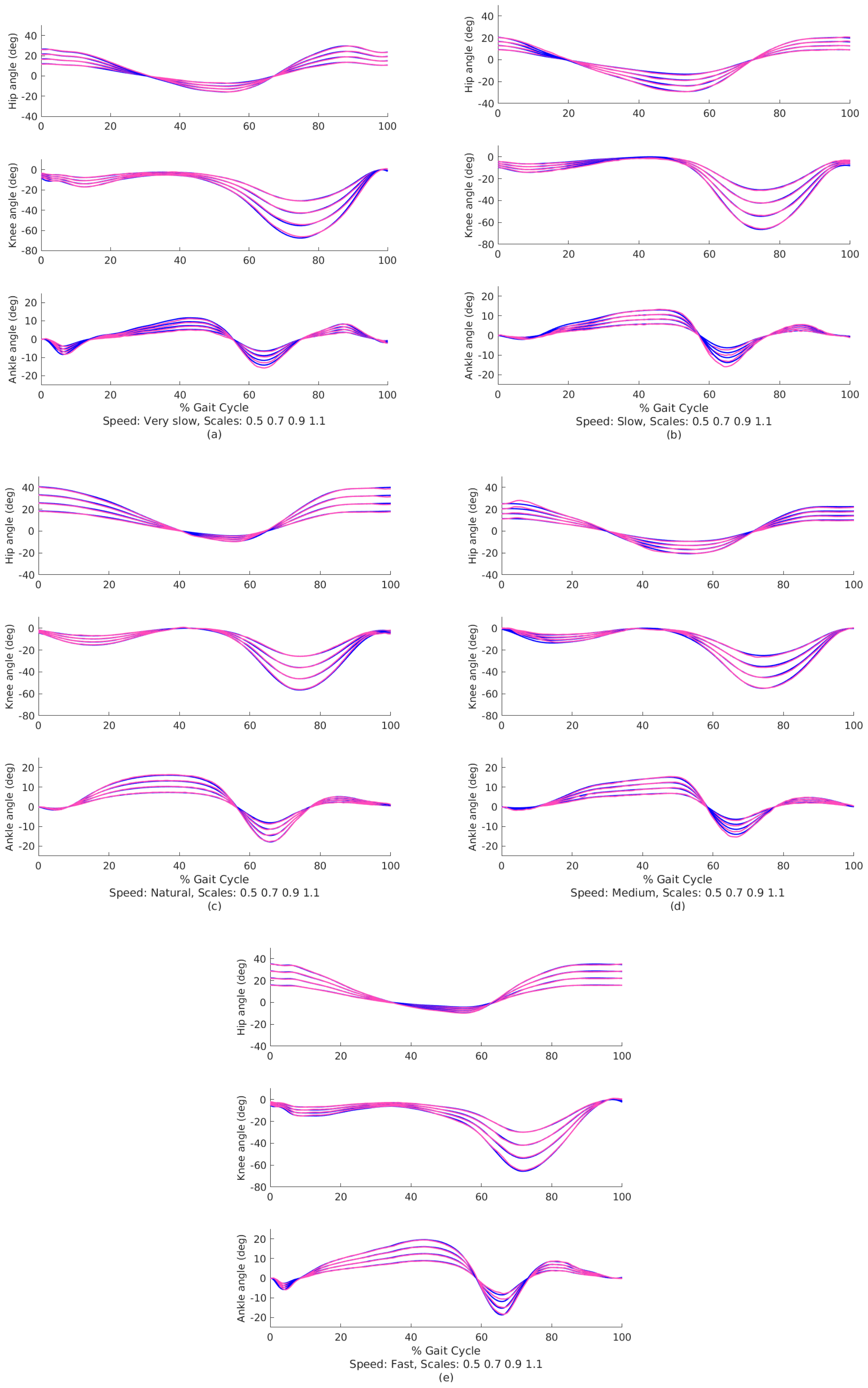

Figure 5.9: Stage 2 tests, with the unseen desired gait patterns from Desired-2 (blue), and the trained exoskeleton gait patterns (purple) for each speed, with scale factors of 0.5-1.1 shown.

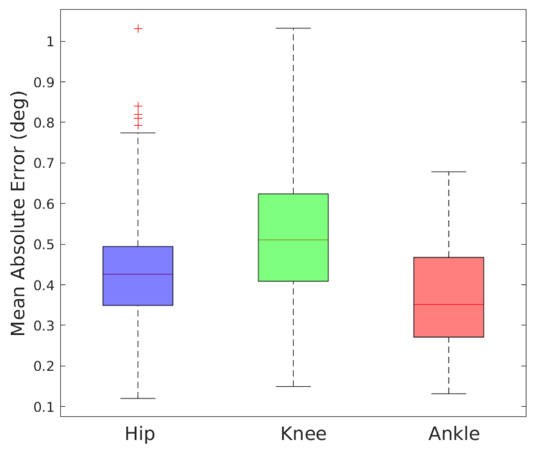

Figure 5.10: Stage 2 mean absolute error for each joint of the unseen gait pattern tests.

The accuracy found through testing on both the gait patterns seen in training in Stage 1, and unseen desired gait patterns in Stage 2, was greater than found in Chapter 3 with a position controller in the Gazebo simulation, while not requiring specific tuning of control parameters. The error found is also comparable to other existing adaptive model-based and learning-based exoskeleton control methods. As previously discussed, it is similar to and lower than the error found in [68] and [71], where the error ranged from 1° to 5°. A lower error was also found than in [113], where a PID controller was used to control an orthotic attached to an OpenSim model, and an average error of 3.8° was found. In [82] a similar mean absolute error of less than 1° was found for an RL-based controller, which also outperformed a tested PID controller.

To further improve the accuracy of controlling the unseen desired gait patterns in Stage 2, however, additional noise and alterations to the desired gait patterns seen in training could be included in order to provide more robust training data. Additional tuning of hyperparameters before or during training, such as the standard deviation of the Gaussian noise or the learning rate, could potentially further reduce the error in both testing stages as well. Retraining beginning with the weights of the current trained agent could also be conducted on a potential set of new gait patterns, for fine tuning prior to their use on an exoskeleton.

### Assist-as-Needed Control

To validate the assist-as-needed ability of this controller, a test is performed where the baseline gait pattern closely matches the desired gait pattern. To do so, a baseline gait pattern based on the natural gait pattern speed seen in Figure 5.1 is first simulated. This baseline gait pattern, and progressively scaled down versions are then tested with the trained agent. This represents the cases where a post-stroke individual is able to reach the desired gait pattern with more of their own power, and also where their function is increasingly limited. Through the DRL scheme, the agent learns to output less torque in the former scenario, as not doing so would increase the error, and progressively output a higher torque as the baseline gait pattern deviates from the desired, where the subject would require greater assistance to reach the goal.

For this test, each baseline gait pattern scale is tested for 20 episodes, with the desired gait pattern remaining constant. A plot of the average joint torque versus the baseline scale from these tests are shown in Figure 5.11, with a line fitted through linear regression, the output of which is seen in Table 5.7. The linear regression results in Table 5.7 indicate an satisfactory goodness-of-fit for the line. It is evident that as the baseline gait pattern scale increased and approached the desired gait pattern, there was a general downward trend in the average joint torque output, where less assistance was therefore provided to the user. This indicates that the DRL controller will avoid overexerting the user if they are able to reach most of the desired gait pattern on their own, but can also provide additional assistance if needed.

| Joint | x | $R^2$ | $p$-value |
|---|---|---|---|
| Hip | -13.33 | 0.839 | $< 0.005$ |
| Knee | -6.4 | 0.748 | $< 0.005$ |
| Ankle | -8.5 | 0.991 | $< 0.005$ |

Table 5.7: Linear regression results for the baseline gait pattern scale vs. mean joint torque, where x is the slope of the fitted line.

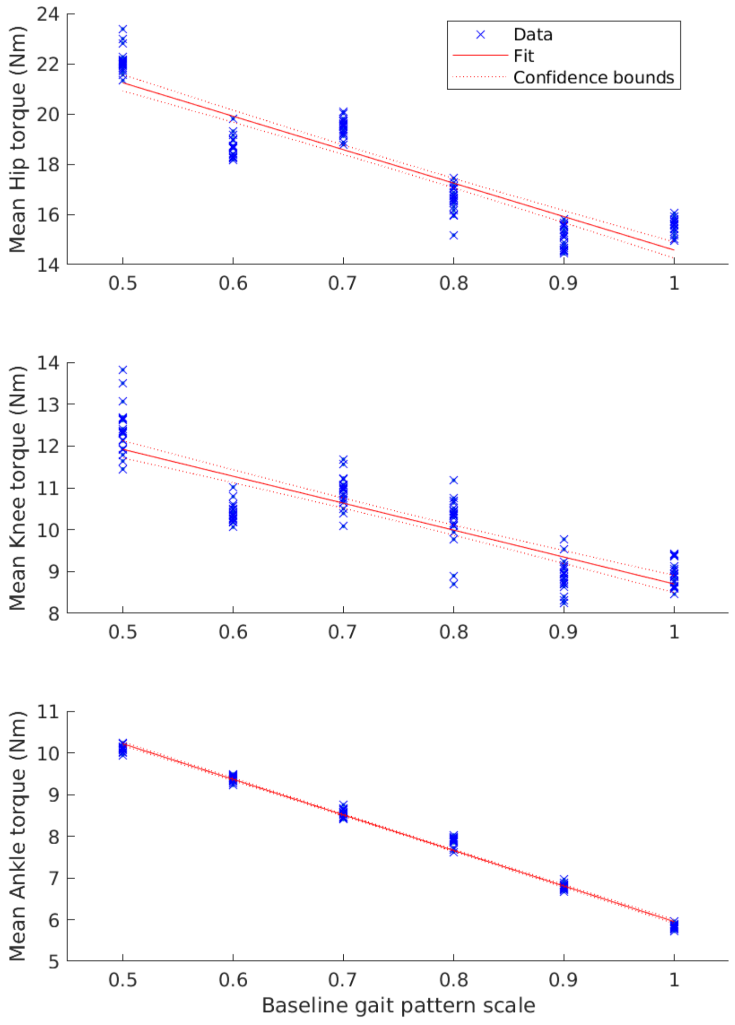

Figure 5.11: The baseline gait pattern scale vs. the average actuator torque for each episode tested, calculated over a moving average every 10 episode steps.

## 5.4 Chapter Summary

In this chapter, a model-free, generalized exoskeleton controller that is robust for multiple post-stroke individuals and a range of desired gait patterns was presented. This was accomplished through an extension of the model-free, end-to-end DRL controller presented in Chapter 4, where datasets of post-stroke individuals and healthy users were included in the training as the baseline and desired gait patterns. Alterations to the DRL framework including the exploration noise, reward function, and network design were also discussed. Experiments in simulation using this approach were conducted, and the results of testing in Stage 1, with the desired gait patterns used in training, and Stage 2, with an unseen set of desired gait patterns were presented. It was found that this approach provided low overall tracking error on both sets of desired gait patterns, and was also robust to varying post-stroke individuals' baseline gait patterns. Additionally, a test was conducted to validate the assist-as-needed ability of this controller, where higher average exoskeleton torque outputs were found for baseline gait patterns that deviated from the desired gait pattern, and therefore greater assistance as needed was provided.

# Chapter 6

# Conclusions

## 6.1 Summary of Contributions

### 6.1.1 Exoskeleton Simulation and Control Framework

To allow for simulation and control of an exoskeleton through a centralized software architecture, a ROS and Gazebo based framework was developed. The proposed architecture allowed for the simulation and control of an exoskeleton to be executed from a central location, where a desired gait pattern only has to be adjusted once to both verify the gait pattern in simulation and then deploy it on an exoskeleton. This framework used a series of ROS nodes to accomplish this. A joint publisher interface node contained the desired gait pattern and also initialized the ROS topics for both the simulation and the exoskeleton controller. The simulation platform was based in the Gazebo simulator and consisted of a simplified exoskeleton model with characteristics similar to the H2 exoskeleton. This was controlled using the ROS-control package to initialize a PID-based position controller for the exoskeleton model. A communication node was also established, to allow the ROS framework to communicate with the H2's on-board controller through a CAN bus network. Experiments were performed to validate this approach, where first a gait pattern obtained from OpenSim was simulated and tested, then changed on-the-fly to a physiotherapist recommended gait pattern. It was found that both the simulated and actual exoskeleton tracked these gait patterns well. However, a higher fidelity simulation including a musculoskeletal model of a human user, and an adaptive controller, would improve on the control accuracy.

### 6.1.2 End-to-end Deep Reinforcement Learning for Exoskeleton Control

To provide subject-specific and adaptive control of an exoskeleton, a novel model-free deep reinforcement learning based control method was proposed, where a predefined dynamics model was not required. This DRL approach used the Deep Deterministic Policy Gradient (DDPG) algorithm to learn the torque actions required to control a desired gait pattern on an exoskeleton in the presence of a user's baseline gait pattern, the user-exoskeleton interaction, and potential perturbations. This was accomplished by using a 3D exoskeleton model paired to a musculoskeletal model in the OpenSim-RL simulation platform. In this DRL scheme the musculoskeletal model's joint angles, velocities, and accelerations were included as observations to the agent, in addition to the exoskeleton actuators' torque and speed values, and the joint angle goals. The joint angle goals at each timestep of the simulation consisted of the hip, knee, and ankle joint angles of the desired gait pattern, which over a full episode represents one gait cycle. Initial experiments validated this DRL approach for exoskeleton control.

A generalized DRL exoskeleton controller was then also proposed, to allow for robust control with multiple post-stroke individuals and a large range of desired gait patterns. In this framework, clinical datasets of post-stroke individuals and healthy users were included in training as the baseline and desired gait patterns, respectively. Experiments with the desired gait patterns used in training, as well as with an unseen set of desired gait patterns, were conducted to validate the approach. The tests from both sets of desired gait patterns provided results with high overall tracking accuracy in controlling the gait patterns, while robust to the various subjects' baseline gait patterns. The assist-as-needed ability of this controller was also validated, where higher exoskeleton actuator torque was provided as the user's baseline gait pattern deviated from the desired.

## 6.2 Future Work

To add to the functionality of the ROS-based simulation architecture, a graphical user interface to allow for rapid customization of gait patterns would be a useful addition. A physiotherapist could use this to adjust a gait pattern more easily on-the-fly in a physiotherapy session, and validate this in the Gazebo simulation before being deployed on an exoskeleton. Furthermore, a higher fidelity simulation that incorporates a musculoskeletal model, as demonstrated in Chapters 4 and 5, in addition to further tuning of the controller, would improve upon the accuracy of the simulation and hardware control. Clinical tests of this architecture with post-stroke individuals would also further validate the proposed framework.

For the proposed DRL controller, additional adjustments could be made to optimize the network architecture, in order to increase sample efficiency and potentially reduce training time. Additionally, the parameters of the exploration functions could be further tuned, such as decreasing the Gaussian noise's standard deviation as training progresses. Reducing the learning rate in a similar fashion could also aid in fine tuning.

In addition to the improvements to the DRL scheme, it would be advantageous to then test this developed controller on exoskeleton hardware. Training would first be conducted in the simulation environment, with the trained model then deployed on the exoskeleton hardware. This could be accomplished through using the developed exoskeleton communication architecture presented in Chapter 3, as well as the integration of the DRL platform with ROS. Through this framework, joint torque actions could be deployed on an exoskeleton through the use of a torque control mode, and observations would be gathered from the various sensors on the exoskeleton such as potentiometers, Hall effect sensors, and strain gauges, as found on the H2 exoskeleton. In addition, human subject joint angles could also be tracked through a motion capture system or wearable sensors such as flexible goniometers.

In order to further adapt this control scheme for use on hardware, sim-to-real techniques would likely be required to account for the differences in the simulated and physical actuators, exoskeleton linkages, and human model. Current literature in this subject suggests including parameter, measurement, and physical characteristic noise into the model to provide robustness in training, in addition to the existing action space noise [50], where this can allow for an easier transfer to hardware. Additionally, add-ons to the OpenSim platform have been developed such as actuator classes that emulate the characteristics of DC motors [124], which would further assist in adding additional realism to the DRL simulation. Additional musculoskeletal model characteristics could also be included such as more anatomically complex muscle and ligament models, as well as further joint and muscle mechanic models, including muscle spasticity characteristics that are often found in post-stroke individuals [8]. The baseline gait patterns could then also be simulated through the use of these muscle models.

## 6.3 Final Concluding Statement

In this thesis, a ROS and Gazebo based simulation and control framework for lower-body exoskeletons, for gait-rehabilitation of post-stroke individuals was first presented. This provides a centralized software structure to both simulate and control an exoskeleton gait pattern, and where a desired gait pattern only has to be adjusted once to control both the simulation and the physical device. This is a useful tool for physiotherapists as it allows for validation of desired gait patterns in simulation, as well as on-the-fly adjustments to a gait pattern in clinical physiotherapy sessions. Experiments with varying gait patterns validated this approach.

A novel end-to-end, model-free deep reinforcement learning controller for exoskeletons was then presented. This control scheme is able to provide subject-specific and adaptive control in the presence of a user's existing gait pattern, their perturbations, and their interaction with the exoskeleton. This is accomplished in a model-free way, where a predefined dynamics model of the exoskeleton and user is not required, and instead learns directly from the interaction with the environment. This technique was validated through a number of experiments in simulation using the OpenSim-RL platform, where clinical data of baseline gait patterns from post-stroke individuals and desired gait patterns from healthy individuals were included in training. High tracking accuracy was found in these experiments with desired gait patterns both seen and unseen in training, while robust to varying post-stroke individuals' baseline gait patterns and level of function. Additional experiments also validated the assist-as-needed nature of this DRL controller.

# Bibliography

[1] H. Krueger, J. Koot, R. E. Hall, C. O'Callaghan, M. Bayley, and D. Corbett, "Prevalence of Individuals Experiencing the Effects of Stroke in Canada: Trends and Projections," *Stroke; a journal of cerebral circulation*, vol. 46, pp. 2226–2231, aug 2015.

[2] S. A. Murray, K. H. Ha, C. Hartigan, and M. Goldfarb, "An assistive control approach for a lower-limb exoskeleton to facilitate recovery of walking following stroke," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 3, pp. 441–449, 2015.

[3] R. B. Huitema, A. L. Hof, T. Mulder, W. H. Brouwer, R. Dekker, and K. Postema, "Functional recovery of gait and joint kinematics after right hemispheric stroke," *Archives of Physical Medicine and Rehabilitation*, vol. 85, no. 12, pp. 1982–1988, 2004.

[4] J. Boudarham, N. Roche, D. Pradon, C. Bonnyaud, D. Bensmail, and R. Zory, "Variations in Kinematics during Clinical Gait Analysis in Stroke Patients," *PLoS ONE*, vol. 8, p. e66421, jun 2013.

[5] G. Wu, C. Wang, X. Wu, Z. Wang, Y. Ma, and T. Zhang, "Gait Phase Prediction for Lower Limb Exoskeleton Robots," in *2016 IEEE International Conference on Information and Automation*, pp. 19–24, IEEE, aug 2016.

[6] N. Mijailovic, M. Gavrilovic, S. Rafajlovic, M. uric-Jovicic, and D. Popovic, "Gait phases recognition from accelerations and ground reaction forces: Application of neural networks," *Telfor Journal*, vol. 1, no. 1, pp. 34–36, 2009.

[7] Y. Wang, M. Mukaino, K. Ohtsuka, Y. Otaka, H. Tanikawa, F. Matsuda, K. Tsuchiyama, J. Yamada, and E. Saitoh, "Gait characteristics of post-stroke hemiparetic patients with different walking speeds," *International Journal of Rehabilitation Research*, vol. 43, pp. 69–75, mar 2020.

[8] S. Li, G. E. Francisco, and P. Zhou, "Post-stroke hemiplegic gait: New perspective and insights," *Frontiers in Physiology*, vol. 9, p. 1021, aug 2018.

[9] E. J. Roth, C. Merbitz, K. Mroczek, S. A. Dugan, and W. W. Suh, "Hemiplegic Gait," *American Journal of Physical Medicine & Rehabilitation*, vol. 76, pp. 128–133, mar 1997.

[10] G. Chen, C. Patten, D. H. Kothari, and F. E. Zajac, "Gait deviations associated with post-stroke hemiparesis: Improvement during treadmill walking using weight support, speed, support stiffness, and handrail hold," *Gait and Posture*, vol. 22, pp. 57–62, aug 2005.

[11] K. K. Patterson, W. H. Gage, D. Brooks, S. E. Black, and W. E. McIlroy, "Changes in gait symmetry and velocity after stroke: A cross-sectional study from weeks to years after stroke," *Neurorehabilitation and Neural Repair*, vol. 24, no. 9, pp. 783–790, 2010.

[12] S. J. Olney and C. Richards, "Hemiparetic gait following stroke. Part I: Characteristics," *Gait and Posture*, vol. 4, pp. 136–148, apr 1996.

[13] C. Selves, G. Stoquart, and T. Lejeune, "Gait rehabilitation after stroke: review of the evidence of predictors, clinical outcomes and timing for interventions," *Acta Neurologica Belgica*, vol. 120, no. 4, pp. 783–790, 2020.

[14] D. R. Louie and J. J. Eng, “Powered robotic exoskeletons in post-stroke rehabilitation of gait: A scoping review,” *Journal of NeuroEngineering and Rehabilitation*, vol. 13, p. 53, dec 2016.

[15] K. P. Westlake and C. Patten, “Pilot study of Lokomat versus manual-assisted treadmill training for locomotor recovery post-stroke,” *Journal of NeuroEngineering and Rehabilitation*, vol. 6, pp. 1–11, jun 2009.

[16] M. Bortole, J. C. Moreno, G. E. Francisco, J. L. Pons, A. Venkatakrishnan, J. L. Contreras-Vidal, F. Zhu, A. Venkatakrishnan, F. Zhu, J. C. Moreno, G. E. Francisco, J. L. Pons, and J. L. Contreras-Vidal, “The H2 robotic exoskeleton for gait rehabilitation after stroke: early findings from a clinical study,” *Journal of NeuroEngineering and Rehabilitation*, vol. 12, p. 54, dec 2015.

[17] B. Hobbs and P. Artemiadis, “A Review of Robot-Assisted Lower-Limb Stroke Therapy: Unexplored Paths and Future Directions in Gait Rehabilitation,” *Frontiers in Neurorobotics*, vol. 14, no. April, 2020.

[18] S. Freivogel, D. Schmalohr, and J. Mehrholz, “Improved walking ability and reduced therapeutic stress with an electromechanical gait device,” *Journal of Rehabilitation Medicine*, vol. 41, pp. 734–739, sep 2009.

[19] M. Pazzaglia and M. Molinari, “The embodiment of assistive devices-from wheelchair to exoskeleton,” *Physics of Life Reviews*, vol. 16, pp. 163–175, 2016.

[20] E. López-Larraz, F. Trincado-Alonso, V. Rajasekaran, S. Pérez-Nombela, A. J. Del-Ama, J. Aranda, J. Minguez, A. Gil-Agudo, and L. Montesano, “Control of an ambulatory exoskeleton with a brain-machine interface for spinal cord injury gait rehabilitation,” *Frontiers in Neuroscience*, vol. 10, no. AUG, 2016.

[21] J. L. Contreras-Vidal, N. A. Bhagat, J. Brantley, J. G. Cruz-Garza, Y. He, Q. Manley, S. Nakagome, K. Nathan, S. H. Tan, F. Zhu, and J. L. Pons, “Powered exoskele-

tons for bipedal locomotion after spinal cord injury," *Journal of Neural Engineering*, vol. 13, no. 3, 2016.

[22] B. S. Rupal, S. Rafique, A. Singla, E. Singla, M. Isaksson, and G. S. Virk, "Lower-limb exoskeletons: Research trends and regulatory guidelines in medical and non-medical applications," *International Journal of Advanced Robotic Systems*, vol. 14, pp. 1–27, nov 2017.

[23] K. Lo, M. Stephenson, and C. Lockwood, "Effectiveness of robotic assisted rehabilitation for mobility and functional ability in adult stroke patients: a systematic review," *JBI Database of Systematic Reviews and Implementation Reports*, vol. 15, no. 12, pp. 3049–3091, 2017.

[24] S. Federici, F. Meloni, M. Bracalenti, and M. L. De Filippis, "The effectiveness of powered, active lower limb exoskeletons in neurorehabilitation: A systematic review," *NeuroRehabilitation*, vol. 37, no. 3, pp. 321–340, 2015.

[25] Y.-H. Kim, "Robotic assisted rehabilitation therapy for enhancing gait and motor function after stroke," *Precision and Future Medicine*, vol. 3, pp. 103–115, sep 2019.

[26] T. G. Hornby, D. S. Straube, C. R. Kinnaird, C. L. Holleran, A. J. Echauz, K. S. Rodriguez, E. J. Wagner, E. A. Narducci, and G. Hornby, "Importance of Specificity, Amount, and Intensity of Locomotor Training to Improve Ambulatory Function in Patients Poststroke," *Topics in Stroke Rehabilitation*, vol. 18, no. 4, pp. 293–307, 2011.

[27] B. E. Fisher and K. J. Sullivan, "Activity-dependent factors affecting poststroke functional outcomes," *Topics in Stroke Rehabilitation*, vol. 8, no. 3, pp. 31–44, 2001.

[28] A. J. Young and D. P. Ferris, "State of the Art and Future Directions for Lower Limb Robotic Exoskeletons," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, pp. 171–182, feb 2017.

[29] J. A. Blaya and H. Herr, "Adaptive Control of a Variable-Impedance Ankle-Foot Orthosis to Assist Drop-Foot Gait," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 12, pp. 24–31, mar 2004.

[30] M. Yamamoto, K. Shimatani, M. Hasegawa, and Y. Kurita, "Effect of an ankle–foot orthosis on gait kinematics and kinetics: case study of post-stroke gait using a musculoskeletal model and an orthosis model," *ROBOMECH Journal*, vol. 6, no. 1, pp. 0–6, 2019.

[31] T. S.L., "Exo h2." www.technaid.com/products/robotic-exoskeleton-exo-exoesqueleto. Accessed: 2019.

[32] X. Zhang, H. Wang, Y. Tian, L. Peyrodie, and X. Wang, "Model-free based neural network control with time-delay estimation for lower extremity exoskeleton," *Neurocomputing*, vol. 272, pp. 178–188, jan 2018.

[33] Z. Aftab and A. Ali, "Simulating a wearable lower-body exoskeleton device for torque and power estimation," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 412–417, 2017.

[34] D. G. Santos and S. Carlos, "ADAMS / Matlab Co-simulation of an Exoskeleton for Lower Limbs," in *20th International Congress of Mechanical Engineering (CoBEM 2009)*, 2009.

[35] F. Ferrati, R. Bortoletto, and E. Pagello, "Virtual modelling of a real exoskeleton constrained to a human musculoskeletal model," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8064 LNAI, pp. 96–107, 2013.

[36] H. J. Asl, T. Narikiyo, and M. Kawanishi, "Neural network velocity field control of robotic exoskeletons with bounded input," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pp. 1363–1368, IEEE, jul 2017.

[37] B. Chen, H. Ma, L. Y. Qin, F. Gao, K. M. Chan, S. W. Law, L. Qin, and W. H. Liao, "Recent developments and challenges of lower extremity exoskeletons," *Journal of Orthopaedic Translation*, vol. 5, pp. 26–37, 2016.

[38] J. Hong, C. Chun, S.-J. Kim, and F. C. Park, "Gaussian Process Trajectory Learning and Synthesis of Individualized Gait Motions," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, pp. 1236–1245, jun 2019.

[39] G. Lv, H. Zhu, and R. D. Gregg, "On the design and control of highly backdrivable lower-limb exoskeletons: A discussion of past and ongoing work," *IEEE Control Systems*, vol. 38, pp. 88–113, dec 2018.

[40] A. McDaid, K. Kora, S. Xie, J. Lutz, and M. Battley, "Human-inspired robotic exoskeleton (HuREx) for lower limb rehabilitation," in *2013 IEEE International Conference on Mechatronics and Automation, IEEE ICMA 2013*, pp. 19–24, IEEE Computer Society, 2013.

[41] R. Mendoza-Crespo, D. Torricelli, J. C. Huegel, J. L. Gordillo, J. L. Pons, and R. Soto, "An Adaptable Human-Like Gait Pattern Generator Derived From a Lower Limb Exoskeleton," *Frontiers in Robotics and AI*, vol. 6, p. 36, may 2019.

[42] T. P. Luu, K. Low, X. Qu, H. Lim, and K. Hoon, "An individual-specific gait pattern prediction model based on generalized regression neural networks," *Gait & Posture*, vol. 39, pp. 443–448, jan 2014.

[43] F. Horst, S. Lapuschkin, W. Samek, K.-R. Müller, and W. I. Schöllhorn, "Explaining the unique nature of individual gait patterns with deep learning," *Scientific Reports*, vol. 9, p. 2391, dec 2019.

[44] H. B. Lim, Trieu Phat Luu, K. H. Hoon, and K. H. Low, "Natural gait parameters prediction for gait rehabilitation via artificial neural network," in *2010 IEEE/RSJ*

*International Conference on Intelligent Robots and Systems*, pp. 5398–5403, IEEE, oct 2010.

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

[46] J. C. Moreno, J. Figueiredo, and J. L. Pons, "Chapter 7 - Exoskeletons for lower-limb rehabilitation," *Rehabilitation Robotics*, pp. 89–99, 2018.

[47] T. Yan, M. Cempini, M. Oddo, and N. Vitiello, "Review of assistive strategies in powered lower-limb orthoses and exoskeletons," *Robotics and Autonomous Systems*, vol. 64, pp. 120–136, 2015.

[48] Y. Long, Z. J. Du, W. D. Wang, and W. Dong, "Robust Sliding Mode Control Based on GA Optimization and CMAC Compensation for Lower Limb Exoskeleton," *Applied Bionics and Biomechanics*, vol. 2016, 2016.

[49] X. Wang, X. Li, J. Wang, X. Fang, and X. Zhu, "Data-driven model-free adaptive sliding mode control for the multi degree-of-freedom robotic exoskeleton," *Information Sciences*, vol. 327, pp. 246–257, jan 2016.

[50] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[51] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," *ArXiv1709.10087*, 2018.

[52] Z. Qu, W. Wei, W. Wang, S. Zha, T. Li, J. Gu, and C. Yue, "Research on Fuzzy Adaptive Impedance Control of Lower Extremity Exoskeleton," in *Proc. of 2019 IEEE International Conference on Mechatronics and Automation*, pp. 939–944, Institute of Electrical and Electronics Engineers Inc., aug 2019.

[53] B. N. Fournier, E. D. Lemaire, A. J. Smith, and M. Doumit, "Modeling and Simulation of a Lower Extremity Powered Exoskeleton," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 8, pp. 1596–1603, 2018.

[54] C. Copilusi, M. Ceccarelli, G. Carbone, and A. Margine, "Mechanism of a leg exoskeleton for walking rehabilitation purposes," in *Mechanisms and Machine Science*, vol. 17, pp. 107–114, Springer, Dordrecht, 2014.

[55] S. B. Winder and J. M. Esposito, "Modeling and control of an upper-body exoskeleton," in *Proceedings of the Annual Southeastern Symposium on System Theory*, pp. 263–268, IEEE, mar 2008.

[56] B. K. Dinh, L. Cappello, and L. Masia, "Localized Extreme Learning Machine for online inverse dynamic model estimation in soft wearable exoskeleton," in *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, vol. 2016-July, pp. 580–587, IEEE, jun 2016.

[57] E. Ceseracciu, A. Mantoan, M. Bassa, J. C. Moreno, J. L. Pons, G. A. Prieto, A. J. Del Ama, E. Marquez-Sanchez, A. Gil-Agudo, C. Pizzolato, D. G. Lloyd, and M. Reggiani, "A flexible architecture to enhance wearable robots: Integration of EMG-informed models," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 4368–4374, 2015.

[58] "Quarc win64 target." http://quanser-update.azurewebsites.net/quarc/documentation/quarc_win64_target.html. Accessed: 2019.

[59] "Simulink and ros interaction." www.mathworks.com/help/ros/ug/simulink-and-ros-interaction.html. Accessed: 2019.

[60] D. Torricelli, C. Cortés, N. Lete, Á. Bertelsen, J. E. Gonzalez-Vargas, A. J. Del-Ama, I. Dimbwadyo, J. C. Moreno, J. Florez, and J. L. Pons, "A Subject-Specific Kinematic Model to Predict Human Motion in Exoskeleton-Assisted Gait.," *Frontiers in neurorobotics*, vol. 12, p. 18, 2018.

[61] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ROS and Gazebo," in *2016 20th International Conference on System Theory, Control and Computing, ICSTCC 2016 - Joint Conference of SINTES 20, SACCS 16, SIMSIS 20 - Proceedings*, pp. 96–101, IEEE, oct 2016.

[62] W. Qian, Z. Xia, J. Xiong, Y. Gan, Y. Guo, S. Weng, H. Deng, Y. Hu, and J. Zhang, "Manipulation task simulation using ROS and Gazebo," in *2014 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2014*, pp. 2594–2598, IEEE, dec 2014.

[63] F. Sado, H. J. Yap, R. Ariffin, R. A. R. Ghazilla, and N. Ahmad, "Exoskeleton robot control for synchronous walking assistance in repetitive manual handling works based on dual unscented Kalman filter," *PLoS ONE*, vol. 13, no. 7, 2018.

[64] D. Sanz-Merodio, M. Cestari, J. Arevalo, X. Carrillo, and E. Garcia, "Generation and control of adaptive gaits in lower-limb exoskeletons for motion assistance," *Advanced Robotics*, vol. 28, pp. 329–338, mar 2014.

[65] S. K. Banala, S. K. Agrawal, S. H. Kim, and J. P. Scholz, "Novel gait adaptation and neuromotor training results using an active leg exoskeleton," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 2, pp. 216–225, 2010.

[66] S. Maggioni, N. Reinert, L. Lünenburger, and A. Melendez-Calderon, "An Adaptive and Hybrid End-Point/Joint Impedance Controller for Lower Limb Exoskeletons," *Frontiers in Robotics and AI*, vol. 5, p. 104, oct 2018.

[67] L. Wang, E. H. Van Asseldonk, and H. Van Der Kooij, "Model predictive control-based gait pattern generation for wearable exoskeletons," *IEEE International Conference on Rehabilitation Robotics*, pp. 1–6, 2011.

[68] D. L. Castro, C. H. Zhong, F. Braghin, and W. H. Liao, "Lower Limb Exoskeleton Control via Linear Quadratic Regulator and Disturbance Observer," in *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*, pp. 1743–1748, Institute of Electrical and Electronics Engineers Inc., jul 2018.

[69] O. Harib, A. Hereid, A. Agrawal, T. Gurriet, S. Finet, G. Boeris, A. Duburcq, M. E. Mungai, M. Masselin, A. D. Ames, K. Sreenath, and J. Grizzle, "Feedback Control of an Exoskeleton for Paraplegics: Toward Robustly Stable Hands-free Dynamic Walking," *ArXiv1802.08322*, feb 2018.

[70] R. Huang, H. Cheng, J. Qiu, and J. Zhang, "Learning Physical Human-Robot Interaction With Coupled Cooperative Primitives for a Lower Exoskeleton," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1–9, 2019.

[71] G. Bingjing, H. Jianhai, L. Xiangpan, and Y. Lin, "Human–robot interactive control based on reinforcement learning for gait rehabilitation training robot," *International Journal of Advanced Robotic Systems*, vol. 16, mar 2019.

[72] Y. Zhang, S. Li, K. J. Nolan, and D. Zanotto, "Adaptive Assist-as-needed Control Based on Actor-Critic Reinforcement Learning," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 4066–4071, Institute of Electrical and Electronics Engineers Inc., nov 2019.

[73] M. Hamaya, T. Matsubara, T. Noda, T. Teramae, and J. Morimoto, "Learning assistive strategies from a few user-robot interactions: Model-based reinforcement learning approach," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3346–3351, 2016.

[74] S. G. Khan, M. Tufail, S. H. Shah, and I. Ullah, "Reinforcement learning based compliance control of a robotic walk assist device," *Advanced Robotics*, pp. 1–12, nov 2019.

[75] S. Lee, M. Park, K. Lee, and J. Lee, "Scalable muscle-actuated human simulation and control," *ACM Transactions on Graphics*, vol. 38, no. 4, 2019.

[76] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[77] A. S. Anand, G. Zhao, H. Roth, and A. Seyfarth, "A deep reinforcement learning based approach towards generating human walking behavior with a neuromuscular model," *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pp. 537–543, 2020.

[78] C. R. Gil, H. Calvo, and H. Sossa, "Learning an efficient gait cycle of a biped robot based on reinforcement learning and artificial neural networks," *Applied Sciences (Switzerland)*, vol. 9, feb 2019.

[79] J. García and D. Shafie, "Teaching a humanoid robot to walk faster through Safe Reinforcement Learning," *Engineering Applications of Artificial Intelligence*, vol. 88, feb 2020.

[80] C. Liu, A. Lonsberry, M. Nandor, M. Audu, A. Lonsberry, and R. Quinn, "Implementation of Deep Deterministic Policy Gradients for Controlling Dynamic Bipedal Walking," *Biomimetics*, vol. 4, no. 1, p. 28, 2019.

[81] V. C. V. Kumar, S. Ha, G. Sawicki, and C. K. Liu, "Learning a Control Policy for Fall Prevention on an Assistive Walking Device," *ArXiv1909.10488*, 2019.

[82] D. Di Febbo, E. Ambrosini, M. Pirotta, E. Rojas, M. Restelli, A. L. Pedrocchi, and S. Ferrante, "Does Reinforcement Learning outperform PID in the control of FES-induced elbow flex-extension?," *2018 IEEE International Symposium on Medical Measurements and Applications, Proceedings*, pp. 1–6, 2018.

[83] M. Lyu, W. H. Chen, X. Ding, and J. Wang, "Knee exoskeleton enhanced with artificial intelligence to provide assistance-as-needed," *Review of Scientific Instruments*, vol. 90, no. 9, 2019.

[84] B. Brahmi, M. Saad, C. O. Luna, P. S. Archambault, and M. H. Rahman, "Passive and active rehabilitation control of human upper-limb exoskeleton robot with dynamic uncertainties," *Robotica*, vol. 36, pp. 1757–1779, nov 2018.

[85] P. Yang, J. Sun, J. Wang, G. Zhang, and Y. Zhang, "Model-free based back-stepping sliding mode control for wearable exoskeletons," in *25th IEEE International Conference on Automation and Computing*, Institute of Electrical and Electronics Engineers Inc., sep 2019.

[86] Z. Li, J. Liu, Z. Huang, Y. Peng, H. Pu, and L. Ding, "Adaptive Impedance Control of Human-Robot Cooperation Using Reinforcement Learning," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 10, pp. 8013–8022, 2017.

[87] H. van Hasselt, "Reinforcement learning in continuous state and action spaces," in *Adaptation, Learning, and Optimization*, vol. 12, pp. 207–251, 2012.

[88] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, sep 2016.

[89] A. Singh, L. Yang, C. Finn, and S. Levine, "End-To-End Robotic Reinforcement Learning without Reward Engineering," *ArXiv1904.07854*, 2019.

[90] "Gait 2392 and 2354 models." https://simtk-confluence.stanford.edu/display/OpenSim/Gait+2392+and+2354+Models. Accessed: 2019.

[91] C. T. John, A. Seth, M. H. Schwartz, and S. L. Delp, "Contributions of muscles to mediolateral ground reaction force over a range of walking speeds," *Journal of Biomechanics*, vol. 45, pp. 2438–2443, 2012.

[92] O. S. R. Foundation, "Ros control." http://gazebosim.org/tutorials/?tut=ros_control. Accessed: 2019.

[93] D. Coleman, "Gazebo ros demos." https://github.com/ros-simulation/gazebo_ros_demos. Accessed: 2019.

[94] M. M. Bassa, "Development of the communication system for a lower limb human exoskeleton using the ros middleware.," *Master's Thesis. University of Padua, Padua, Italy.*, 2015.

[95] "March iv exoskeleton." https://github.com/project-march/march-iv. Accessed: 2019.

[96] "Introduction to the controller area network (can)." http://www.ti.com/lit/an/sloa101b/sloa101b.pdf. Accessed: 2019.

[97] "Beagleboard,." https://beagleboard.org/black-wireless. Accessed: 2019.

[98] "Linux-can / socketcan user space applications." https://github.com/linux-can. Accessed: 2019.

[99] "Rs485 can cape." https://www.waveshare.com/wiki/RS485_CAN_CAPE. Accessed: 2019.

[100] M. Lüdtke and I. Wanders, "socketcan_bridge." http://wiki.ros.org/socketcan_bridge. Accessed: 2019.

[101] M. Plappert, "Keras-rl." https://github.com/keras-rl/keras-rl. Accessed: 2019.

[102] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *ArXiv1312.5602*, 2013.

[103] Ł. Kidziński, S. P. Mohanty, C. F. Ong, J. L. Hicks, S. F. Carroll, S. Levine, M. Salathé, and S. L. Delp, "Learning to Run Challenge: Synthesizing Physiologically Accurate Motion Using Deep Reinforcement Learning," in *Escalera S., Weimer M. (eds) The NIPS '17 Competition: Building Intelligent Systems. The Springer Series on Challenges in Machine Learning*, pp. 101–120, Springer, Cham, 2018.

[104] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the Brownian motion," *Physical Review*, vol. 36, pp. 823–841, sep 1930.

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

[106] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *ArXiv1606.01540*, 2016.

[107] E. P. Grabke, K. Masani, and J. Andrysek, "Lower Limb Assistive Device Design Optimization Using Musculoskeletal Modeling: A Review," *Journal of Medical Devices*, vol. 13, no. 4, 2019.

[108] M. Khamar, M. Edrisi, and M. Zahiri, "Human-exoskeleton control simulation, kinetic and kinematic modeling and parameters extraction," *MethodsX*, vol. 6, pp. 1838–1846, 2019.

[109] D. Coll Pujals, *Simulation of the assistance of an exoskeleton on lower limbs joints using Opensim.* Master's thesis, Polytechnic University of Catalonia, 2017.

[110] "Thelen 2003 muscle model." https://simtk-confluence.stanford.edu/display/OpenSim/Thelen+2003+Muscle+Model . Accessed: 2020.

[111] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: Multibody dynamics for biomedical research," *Procedia IUTAM*, vol. 2, pp. 241–261, 2011.

[112] S. Qiu, W. Guo, D. Caldwell, and F. Chen, "Exoskeleton Online Learning and Estimation of Human Walking Intention Based on Dynamical Movement Primitives," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, jan 2020.

[113] D. Mosconi, P. F. Nunes, and A. A. G. Siqueira, "Modeling and control of an active knee orthosis using a computational model of the musculoskeletal system," *Journal of Mechatronics Engineering*, vol. 1, no. 3, p. 12, 2018.

[114] L. Rose, M. C. F. Bazzocchi, C. de Souza, J. Vaughan-Graham, K. Patterson, and G. Nejat, "A Framework for Mapping and Controlling Exoskeleton Gait Patterns in both Simulation and Real World," in *Proc. of the 2020 Design of Medical Devices Conf.*, (Minneapolis, Minnesota), 2020.

[115] G. Bovi, M. Rabuffetti, P. Mazzoleni, and M. Ferrarin, "A multiple-task gait analysis approach: Kinematic, kinetic and EMG reference data for healthy young and adult subjects," *Gait and Posture*, vol. 33, no. 1, pp. 6–13, 2011.

[116] F. Horst, S. Lapuschkin, W. Samek, K. R. Müller, and W. I. Schöllhorn, "Explaining the unique nature of individual gait patterns with deep learning," *Scientific Reports*, vol. 9, no. 1, 2019.

[117] J. K. Moore, S. K. Hnat, and A. J. van den Bogert, "An elaborate data set on human gait and the effect of mechanical perturbations," *PeerJ*, vol. 2015, no. 3, 2015.

[118] N. D. Neckel, N. Blonien, D. Nichols, and J. Hidler, "Abnormal joint torque patterns exhibited by chronic stroke subjects while walking with a prescribed physiological gait pattern," *Journal of NeuroEngineering and Rehabilitation*, vol. 5, p. 19, dec 2008.

[119] W. Wang, J. Chen, Y. Ji, W. Jin, J. Liu, and J. Zhang, "Evaluation of lower leg muscle activities during human walking assisted by an ankle exoskeleton," *IEEE Transactions on Industrial Informatics*, vol. 3203, no. c, pp. 1–1, 2020.

[120] B. A. Knarr, T. M. Kesar, D. S. Reisman, S. A. Binder-Macleod, and J. S. Higginson, "Changes in the activation and function of the ankle plantar flexor muscles due to gait retraining in chronic stroke survivors.," *Journal of neuroengineering and rehabilitation*, vol. 10, p. 12, 2013.

[121] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *35th International Conference on Machine Learning, ICML 2018*, vol. 4, pp. 2587–2601, 2018.

[122] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[123] T. Lencioni, I. Carpinella, M. Rabuffetti, A. Marzegan, and M. Ferrarin, “Human kinematic, kinetic and EMG data during different walking and stair ascending and descending tasks,” *Scientific Data*, vol. 6, no. 1, pp. 1–10, 2019.

[124] V. Q. Nguyen, A. K. LaPre, M. A. Price, B. R. Umberger, and F. C. Sup, “Inclusion of actuator dynamics in simulations of assisted human movement,” *International Journal for Numerical Methods in Biomedical Engineering*, vol. 36, may 2020.