Subject-Specific Assistive Control of a Stroke Rehabilitation Robot

by

Jason Hunter

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The branch of medicine known as rehabilitation involves an injured patient employing repetitive practices under the supervision of a physiotherapist in an effort to help the patient recover motor control. But during stroke patient rehabilitation, therapists lack objective measures of the subject's movement performance unless sensors/markers are physically placed on the subject's body, which can often present an inconvenience and lack of tangible benefit to the subject. Without removing the need for physiotherapists, rehab robots can assist in improving the patient's performance during repetitive movements. The goal of this research is to focus on the symbiosis between the patient and robot such that the rehabilitation exercise is tailored to each specific patient's needs; subject-specific assistive control of rehab robotics evaluates the user's voluntary motion, from which robotic assistance is then tailored based on the user's ability or performance.

The end-effector-based robot used in this study is an actuated 2-degree of freedom (DOF) 4-linkage planar parallelogram manipulator. To measure the user's movement, an inverse dynamic model of their upper arm as a planar 2-DOF linkage was used to estimate the user's shoulder and elbow joint torques, using real-time kinematic data. To obtain this real-time kinematic data without physically placing sensors/markers on the user, a system of two equations (which defined the user's planar arm model in terms of their shoulder and elbow joint angles) was solved in real-time, assuming the user's shoulder joint centre didn't move in the horizontal plane. This joint angle estimation method was experimentally validated against the gold standard of a digital goniometer on a healthy subject, and further validated against pre-trained pose estimation models. The equation solver achieved a root mean squared error (RMSE) of 0.66 degrees with respect to 10 frames of goniometer measurements, and an RMSE of 0.84 degrees with respect to a pre-trained computer vision pose estimation model used on the same 10 pose instances.

To measure each specific user's positional and directional performance in following a desired trajectory, the robot was equipped with a force sensor on the end-effector. The directional performance was then used to implement a subject-specific adaptive control approach. The idea was that if the resultant force direction of the user and the desired resultant direction of the end-effector were within a small arbitrary threshold angle, then the level of assistive torque applied by the robot would decrease. This scheme would not only help promote human engagement but also maybe provide an indirect means of measuring the patient's motor control recovery because this would indicate the degree to which the patient was improving their motor control. Future experiments will be conducted on post-stroke patients at a local hospital, to evaluate the effectiveness of this control scheme.

Different machine learning control strategies were also explored to try to remove the requirement of finding an accurate physics-based model of the rehab robot, given that the input data (i.e., trajectory parameters) and output (i.e., robot joint torque) data of the multibody system used to represent its dynamics, could reliably be measured from previous trials by mounted sensors. The effectiveness of model-based control of a dynamic multibody system is reliant upon the accuracy of a model of the actual system. Feed-forward Computed-Torque control (CT), in particular, is only applicable when a perfect (or accurate enough) model of the dynamic system is available. Otherwise, as the mismatch between the model of the system and the real dynamics increases, an outer-loop feedback term (e.g., proportional-integral-derivative (PID) control) becomes increasingly crucial to correct for this increasing error. The three machine learning architectures that were implemented using MATLAB's neural network toolbox included a Deep Lagragian Network (DeLaN), a Feed-forward Neural Network (FNN), and a Recurrent Neural Network (RNN). None of these models produced promising results when deployed as part of a CT PID controller tested on the robot with no human using the robot.

An EMG-based human-robot interaction model between the rehab robot and a subject user was then introduced since it was hypothesized that a subject's muscle activations (bicep brachii, tricep brachii, and deltoid) measured via sEMGs could map to the elbow and shoulder joint torques exerted by the subject. A traditional way to convert normalized EMG signals into the appropriate muscle joint torques is to use a biomechanical muscle model, such as the three-component Hill model. However, including a general muscle model within a multibody model introduces various drawbacks including muscle redundancy, complex musculoskeletal geometry such as wrapping pathways, difficult-to-fit parameters for each muscle, and parameter sensitivity. One method to potentially alleviate the complex muscle geometry, redundancy, and interpretation of EMG signals within a control framework is a machine learning model (i.e., neural network) trained by experimental data. However, many neural network architecture configurations were not able to achieve acceptable results on the testing data trial when trained on the training data set. Experimental results showed that using muscle activations as an indicator for predicting the subject's joint torques to eventually replace the force sensor, was deemed not reliable.

Acknowledgements

I would like to thank my supervisor, Prof. John McPhee, for his support throughout my Master's degree. I would also like to thank my committee members for their advice and support in crafting this thesis. Finally, I would like to thank all members of the Motion Research Group for their help and advice. Being a part of this team was a unique experience.

Dedication

This thesis is dedicated to my family.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

New cases of stroke incidents have unfortunately become increasingly common occurrences, especially among the higher-age populations [39]. Research into rehabilitation robotics, however, has fortunately grown extensively and the number of these rehabilitation robots on the market has dramatically grown. Robotic rehabilitation therapy can deliver either high-dosage or high-intensity training, making it useful for patients with motor disorders caused by stroke. Robotic devices used for human motor rehabilitation include either end-effector or exoskeleton types; this thesis focuses on an end-effector-based robot.

1.1 Motivation and Thesis Goals

This thesis introduces a novel control scheme to enable more subject-specific adaptability and examines the role that machine learning can play in identifying a robot's unknown dynamic model for model-based control. In an effort to make the control scheme subject-specific and adaptive, the goal for this robot will be to adjust its joint torque contribution in accordance with the user's performance. Performance metrics will include the deviations from the desired trajectory to measure the user's positional accuracy, and the subject's directional performance, which is a function of the user's force exertion on the robot's end-effector where a force sensor is mounted.

The effectiveness of model-based control of a dynamic multibody system is reliant upon the accuracy of a model of the actual system. Computed-Torque feed-forward control, in particular, is only applicable when a perfect (or accurate enough) model of the dynamic system is available. Otherwise, as the mismatch between the model of the system and the

real dynamics increases, an outer-loop feedback term (e.g., proportional-integral-derivative (PID) control) becomes increasingly crucial to correct for this increasing error. Different machine learning control strategies will be explored to try to remove the requirement of analytically finding an accurate model of a dynamic system, given that the input and output data of the multibody system used to represent its dynamics, can be reliably measured from previous trials using mounted sensors. The robot was also intended to be used for experiments on post-stroke patients, but COVID and other reasons prevented us from testing on post-stroke patients.

1.2 Thesis Organization

- Chapter 1 states the research motivations and goals. It then presents the thesis organization.
- Chapter 2 provides the background of this thesis. A detailed review of rehabilitation robotics is presented and the challenges and control strategies are discussed. Sections are also dedicated to the computer vision application of pose estimation and machine learning control strategies, both of which serve as background information for chapters 3 and 4, respectively.
- Chapter 3 introduces the subject-specific performance-based control by first introducing the rehabilitation robot including its hardware, software, kinematics, and dynamics. Then, a 2D human arm model is introduced including its kinematics and dynamics. The physical interaction model between the robot and the human is also discussed. A novel subject-specific performance-based control scheme is then introduced and used in an experiment on a healthy subject.
- Chapter 4 discusses the role of machine learning in robot control, and explores three machine learning control strategies: a Deep Lagrangian Network (DeLaN), a Feed-forward Neural Network (FNN), and a Recurrent Neural Network (RNN), to be used to identify the robot's dynamic model in a model-based computed-torque proportional-integral-derivative (PID) controller.
- Chapter 5 introduces the concept of subject-specific active-based control using surface electromyography (sEMG) to predict a subject's joint torques, while using an end-effector-based rehab robot that is not equipped with a force sensor.
- Chapter 6 summarizes this research. The limitations and recommended future work of this project are also discussed.

Chapter 2

Background and Literature Review

2.1 Overview

In this chapter, a literature review is conducted into all of the relevant fields of study that are used in this thesis. Furthermore, the technical foundation of these concepts, which is later referenced in the subsequent chapters, is outlined.

2.2 Stroke Rehabilitation Robotics

According to the Centers for Disease Control and Prevention (CDC), "a stroke, sometimes called a brain attack, occurs when something blocks the blood supply to a part of the brain or when a blood vessel in the brain bursts" [12]. When this occurs, the muscles in the arms, legs, hands and feet, are all affected as a result of damage to the central nervous system [21]. In particular, in the days following a stroke episode, the muscles become limp and may feel very heavy (i.e., flaccid or low tone). In the weeks or months after a stroke, the muscles may shorten and become very tight, making them more difficult to move. This is called spasticity (high-tone). These changes in muscle tone can lead to stiff, swollen, and even painful joints. Post-stroke rehabilitation tries to mitigate these effects of muscle tone changes that occur during post-stroke. But one of the most efficient and practical methods of rehabilitating post-stroke patients' motor recovery is by utilizing robotics.

The branch of medicine known as rehabilitation involves an injured patient employing daily repetitive practices under the supervision of a physiotherapist. Without removing

the need for physiotherapists, the robot essentially operates as a means of assistance in improving the performance of repetitive movements. The choice of movements, in coordination with the robotic assistance, requires the physiotherapist's clinical insight and is thus selected specifically for each patient.

In efforts to automate the rehabilitation process for post-stroke patients, while also saving considerable time and effort on the part of the physiotherapist, robots have historically been proven to be very effective [14, 38, 3, 48, 27]. Many of these rehabilitation robots are also often equipped with force/torque sensors and encoders. The data obtained by these mounted sensors can then be utilized to quantitatively analyze the patient's progress over multiple practice trials.

Utilizing rehabilitation robotics in a clinical setting also requires the development of a user interface. Rehabilitation practices often include repetitive trials which are not always compelling to stick to due to boredom after prolonged use. The practices promoted by these robots also have to be more engaging for the patients, while also useful from the perspective of the therapist. Hence, developing appropriate user interfaces for these robots in consultation with therapists is paramount in order to maximize the use of rehabilitation robots until a patient's full recovery.

2.3 Planar Rehabilitation Robot Control

The categorization of control strategies for rehabilitation robotics can be divided into **High-Level** and **Low-Level**. High-level control strategies are "explicitly designed to provoke motor plasticity". One example of high-level control is force-field control [26, 37, 36]. On the other hand, low-level strategies strive to "control the force, position, impedance, or admittance factors of high-level control strategies". Much of the work done in [19] have already addressed the performance of various low-level control strategies including Sliding Mode Control (SMC), Linear Quadratic Regulator (LQR) control, and Nonlinear Model Predictive Control (NMPC). The high-level control strategy that is of most particular interest in this thesis is subject-specific assistive control.

It is worthwhile to note, however, that the level of assistance provided by the robot to guide the subject in moving along a desired trajectory greatly affects the robot's impact on patient rehabilitation. Too little assistance may cause frustration and decrease motivation, whereas too much assistance decreases active output and encourages slacking. This is often referred to as challenge point theory, which was created by Mark A. Guadagnoli and Timothy D. Lee. In accordance with the divisions of subject-specific assistive control

highlighted in [29, 31], the two divisions of subject-specific assistive control that are of utmost interest in this thesis are active-based and performance-based adaptive controllers.

Active-Based Control

Active-based adaptive control methods primarily employ surface Electromyography (sEMG) sensors/electrodes to estimate a specific subject's **motion intention**. For instance, EMG data can be mapped back to muscle activation to then be used to determine the intended contribution of a particular muscle or a group of muscles. These control methods essentially use these proportional EMG signals to determine/trigger the robot's level of assistance for each specific patient's motor needs. In other words, they enable the robot to provide the patient with "active" training. According to [22], active training involves the robot providing limited assistance according to the participant's motion intention. This promotes subject-specific motor relearning through the participant adjusting adaptively in response to the robot's corrective forces.

The interactive model by which a rehabilitation robot measures a user's "intention" often concerns the cognitive Human-Robot Interaction (cHRI) model [41]. The cHRI model fundamentally concerns mental models (e.g., the central nervous system) and the communication channels between humans and robots. Hence, the human intent is detected before the user actually moves, and the necessary torque, velocity, and angle for the human joints may be predicted [18]. Extensive research has already been widely conducted on EMG-based active control when a subject is using a planar "end-effector" robot for the upper extremity [6, 8, 47].

For context, EMG signals record muscle activation from the muscles underlying the subject's skin resulting from brain impulses sent by the central nervous system (CNS). It delivers key movement intentions approximately 50–100 milliseconds before the user's action and then detects human muscular strength while the patient is in motion [23]. However, the intricacy of the musculoskeletal system between patients, the variations in body composition between patients, and the possibility of improperly placing electrodes that in turn cause skin irritation, all contribute to some of the limitations and drawbacks of using EMG. It is also worthwhile to note that EMG tends to not be as well-indicative of post-stroke patients' movement intentions as it can be for predicting healthy patients' intentions [6, 8]. This is because the improper placement of electrodes on post-stroke patients who are often overweight is much more prevalent, post-stroke patients often have higher movement variability when using a robotic device due to the inter-subject variability in neuromuscular system damage, and there exists a higher intra-subject variability of

muscle activation patterns amongst post-stroke patients, all of which makes it more difficult for robot controllers to accurately predict the post-stroke user's motion intention.

Performance-Based Control

Performance-based adaptive controllers use the performance data of the patient, which are often obtained using sensors, to adapt the assistance (force, impedance parameters, etc.) or to reset the practice (trajectory, time, admittance parameters, etc.). These types of controllers help to provide what is often referred to as "assist-as-needed training" and thus enable these controllers to be subject-specific based on the specific subject's performance.

The interactive model by which a rehabilitation robot conducts performance-based adaptive control is the physical Human-Robot Interaction (pHRI) model [41]. The pHRI model defines the scenario of humans and robots sharing the same workspace, coming in touch with each other, exchanging forces, and cooperating in doing actions on the environment.

2.4 Pose Estimation

Using pose estimation can allow for the estimation of the subject's movement in real-time when they are using the rehab robot. Pose estimation is a fundamental task in computer vision and artificial intelligence (AI) that involves detecting, associating, and tracking semantic key points, such that the position and orientation of human body parts in images or videos are indicated [35]. These semantic key points of the human body are more commonly referred to as human joints. This method of human joint detection is preferable amongst subjects over marker-based motion capture approaches because markerless pose estimation is non-invasive and inexpensive. Moreover, existing state-of-the-art models are experimentally accurate enough to be applied to the application of tracking post-stroke patients using a rehabilitation robot. A detailed review of the key chronological advancements in pose estimation methods is highlighted in the following subsections.

2.4.1 Traditional Pose Estimation Methods

The earliest developed pose estimation methods utilized either parts-based models or pixel-based approaches. Ramanan [40], for example, applied an edge-based deformable model to obtain soft estimates of body part positions by defining pixel labels into region types.

Low-level segmentation cues were then learned to build part-specific region models. Later iterations modified these algorithms by including methods such as grab cuts, foreground extractors, and Gaussian mixture models [11], with Support Vector Machines (SVMs) and Histogram of Orientated Gradients (HOGs) descriptors being used as detection methods [11, 10, 53].

Other early models embraced poselets, which are body parts of one's pose but are tightly clustered in both appearance and configuration space. Bourdev et al. [4], for example, used 3D human pose annotations as pose references. Later pose estimation models attempted to improve upon these methods. For instance, Johnson et al. [25] used a pictorial structure model (PSM). A PSM models a person as a connected collection of parts such as the head, torso, and limbs, by using an assortment of linear SVMs to encapsulate these parts represented by HOG descriptors.

2.4.2 Deep Learning

Pose estimation research then shifted from traditional methods to deep learning when 'DeepPose' by Toshev et al. [46] was published. DeepPose was based on the premise that (x,y) coordinates of joints are regressed using a cascade regression of deep neural networks.

In recent years, state-of-the-art pose estimation methods began to use heat maps. Heat maps were first introduced by Tompson et al. [44], where they tried to solve the regression problem of joint locations using Gaussian probability. That is, for each coordinate pair, there exists a 2D Gaussian with a small variance and mean centered at that joint location. This regression problem is thus visualized by heat maps, where the probability of the existence of a joint location at a region is intuitively indicated by the colour red, whereas the probability that there is no joint located at a region would be indicated by the colour blue.

Heat maps in pose estimation are involved in many deep learning architectures, but they are most commonly used in CNNs [34]. One of these popular architectures is the stacked hourglass architecture, which was first introduced by Newell et al. [34]. By its design, the stacked hourglass network encompasses a CNN with a multi-context attention mechanism for pose estimation through the use of residual layers. This approach uses convolutional modules in a bottom-up, top-down configuration resembling hourglasses that are stacked.

Heatmaps, however, intrinsically suffer from quantization error and require excessive computation to generate and post-process. McNally et al. [30] proposed to model individual key points and sets of spatially related key points (i.e., poses) as objects within a dense single-stage anchor-based detection framework. This method was called KAPAO

(pronounced "Ka-Pow"), for Keypoints And Poses As Objects. KAPAO was experimentally found to be faster and more accurate than previous heatmap-based methods, due to the post-processing.

2.5 Machine Learning Control Strategies

The effectiveness of the model-based control of a dynamic multi-body system is reliant upon the accuracy of a model of the actual system. Specifically, low-level control strategies such as LQR, MPC, SMC, and computed-torque PID are, in their formulations, reliant upon an accurately derived model of a system these controllers are tuned to control. Computed-Torque PID control is the focus of this section.

Computed-Torque PID control is a method for linearizing and decoupling the robotic dynamics by using perfect dynamical models of robotic systems in order to control each joint using other linear control strategies [43]. It is an approach that tries to be anticipatory, rather than totally reactionary. However, as stated, it is only applicable when a perfect (or accurate enough) model of the dynamical system is available. Otherwise, as the mismatch between the model of the system and the real dynamics increases, the outer-loop feedback PID term becomes increasingly crucial to correct for this increasing error.

Machine learning control strategies can serve to remove the requirement to find an accurate model of a system, given that the input and output data of the multi-body system used to represent its dynamics, can be reliably measured from previous trials using mounted sensors. Three machine learning architectures are explored in further detail in the following subsections below.

2.5.1 Feed-forward Neural Network (FNN)

A Feed-forward Neural Network (FNN) is a type of artificial neural network where the connections between nodes do not form a cycle [54]. The desire to learn the dynamics of any multibody system is inspired by the universal approximation theorem [42, 7]. The universal approximation theorem for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. Hence, given the fact that kinematic data representing a trial of a moving robot can be measured as input, and that torque exerted by the joints can be measured as output, an FNN can be trained to

represent the dynamics of any multibody system. In order to train such a network, back-propagation is most often used [15]. In this algorithm, the network's predicted output values are compared with the correct output values, in order to calculate the value of some predefined error function. By various techniques, the error can then be "propagated back" through the network to adjust the weights accordingly.

2.5.2 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is another type of artificial neural network that is different from an FNN in the sense that connections between nodes can form a cycle, allowing output from some nodes to affect subsequent input to the same nodes. This allows it to exhibit temporal dynamic behavior, process variable-length sequences of inputs, and thus, potentially exceed the performance of an FNN trained on the same data [9, 1, 2]. In keeping with the universal approximation theorem, an RNN should be able to be trained to represent the dynamics of any multibody system.

2.5.3 Deep Lagrangian Network (DeLaN)

Notable results are often achieved by deep learning models that have been trained with large amounts of data and have developed a suitable generalization capacity to perform well on data not seen before. In the physical world, engineers tend to prefer to employ classical modelling techniques to achieve their own notable results because they know these models ensure physical plausibility, and are thus, guaranteed to generalize to unseen data. These techniques, however, rely on physical assumptions that require knowledge about the physical embodiment of a system. Moreover, obtaining the large amounts of data that are necessary for deep learning models to adequately learn these physically embodied systems in real time can be very challenging, and often, impractical. Instead, research groups such as Lutter et. al. [28], have tried to bridge these techniques by developing novel deep learning models that both directly incorporate physical insight and generalize well beyond the training samples, while also requiring fewer training samples. This novel model is known as the "Deep Lagrangian Network" (DeLaN) upon which Lagrangian Mechanics are enforced through encoding a physics prior differential equation in a deep learning network topology [1]. This differential equation is the Euler-Lagrange equation, a second-order ODE, which is identical for all mechanical systems, except for systems with non-linear dampers and nonholonomic constraints.

DeLaNs are unique in the sense that they're able to learn the dynamics of any kinematic structure, including trees and closed loop structures, without requiring any explicit knowledge about the kinematic structure [28]. This is in strong contrast to the Newton-Euler approaches, where the features are instead specific to the kinematic structure. It must also be stated that the second-order ODE of the Euler-Lagrange equations is not being explicitly solved by the DeLaN, but rather, the DeLaN only serves to guide the deep learning model in deducing the system's equations of motion. Therefore, the ODE is essentially being generated by the DeLaN.

Within this project, Lagrangian Mechanics was used with generalized coordinates $\mathbf{q}$ that defined a system, and with the generalized forces $\mathbf{F}$ that included both conservative forces and non-conservative forces (e.g., friction), provided that they satisfied D'Alembert's principle. The Lagrangian, often denoted by L, is a non-unique function of generalized coordinates $\mathbf{q}$ that fully describes the dynamics of a mechanical system. It's defined as:

$$L = T - V \tag{2.1}$$

Where T is the kinetic energy of the system and V is the potential energy of the system. The kinetic energy T is also a function of generalized coordinates $\mathbf{q}$, and is defined as follows:

$$T = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{H}(\mathbf{q})\dot{\mathbf{q}} \tag{2.2}$$

Where $\mathbf{H(q)}$ is the frequently encountered symmetric and positive definite inertia matrix. The purpose of the positive definiteness property of the inertia matrix $\mathbf{H(q)}$ is to ensure that all non-zero velocities result in positive kinetic energies and that the inertia matrix $\mathbf{H(q)}$ is invertible (non-singular) such that forward dynamics can be conducted. After applying the calculus of variations, the familiar Euler-Lagrange equation with non-conservative forces can be obtained:

$$\frac{d}{dr}\frac{\partial L}{\partial \dot{\mathbf{q}}_i} - \frac{\partial L}{\partial \dot{\mathbf{q}}_i} = \boldsymbol{\tau}_i \tag{2.3}$$

Where $\boldsymbol{\tau}$ are the generalized forces and torques for each of the i generalized coordinates. Substituting equation 2.1 for L and $\frac{\partial V}{\partial \mathbf{q}_i} = \mathbf{g(q)}$ into equation 2.3 above results in:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{H}}(\mathbf{q})\dot{\mathbf{q}} - \frac{1}{2}\left(\frac{\partial}{\partial \mathbf{q}}(\dot{\mathbf{q}}^T \mathbf{H}(\mathbf{q})\dot{\mathbf{q}})\right)^T + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{2.4}$$

or

$$\mathbf{H(q)}\ddot{\mathbf{q}} + \mathbf{c(q,\dot{q})} + \mathbf{g(q)} = \boldsymbol{\tau} \tag{2.5}$$

where

$$\mathbf{c(q,\dot{q})} = \dot{\mathbf{H}}(\mathbf{q})\dot{\mathbf{q}} - \frac{1}{2}\left(\frac{\partial}{\partial \mathbf{q}}(\dot{\mathbf{q}}^T\mathbf{H}(\mathbf{q})\dot{\mathbf{q}})\right)^T \tag{2.6}$$

$\mathbf{c(q,\dot{q})}$ represents the centripetal and Coriolis forces and $\mathbf{g(q)}$ is the gravitational and all conservative forces. Note from equation 2.4 that $\mathbf{c(q,\dot{q})}$ is in fact a function of $\mathbf{H(q)}$. Most engineering approaches would approximate $\mathbf{H(q)}$ and $\mathbf{g(q)}$ using measured masses, lengths, and moments of inertia. Deep learning approaches, on the other hand, would not learn to approximate $\mathbf{H(q)}$ and $\mathbf{g(q)}$, but rather, it would ignore the underlying physical structure and learn the desired joint torques directly from data, essentially as a function of the generalized coordinates and its derivatives ($\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$). As stated before, DeLaN tries to bridge this gap. The means through which it achieves this is by representing the unknown functions $\mathbf{H(q)}$, $\mathbf{g(q)}$, and its derivatives as feed-forward neural networks that only depend on the generalized coordinates $\mathbf{q}$, and not its derivatives, contrary to conventional deep learning approaches.

To demonstrate the applicability and extrapolation of DeLaNs, two applications were undertaken for the purpose of analyzing the performance of DeLaNs simulated on multibody dynamic systems. The first application of DeLaNs was performed by Lutter: simulating a 2-DOF robot drawing cosine trajectories (Figure 2.1) and character letter trajectories. The last application of DeLaN was performed by this thesis' author on a simple sliding inverted pendulum model, for the purpose of training and deploying an original DeLaN architecture and evaluating its performance against FNNs and RNNs on a simpler multibody dynamic system.

2-DOF Robot

Lutter et. al. [28] evaluated the performance of their DeLaN model using the tracking error on their train and test trajectories and compared the model to an analytic inverse dynamics model (Recursive Newton-Euler algorithm), a standard feedforward neural network (FNN), and a PD-Controller. They also strictly limited all DeLaN model predictions to real-time and performed the learning online, which means that the training ran in a separate process on the same machine and solved the optimization problem online starting from random

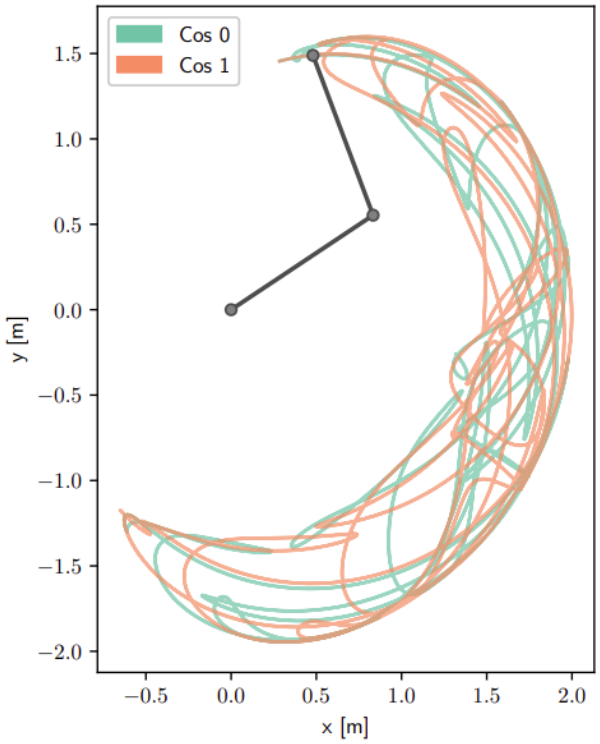

Figure 2.1: The simulated 2-DOF robot drawing the cosine trajectories [28].

initialization. Once a new model became available, the inverse model $\hat{f}^{-1}$ in the control loop was updated. Due to the limitations of the Quarc control software on Simulink, however, online learning of the stroke robot used in this thesis was not possible, but rather, only offline learning was possible and performed.

Below in figure 2.2 is Lutter's [28] real-time control loop using a PD-Controller with a feed-forward torque $\boldsymbol{\tau_{FF}}$ block to compensate the multibody system dynamics and control the robot's joint torques $\boldsymbol{\tau}$. The training process essentially reads the joint states and applies joint torques to the joints in order to learn the system dynamics online starting from random initialization.

Lutter's 2-DOF robot was simulated using PyBullet (Coumans and Bai, 2016–2018) and executed the desired character trajectories. This data set contained 20 single-stroke characters [49], spatially and temporally re-scaled to comply with the robot kinematics. Due to the different characters, the desired trajectories contained smooth and sharp turns and covered a wide variety of different shapes, but were also limited to a small task space

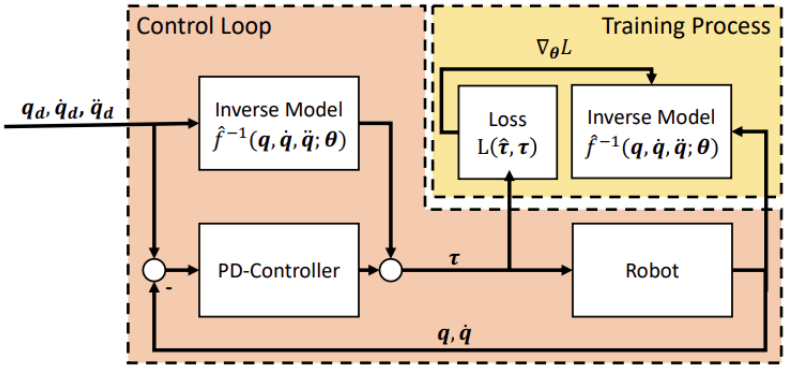

Figure 2.2: A deep Lagrangian network's training process [28].

region. Figure 2.3 below shows the ground truth torques of the characters 'a', 'd', and 'e' in black, the torque ground truth components in black, and the learned decomposition using their DeLaN model.

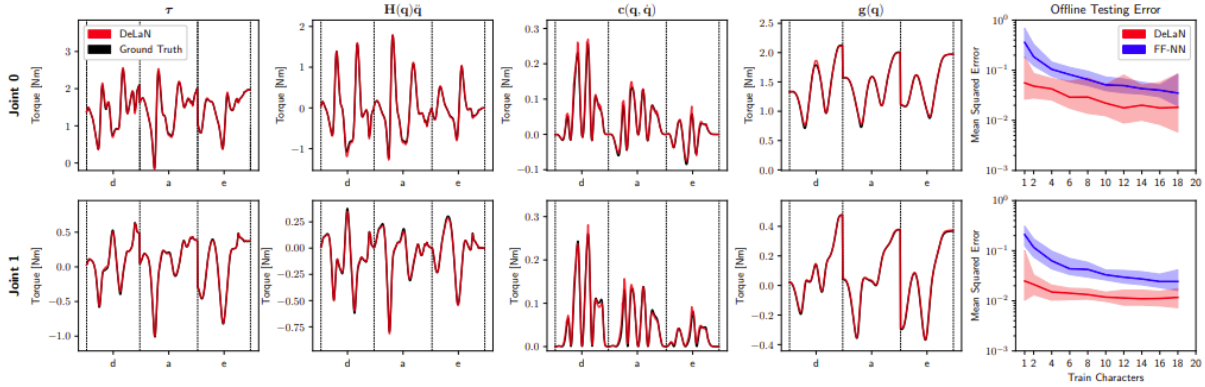

Figure 2.3: The torque required to generate the characters 'a', 'd' and 'e' in black. Using these samples DeLaN was trained offline and outputted the red trajectory [28].

It is evident from figure 2.3 that given super-imposed torques as target data, DeLaNs are able to learn and disambiguate the inertial torque contribution $\mathbf{H(q)\ddot{q}}$, the Coriolis and Centrifugal torque contribution $\mathbf{c(q,\dot{q})}$ and the gravitational and conservative torque contribution $\mathbf{g(q)}$, since the respective curves overlap quite consistently. Hence, a DeLaN is capable of learning the underlying physical model using the proposed network topology trained with standard end-to-end optimization. The rightmost column in figure 2.3 shows the offline mean squared error (MSE) on the test set for the FF-NN/FNN and the

DeLaN with respect to different training set sizes. Note also that different training set sizes correspond to the combination of n random characters (i.e., a training set size of 1 corresponds to training the model on a single character and evaluating the performance on the remaining 19 characters). Nonetheless, the DeLaN conclusively obtained a lower test MSE compared to the FF-NN/FNN, and this difference in performance tended to increase when the training set was reduced. This increasing difference highlights DeLaN's reduced sample complexity and its good extrapolation behaviour to unseen samples.

When a DeLaN is learned online starting from random initialization, this difference in performance between a DeLaN and an FF-NN/FNN is even more apparent. Figure 2.4 below shows the accumulated tracking error per testing character and the testing error averaged over all test characters. The obtained tracking error is comparable to the analytic model, which in this case contains the simulation parameters and is optimal. The quantitative comparison of the accumulated tracking error shows that a DeLaN is able to obtain a lower tracking error on all training set sizes compared to an FF-NN/FNN. This good performance using only a few training characters shows that the DeLaN characteristically has a lower sample complexity and a better ability to extrapolate to unseen trajectories when compared to an FF-NN/FNN.

Sliding Inverted Pendulum

The sliding inverted pendulum was modelled as a 2-DOF multibody system with one frictionless revolute pivot joint connecting the two masses of the rod and the base, and one prismatic joint connecting the base to the ground. Absolute coordinates were also used for both masses (i.e., $n = 6$). Below in figure 2.5 is the model that was used for the sliding inverted pendulum model, showing the definitions of two of the six absolute coordinates ($[x_1, y_1, \theta_1, x_2, y_2, \theta_2]^T$), representing the two degrees of freedom of the system. The goal was for the rod to move from -45^o to 45^o. Note that for the derivation of the equations of motion for this system, M in the figure below will be referred to as m_1, and m will be referred to as m_2.

As a baseline measure of comparison, the inverse dynamics performance of the DeLaN in computing the driving torque of the sliding inverted pendulum was compared to the driving torques computed by both a standard feed-forward artificial neural network (ANN) and a recurrent neural network (RNN). The performance of these deep learning models was determined based on the torques computed by the proprietary multibody dynamics simulation software of MSC.ADAMS, which essentially serves as a gold standard for testing deep learning models. The following measures were then taken in implementing these deep learning models: the number of parameters for each model was set such that they were

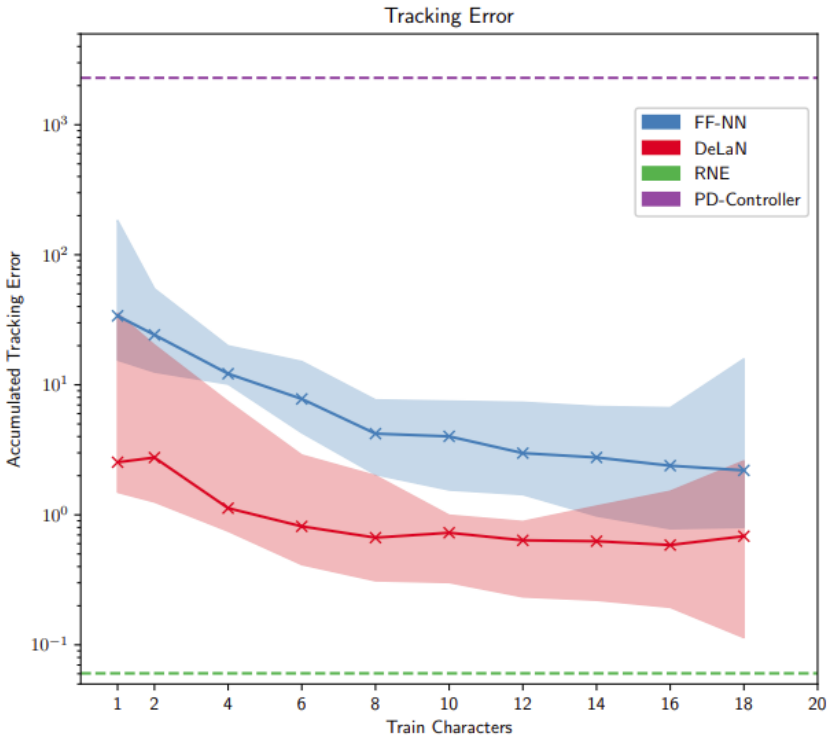

Figure 2.4: The median performance of DeLaN, the feed-forward neural network, and the analytic baselines. The shaded areas highlight the 5^{th} and the 95^{th} percentile [28].

approximately equal, the maximum number of epochs was set to 100 for training each of these three models such that a more reliable and valid comparison could be made, and lastly, the Adam (adaptive moment estimation) optimizer was used in all of the three models' parameter optimization training processes as well. In training these models, 5-fold cross-validation with random shuffling was used such that every training sample had an equal opportunity to be both trained and tested as part of the validation set and thus, make these deep learning models more robust to overfitting or selection bias. 101,000 data points were synthetically generated using the constraints of the joints and a constant rotational driving constraint (i.e., method of appended driving constraints), for different angular speeds of the rod about the revolute joint. As a result, the total number of training samples was reserved to be 100,899, while the remaining 101 samples (i.e., one phase cycle), representing the angular speed of $\theta_2 = 5$ rad/s for the sliding inverted pendulum, was reserved for testing the models against ADAMS. The detailed layout of each of these three deep learning models is detailed in the form of illustrative flowcharts below.

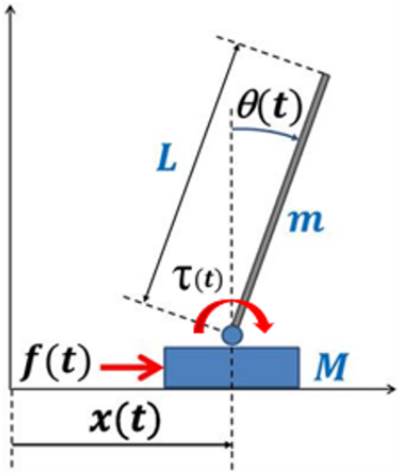

Figure 2.5: 2-DOF sliding inverted pendulum model with 2 absolute coordinates. Note that $f(t)$ is meant to represent the dynamic friction force between the foot and the ground.

According to Nasr et al. [33], the most effective configuration for this multi-layer model in mapping kinematic signals to joint torque with minimal error is a wide and shallow formation. Hence, in accordance with their recommended architecture, the ANN was implemented with only two hidden layers, and the number of neurons in each of the two hidden layers was set to approximately 4 times the number of inputs (number of inputs=6: x_1, θ_2, and its first and second derivatives), resulting in 24 neurons for each of the two hidden layers. Moreover, each layer was implemented with an activation function of the rectified linear unit (ReLU). As a result, the total number of parameters of the network was computed to be 793. The ANN configuration is depicted below in figure 2.6.

It can be stated that joint motion prediction can be improved by incorporating motion history. Therefore, it was hypothesized that RNNs might have good accuracy in joint modeling. Similar to the ANN, a wide and shallow configuration was adopted with two hidden LSTM layers, with the first LSTM layer having an input shape of 101 (101 time steps of each stance phase cycle from 0% to 100%) by 6 (number of input features: x_1, θ_2, and its first and second derivatives). This input shape was achieved by simply reshaping the input data into a 100,899 by 101 (time steps) by 6 matrix, such that each of the 999 stance phase cycles could be treated as an input to the RNN. Moreover, the number of hidden units in both of the hidden LSTM layers was set such that the number of parameters of the RNN closely matched the number of parameters used in the implemented ANN. Hence, this resulted in 7 hidden units being implemented in the first LSTM layer and 7 hidden

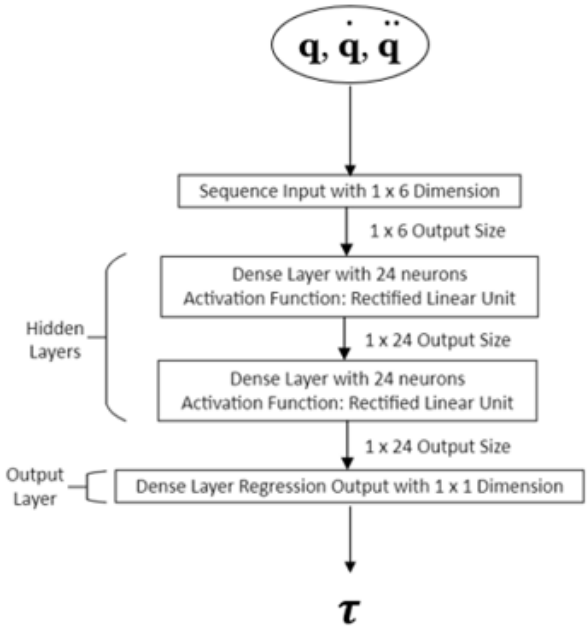

Figure 2.6: ANN model configuration.

units being implemented in the second LSTM layer of the network topology. The RNN configuration is depicted below in figure 2.7.

The DeLaN was implemented in accordance with the number of parameters used for the ANN, which was computed to be 793 using Python's model summary function. Since the DeLaN network topology consists of three parallel feed-froward artificial neural networks representing $\mathbf{H(q)}$; $\mathbf{c(q,\dot{q})}$, which is a function of $\mathbf{H(q)}$ and $\dot{\mathbf{H}}\mathbf{(q)}$; and $\mathbf{g(q)}$; the number of parameters for each of the three models was aimed to be about 265 (793/3). In an effort to satisfy this, the number of neurons in the first hidden layer was set to 12, while the number of neurons in the second hidden layer was set to 13, each with an activation function of the rectified linear unit (ReLU). In terms of inputs to the network, the number of inputs for this network was set to $n = 2$, where n is the number of degrees of freedom of the system. Analogously, the number of outputs of $\mathbf{H(q)}$ was set to $2n$, the number of outputs of $\dot{\mathbf{H}}\mathbf{(q)}$ was set to $2n$, and the number of outputs of $\mathbf{g(q)}$ was set to n. The DeLaN configuration is depicted below in figure 2.8.

In analyzing the sliding inverted pendulum's model, the first step will be to derive its equations of motion using the Lagrange formulation. Then, the performance of each of the three deep learning models will be compared based on the principles of inverse dynamics,

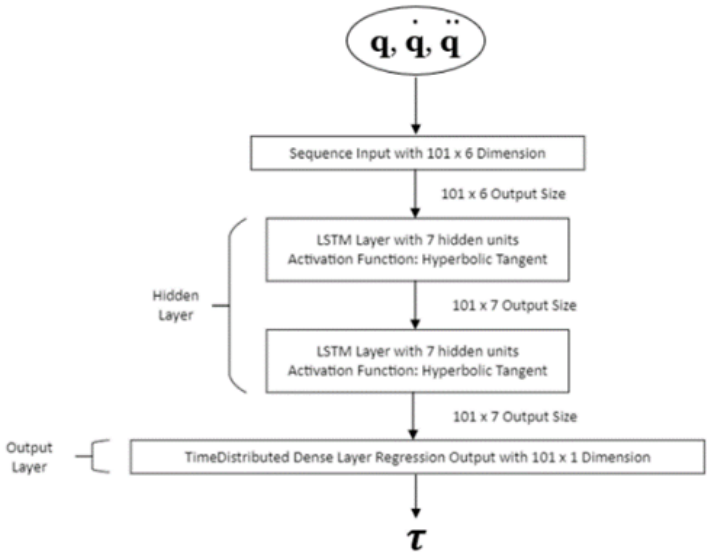

Figure 2.7: RNN model configuration.

and see how well their performances compare to the established gold standard of ADAMS. Following this comparative analysis, the performance of each of the three deep learning models will be compared again, based on a forward dynamics-driven tracking error metric of the joint angle θ_2.

When deriving the equations of motion of this 2-DOF model, the absolute coordinates of $q = [x_1, y_1, \theta_1, x_2, y_2, \theta_2]^T$ were used, and the figure 2.9 showing the local coordinate systems of each rigid body was used, with both having the same origin point in the global coordinate system (Figure 18). Note that $x_2 = x(t)$ and $\theta_2 = \theta(t)$ in figure 2.9 below.

The kinematic constraints of the system were derived based on the equations derived by Haug [20] for revolute and prismatic joints, and the motion-driving constraint of the revolute joint. Namely,

$$\{\phi\} = \begin{Bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ y_1 \\ \theta_1 \\ \theta_2 - \omega t \end{Bmatrix} = 0 \tag{2.7}$$

Now, the 5 x 6 Jacobian matrix $\{\phi\}_q$ can be easily obtained as follows:

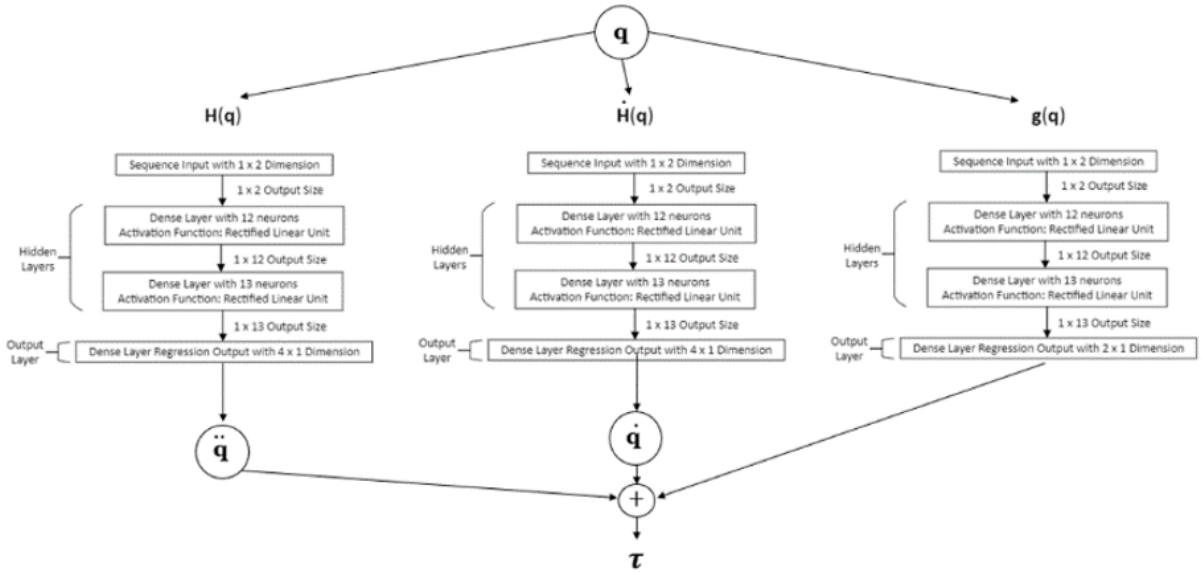

Figure 2.8: DeLaN model configuration.

$$\{\phi\}_q = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

It was from these five constraint equations where the 5 x 6 Jacobian matrix was derived and then used to synthetically generate 101,000 data points of y_1, θ_1, x_2, y_2, and θ_2. Since $\{\phi\}_q$ is a non-square matrix, and thus, non-singular for all q_i at all time steps, this system has no singularities. However, it is an unstable system at its upright equilibrium position. Specifically, it's at this point in space where there is no torque at the joint angle, but any perturbation to the system would cause a gravitational torque to act on the pendulum to make it fall over.

The velocity of the leg m can then be expressed in terms of the global frame coordinates:

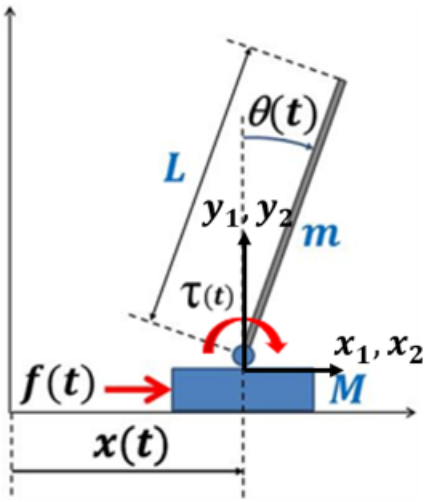

Figure 2.9: 2-DOF sliding inverted pendulum model with all 6 absolute coordinate definitions.

$$\begin{aligned} v_m &= \dot{x}_1\hat{i} + (-\dot{\theta}_2\hat{k}) \times \frac{L}{2}\left(sin(\theta_2)\hat{i} + cos(\theta_2)\hat{j}\right) \\ &= \left(\dot{x}_1 + \dot{\theta}_2\frac{L}{2}cos(\theta_2)\right)\hat{i} + \left(-\dot{\theta}_2\frac{L}{2}sin(\theta_2)\right)\hat{j} \end{aligned} \tag{2.9}$$

The Lagrangian function then becomes:

$$L = \frac{1}{2}M\dot{x}_1^2 + \frac{1}{2}I_m\dot{x}_1^2 + \frac{1}{2}m\left(\left(\dot{x}_1 + \dot{\theta}_2\frac{L}{2}cos(\theta_2)\right)^2 + \left(-\dot{\theta}_2\frac{L}{2}sin(\theta_2)\right)^2\right) - mg\frac{L}{2}cos(\theta_2) \tag{2.10}$$

In an effort to obtain the Euler-Lagrange equation for each of the 6 generalized coordinates, the Lagrange function L was inputted into the following function for each of the 6 generalized coordinates $q_i (i = 1, \ldots, 6)$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i - \sum_{j=1}^{5}\lambda_j \left(\phi_q\right)_{ji} \tag{2.11}$$

$q_1 = x_1$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_1}\right) - \frac{\partial L}{\partial x_1} = Q_{x_1} - \sum_{j=1}^{5} \lambda_j \left(\phi_q\right)_{ji,i=1} \tag{2.12}$$

$$\frac{d}{dt}\left(M\dot{x}_1 + m\left(\dot{x}_1 + \dot{\theta}_2 \frac{L}{2} cos(\theta_2)\right)\right) - 0 = f(t) - \lambda_1 \tag{2.13}$$

Simplifying,

$$M\ddot{x}_1 + m\ddot{x}_1 + m\frac{L}{2}cos(\theta_2)\ddot{\theta}_2 - m\frac{L}{2}sin(\theta_2)\dot{\theta}_2^2 = f(t) - \lambda_1 \tag{2.14}$$

$$(M+m)\ddot{x}_1 + \left(m\frac{L}{2}cos(\theta_2)\right)\ddot{\theta}_2 - m\frac{L}{2}sin(\theta_2)\dot{\theta}_2^2 + \lambda_1 = f(t) \tag{2.15}$$

$q_2 = y_1$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}_1}\right) - \frac{\partial L}{\partial y_1} = Q_{y_1} - \sum_{j=1}^{5} \lambda_j \left(\phi_q\right)_{ji,i=2} \tag{2.16}$$

$$0 - 0 = -\lambda_2 - \lambda_3 \tag{2.17}$$

$$\lambda_2 + \lambda_3 = 0 \tag{2.18}$$

$q_3 = \theta_1$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1} = Q_{\theta_1} - \sum_{j=1}^{5} \lambda_j \left(\phi_q\right)_{ji,i=3} \tag{2.19}$$

$$0 - 0 = -\lambda_4 \tag{2.20}$$

$$\lambda_4 = 0 \tag{2.21}$$

$q_4 = x_2$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_2}\right) - \frac{\partial L}{\partial x_2} = Q_{x_2} - \sum_{j=1}^{5} \lambda_j \left(\phi_q\right)_{ji,i=4} \tag{2.22}$$

$$0 - 0 = \lambda_1 \tag{2.23}$$

$$-\lambda_1 = 0 \tag{2.24}$$

$q_5 = y_2$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}_2}\right) - \frac{\partial L}{\partial y_2} = Q_{y_2} - \sum_{j=1}^{5} \lambda_j \left(\phi_q\right)_{ji,i=5} \tag{2.25}$$

$$0 - 0 = \lambda_2 \tag{2.26}$$

$$-\lambda_2 = 0 \tag{2.27}$$

Therefore, from equation 2.18,

$$\lambda_3 = 0 \tag{2.28}$$

$q_6 = \theta_2$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2} = Q_{\theta_2} - \sum_{j=1}^{5} \lambda_j \left(\phi_q\right)_{ji,i=6} \tag{2.29}$$

$$\begin{aligned}
&\frac{d}{dt}\left[I_m\dot{\theta}_2 + m\left(\dot{x}_1 + \dot{\theta}_2\frac{L}{2}cos(\theta_2)\right)\frac{L}{2}cos(\theta_2) + m\left(-\dot{\theta}_2\frac{L}{2}sin(\theta_2)\right)\left(-\frac{L}{2}sin(\theta_2)\right)\right] \\
&- \left[-m\left(\dot{x}_1 + \dot{\theta}_2\frac{L}{2}cos(\theta_2)\right)\dot{\theta}_2\frac{L}{2}sin(\theta_2) + m\left(-\dot{\theta}_2\frac{L}{2}sin(\theta_2)\right)\left(-\dot{\theta}_2\frac{L}{2}cos(\theta_2)\right) + mg\frac{L}{2}sin(\theta_2)\right] \\
&= \tau(t) - \lambda_5
\end{aligned} \tag{2.30}$$

Simplifying and substituting the equation for the mass moment of inertia of a slender rod results in:

$$I_m\ddot{\theta}_2 + m\frac{d}{dt}\left[\dot{x}_1\frac{L}{2}cos(\theta_2) + \dot{\theta}_2\frac{L^2}{4}\right] + m\dot{x}_1\dot{\theta}_2\frac{L}{2}sin(\theta_2) - mg\frac{L}{2}sin(\theta_2) = \tau(t) - \lambda_5 \quad (2.31)$$

$$\frac{1}{12}mL^2\ddot{\theta}_2 + m\frac{L}{2}cos(\theta_2)\ddot{x}_1 - m\dot{x}_1\dot{\theta}_2\frac{L}{2}sin(\theta_2) + m\ddot{\theta}_2\frac{L^2}{4} + m\dot{x}_1\dot{\theta}_2\frac{L}{2}sin(\theta_2) - mg\frac{L}{2}sin(\theta_2) = \tau(t) - \lambda_5 \quad (2.32)$$

$$\frac{1}{3}mL^2\ddot{\theta}_2 + m\frac{L}{2}cos(\theta_2)\ddot{x}_1 - mg\frac{L}{2}sin(\theta_2) = \tau(t) - \lambda_5 \quad (2.33)$$

$$\left(\frac{1}{3}mL^2\right)\ddot{\theta}_2 + \left(m\frac{L}{2}cos(\theta_2)\right)\ddot{x}_1 - mg\frac{L}{2}sin(\theta_2) + \lambda_5 = \tau(t) \quad (2.34)$$

In matrix form,

$$\begin{bmatrix} m+M & 0 & 0 & 0 & 0 & m\frac{L}{2}cos(\theta_2) \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ m\frac{L}{2}cos(\theta_2) & 0 & 0 & 0 & 0 & m\frac{L^2}{3} \end{bmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{\theta}_1 \\ \ddot{x}_2 \\ \ddot{y}_2 \\ \ddot{\theta}_2 \end{pmatrix} + \begin{bmatrix} -m\frac{L}{2}\dot{\theta}_2^2 sin(\theta_2) \\ 0 \\ 0 \\ -1 \\ 0 \\ -mg\frac{L}{2}sin(\theta_2) \end{bmatrix}$$
$$+ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{pmatrix} = \begin{bmatrix} f(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ \tau(t) \end{bmatrix} \quad (2.35)$$

Reducing the 6 ODEs above to a minimal set of forward dynamics equations in terms of the two degrees of freedom defined by x_1 and θ_2 results in:

$$\begin{bmatrix} m+M & m\frac{L}{2}cos(\theta_2) \\ m\frac{L}{2}cos(\theta_2) & m\frac{L^2}{3} \end{bmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{\theta}_2 \end{pmatrix} + \begin{bmatrix} -m\frac{L}{2}\dot{\theta}_2^2 sin(\theta_2) \\ -mg\frac{L}{2}sin(\theta_2) \end{bmatrix} = \begin{bmatrix} f(t) \\ \tau(t) \end{bmatrix} \quad (2.36)$$

In order to synthetically generate the data samples for the generalized coordinates $q_i = [x_1, y_1, \theta_1, x_2, y_2, \theta_2]^T$, the method of appended driving constraints requires the matrices of $-\{\phi\}_t$ and $\{\gamma(q, \dot{q}, t)\}$. Below are these derived matrices:

$$-\{\phi\}_t = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \omega \end{bmatrix} \tag{2.37}$$

$$\begin{aligned} \{\gamma(q, \dot{q}, t)\} &= -\left(\{\phi\}_q \{\dot{q}\}\right)_q \{\dot{q}\} - 2\{\phi\}_{qt}\{\dot{q}\} - \{\phi\}_{tt} \\ &= -\left(\{\phi\}_q \{\dot{q}\}\right)_q \{\dot{q}\} \\ &= -\begin{bmatrix} \dot{x}_1 - \dot{x}_2 \\ \dot{y}_1 - \dot{y}_2 \\ \dot{y}_1 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}_q \{\dot{q}\} \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{2.38}$$

Referring back to equation 2.36, $f(t)$ is the static friction force between the foot and the ground. The formula is as follows:

$$\begin{aligned} f(t) &= \mu N \\ &= \mu g(M + m\frac{L}{2}cos(\theta_2)) \end{aligned} \tag{2.39}$$

where μ is the static coefficient of friction and N is the vertical normal force of the entire system on the ground. The value of μ that was used was 0.6 for a concrete surface [45]. The Euler-Lagrange equation that was of utmost interest, however, was the second one (i.e., row) of equation 2.36. Namely,

$$m\frac{L}{2}cos(\theta_2)\ddot{x}_1 + m\frac{L^2}{3}\ddot{\theta}_2 - mg\frac{L}{2}sin(\theta_2) = \tau(t) \tag{2.40}$$

because this is the equation that the deep learning model comparative inverse and forward dynamic analyses were based upon. Also note that the values used for the mass and length parameters were $M = 1$ kg, $m = 10$ kg, and $L = 1$ m.

In contrast to the synthetic data generation process for y_1, θ_1, x_2, y_2, and θ_2, synthetic data for the x-position of the foot x_1 was generated by solving the first of the two ODEs above (first row of equation 2.36) for $\ddot{x}_1$, and then using backward Euler integration from the current time step to the next time step, to solve for $\dot{x}_1$ and x_1.

The ODEs of equation 2.36 were also validated against the extracted equations of a MapleSim model shown below in figure 2.10. The extracted equations matched exactly those derived.

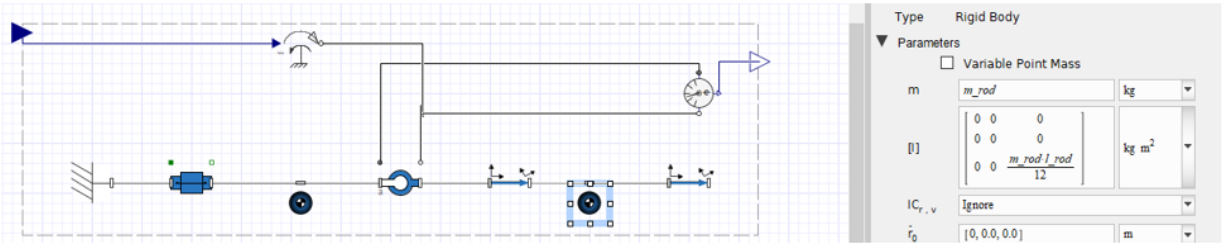

Figure 2.10: MapleSim model of the sliding inverted pendulum.

As a result of synthetically generating data for the x-position of the base, x_1, below in figure 2.11 is the horizontal displacement of the base M for an angular speed of $\dot{\theta}_2 = 5$ rad/s. Since the base was not designed to be fixed to the ground, but rather as a rigid body constrained to move along a prismatic joint connected to the ground, considerable horizontal displacement was observed.

After training each of the deep learning models on 100,899 training samples, they were then tested on the remaining 101 samples (i.e., one phase cycle), which were data samples generated for an angular speed of $\dot{\theta}_2 = 5$ rad/s. Each model's performance was then compared against the driving torque computed by ADAMS using the root mean squared error (RMSE) metric (Table 2.1), as well as plotted on the same axis as the curve generated by ADAMS (Figures 2.12 - 2.14).

Note the high peak/spike at the beginning of the stance phase cycle as computed by ADAMS. The reason for this high spike was that it was found that both the rod (m) and the base (M) endured a significant peak in accelerations at the beginning of the cycle, despite imposing the initial angular speed condition of $\dot{\theta}_2 = 5$ rad/s.

According to figures 2.12 - 2.14 and table 2.1, each of the deep learning models performed relatively well in conducting inverse dynamics on the inverted pendulum model,

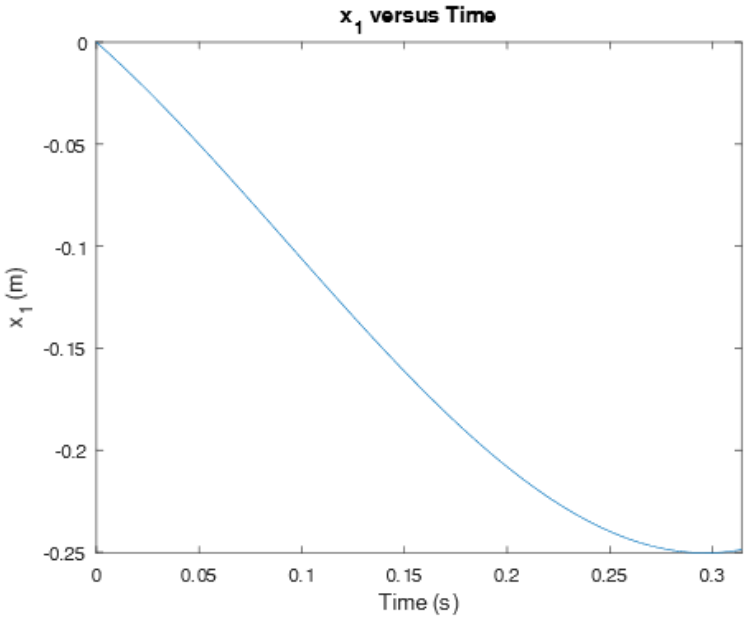

Figure 2.11: Horizontal displacement of the base.

Table 2.1: Performance of each Deep Learning Model Against ADAMS for $\dot{\theta}_2 = 5$ rad/s.

Model	**RMSE (N · m)**	**Number of Parameters**
DeLaN	2.097	801
ANN	2.257	793
RNN	2.221	820

despite the DeLaN performing slightly better than both the ANN and the RNN, even with approximately the same number of parameters in the models. Thus, in an effort to further analyze its potential use case, each of the deep learning models' performance was analyzed for different numbers of training samples, and also for a constant rotational motion driver of $\dot{\theta}_2 = 5$ rad/s. Below in figure 2.15 is the performance of each deep learning model for different numbers of training samples (i.e., distinct phase cycles), in terms of the RMSE metric. Also note that 5-fold cross-validation was used for training as well, so the RNN didn't predict until 5 phase cycles were in the training set.

It's evident from figure 2.15 that the DeLaN does indeed converge quicker than the other two deep learning models. Given the scenario of when 999 stance phase cycles are

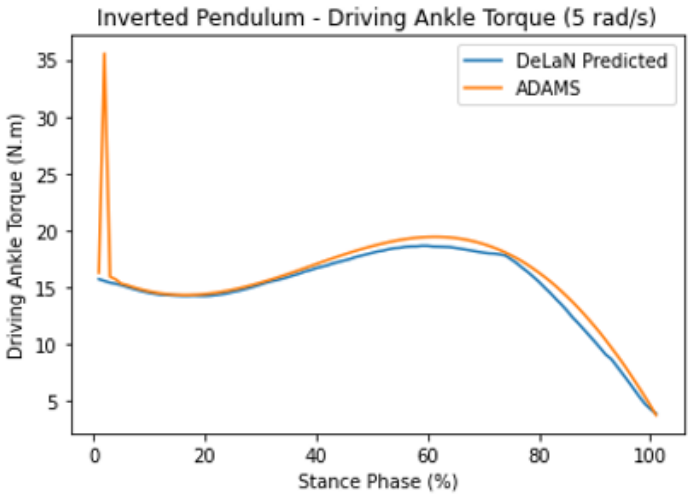

Figure 2.12: DeLaN's Performance Against ADAMS.

not available for a deep learning model to be trained on, the DeLaN is undoubtedly the preferred choice of a learning model. Moreover, unlike the ANN, RNN, and even ADAMS, the DeLaN is capable of learning the underlying physical model and disambiguating the individual torque components of the Euler-Lagrange equation. Below is the result of the DeLaN being applied to this inverted pendulum:

It's evident from figures 2.16 - 2.18 that the DeLaN is indeed capable of learning the underlying physical model and disambiguating the individual torque components of the Euler-Lagrange equation.

Forward dynamics (integration) calculations of equation 2.36 were performed for the constant rotational motion driver of $\dot{\theta}_2 = 5$ rad/s (i.e., $\ddot{\theta}_2 = 0$ rad/s^2) using backward Euler integration. Below in table 2.2 are the θ_2 tracking results in terms of the root mean squared tracking error metric.

It's evident from table 2.2 above that all three deep learning models track the joint angle θ_2 well for very small time steps (450 time steps), since equation 2.36 are stiff ODEs.

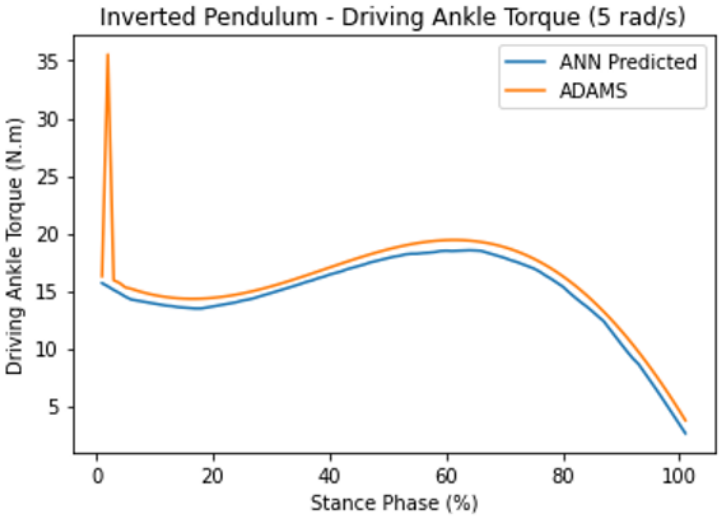

Figure 2.13: FNN/ANN's Performance Against ADAMS.

Table 2.2: Tracking performance of each deep learning model for $\dot{\theta}_2 = 5$ rad/s vs. the original generated data.

Model	θ_2 **RMSE** (rad)
DeLaN	2.089×10^{-18}
ANN	6.037×10^{-18}
RNN	5.025×10^{-18}

2.6 Summary

In this chapter, we presented the background and literature review on stroke rehabilitation robotics, robot control, pose estimation, and machine learning control strategies including FNNs, RNNs, and DeLaNs. But in particular to DeLaNs, by applying them to a simple inverted pendulum model, a series of simulations and comparisons to solutions generated by ADAMS determined that the DeLaN was the most capable of learning the underlying physical model using the proposed network topology trained with standard end-to-end optimization. That is, even though a DeLaN is trained on super-imposed torques, it's able to disambiguate the individual torque components of the Euler-Lagrange equation (i.e., inertial force, Coriolis and centrifugal forces, and gravitational force).

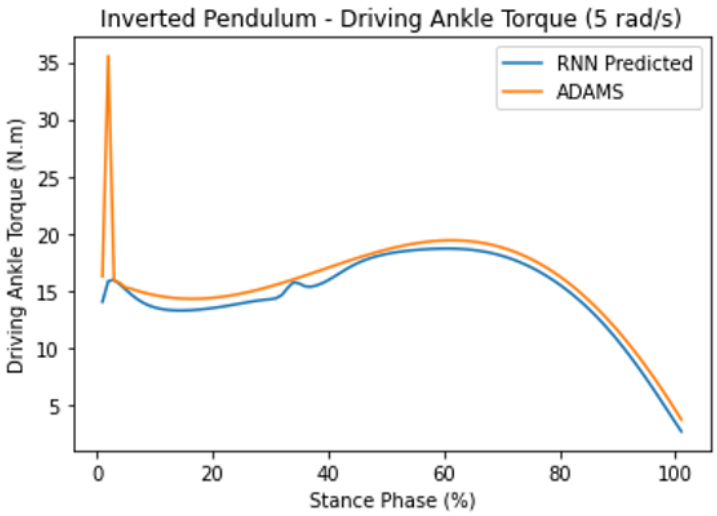

Figure 2.14: RNN's Performance Against ADAMS.

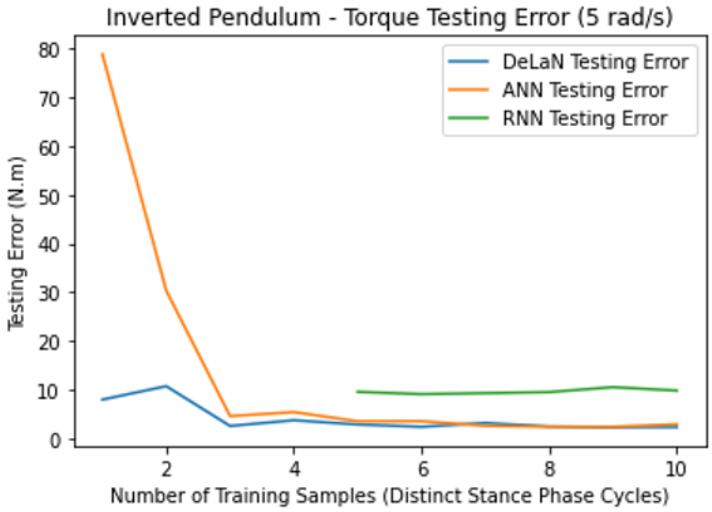

Figure 2.15: Deep Learning Model's Performance Against ADAMS vs. Number of Training Samples.

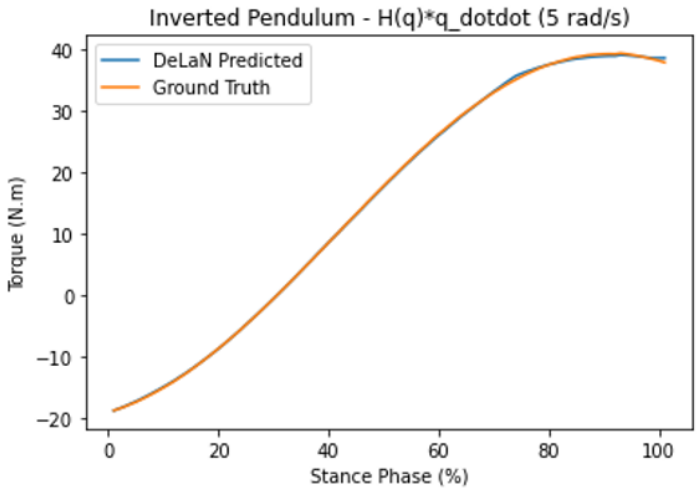

Figure 2.16: Mass Torque Components Predicted by DeLaN Against Ground Truth Values.

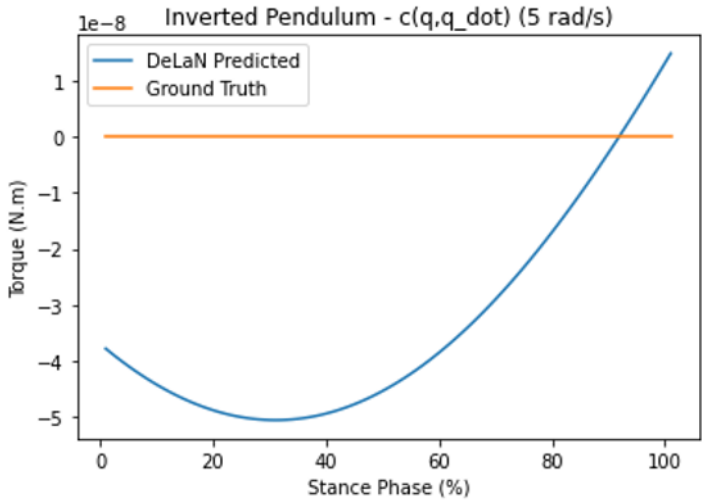

Figure 2.17: Coriolis/Centripetal Torque Components Predicted by DeLaN Against Ground Truth Values.

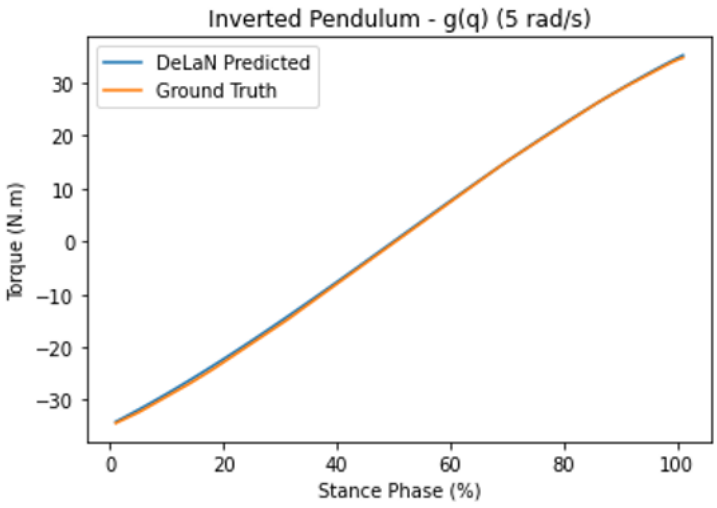

Figure 2.18: Gravitational and Conservative Torque Components Predicted by DeLaN Against Ground Truth Values.

Chapter 3

Performance-Based Assistive Control

3.1 Introduction

In this chapter, a unified model of the rehab robot and human arm is designed to facilitate a performance report for each patient. A cognitive (i.e., EMG-based) interaction model between the human and the robot was not developed because EMG was not intended to be used on post-stroke patients for the reasons described in subsection 2.3. In an effort to make the robot's performance-based control scheme subject-specific, the goal for this robot will be to adjust its joint torque contribution in accordance with the user's performance. Performance metrics will include the sum of deviations from the desired trajectory to measure the user's positional accuracy, and the direction of the user's force contribution to measure the user's directional accuracy.

3.2 Rehabilitation Robot

The rehabilitation robot studied in this thesis was developed by Quanser Inc., the Toronto Rehabilitation Institute (TRI), and the Motion Research Group (MoRG) at the University of Waterloo. The intent of developing this robot was to help in the expedition of upper extremity motor recovery. The device is an end-effector-based robot and operates only in the horizontal plane. A post-stroke patient who uses the robot does so by first grabbing the end-effector, and then repetitively moving it in the desired direction. Through the continued use of the device, the robot's purpose is to help the patient by providing varying levels of assistive/resistive forces on the user's hand.

3.2.1 Hardware

The manipulator that defines this robot is a 2-degree-of-freedom (DOF) fully actuated planar parallelogram mechanism, and consists of four Aluminum links (Fig. 3.1). It is equipped with two DC motors and two optical encoders, which are both connected to the actuated joints driving l_1 and l_2 by disc-and-timing belt mechanisms. The robot is also equipped with a six-axis force/torque sensor (ATI Industrial Automation F/T Sensor: Nano25) on the end-effector. The specification of the motors, encoders, and force sensor is shown below in Table 3.1.

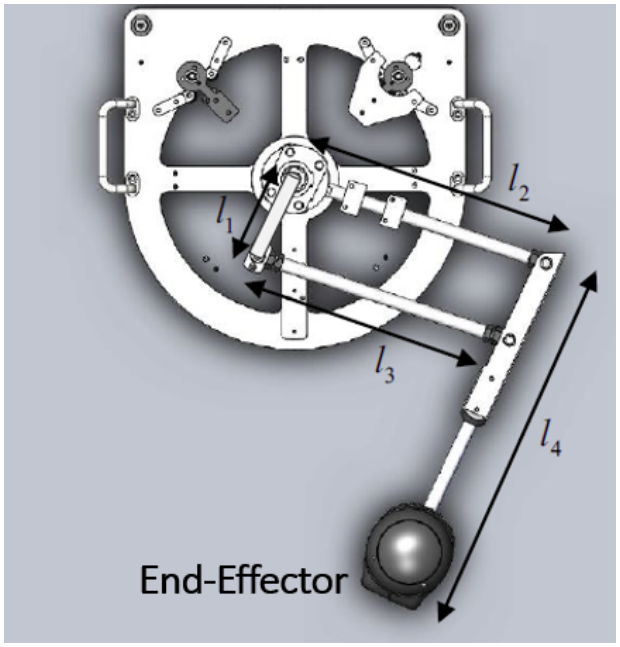

Figure 3.1: Stroke rehabilitation robot (top view).

3.2.2 Software

Quanser's QUARC real-time control software is used by this robot. Upon the building of a Simulink model, Quarc generates real-time code and then runs this code on the Windows target of the actual robot by sending the generated C code to the RAM for implementation in the external mode. The software uses a Q8 Quanser Data Acquisition (DAQ) card. The encoder data is read by a HIL Read block in the Simulink model. The force sensor data

Table 3.1: Stroke rehabilitation robot hardware specifications.

Motor Torque Constant	$K_T = 0.115\ N \cdot m/Amp$
Gear Ratio	$r = 16 : 307$
Rated Motor Torque	115 mN-m
Motor Encoder Resolution	4000 count/revolution
Force Sensor Limit	250 N (Horizontal), 1000 N (Normal)
Force Sensor Resolution	1/24 N (Horizontal), 1/48 N (Normal)

is read using a National Instruments (NI) card and a HIL Analog Read block. The motor current is calculated by dividing the controller torque by the torque constant, which is defined in Table 3.1. The HIL Write block is then utilized for sending the torque commands to the physical motors. QUARC stream functions also enable the data transfer between MATLAB scripts and Simulink models.

3.2.3 Kinematics

To derive the kinematics of this robot, the length of the links that comprise the planar parallelogram mechanism first needs to be defined. Hence, the link lengths are visualized in 3.1 and stated below in Table 3.3.

Table 3.2: Link lengths.

l_1	0.100 m
l_2	0.310 m
l_3	0.310 m
l_4	0.375 m

Below in figure 3.2 is an augmentation of figure 3.1, where the origin of the coordinate system and the end-effector position are symbolically displayed.

From figure 3.2, the robot joint angles representing the generalized coordinates of the multibody dynamic system can be defined using inverse kinematics:

$$\boldsymbol{\theta_R} = \begin{bmatrix} \theta_{R1} \\ \theta_{R2} \end{bmatrix} = \begin{bmatrix} tan^{-1}\left(\frac{l_{1,y}}{l_{1,x}}\right) \\ tan^{-1}\left(\frac{l_{2,y}}{l_{2,x}}\right) \end{bmatrix} \tag{3.1}$$

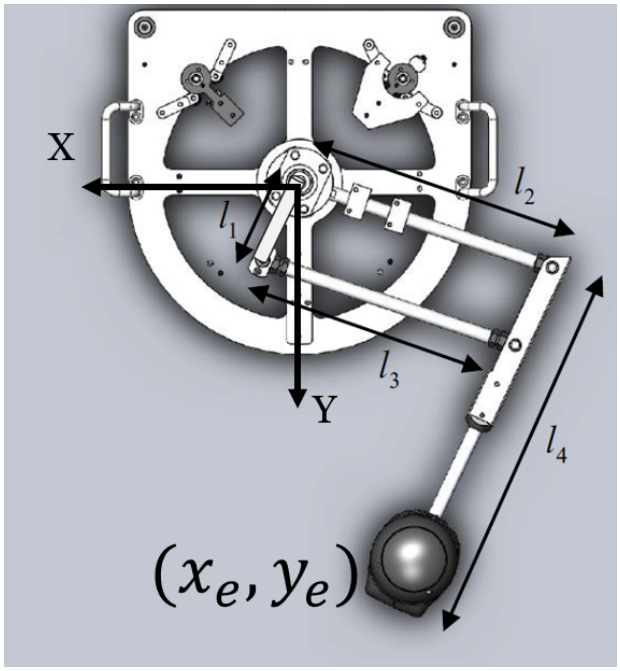

Figure 3.2: Stroke rehabilitation robot (top view) with the coordinate system location and end-effector position displayed.

Equation 3.2 below represents the forward kinematics of the robot, while equation 3.3 outlines the resulting robot's geometric Jacobian.

$$\boldsymbol{P_R} = \begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} l_2 cos(\theta_{R2}) + l_4 cos(\theta_{R1}) \\ l_2 sin(\theta_{R2}) + l_4 sin(\theta_{R1}) \end{bmatrix} \tag{3.2}$$

$$\boldsymbol{J_R} = \frac{\partial P_R}{\partial \theta_R} = \begin{bmatrix} -l_2 sin(\theta_{R2}) & -l_4 sin(\theta_{R1}) \\ l_2 cos(\theta_{R2}) & l_4 cos(\theta_{R1}) \end{bmatrix} \tag{3.3}$$

It is important to note that the forward kinematics of the robot transforms the robot joint space into the end-effector workspace. Hence, $\boldsymbol{P_R}$ is the robot's end-effector position in the workspace. Using the geometric Jacobian defined in equation 3.3, the relation between the velocity and acceleration levels in the joint space and workspace can be derived below in equations 3.4 and 3.5:

$$\dot{\boldsymbol{P}}_{\boldsymbol{R}} = \boldsymbol{J_R} \dot{\boldsymbol{\theta}}_{\boldsymbol{R}} \tag{3.4}$$

$$\ddot{\boldsymbol{P}}_{\boldsymbol{R}} = \dot{\boldsymbol{J}}_{\boldsymbol{R}}\dot{\boldsymbol{\theta}}_{\boldsymbol{R}} + \boldsymbol{J}_{\boldsymbol{R}}\ddot{\boldsymbol{\theta}}_{\boldsymbol{R}} \tag{3.5}$$

3.2.4 Dynamics

The robot's mass, inertia, and center of gravity (CG) locations were required for modeling and are shown below in Table 3.3:

Table 3.3: Link masses, moments of inertia, and centres of gravity (CG).

Link l_i	**Mass** (kg)	**Moment of Inertia** $(kg \cdot m^2)$	**CG X**	**CG Y**
$i = 1$	$m_1 = 2.578$	$J_1 = 0.022$	$x_1 = -0.005$	$y_1 = 0$
$i = 2$	$m_2 = 3.399$	$J_2 = 0.061$	$x_2 = 0.001$	$y_2 = 0$
$i = 3$	$m_3 = 0.062$	$J_3 = 0.001$	$x_3 = 0.158$	$y_3 = 0$
$i = 4$	$m_4 = 1.083$	$J_4 = 0.01$	$x_4 = 0.274$	$y_4 = 0.008$

The dynamic model of the robot, which was derived and used in all model-based controllers by previous research [13, 19, 26], is as follows:

$$\boldsymbol{M_R}\ddot{\boldsymbol{\theta}}_{\boldsymbol{R}} + \boldsymbol{C_R}\dot{\boldsymbol{\theta}}_{\boldsymbol{R}} + \boldsymbol{K_R}(\boldsymbol{\theta_R} - \boldsymbol{\theta_{R_0}}) + \boldsymbol{J^T}_{\boldsymbol{R}}\boldsymbol{f_{RE}} + \boldsymbol{f_{RJ}} = \boldsymbol{\tau_R} \tag{3.6}$$

where $\boldsymbol{M_R}$ is the inertia matrix, $\boldsymbol{C_R}$ is the Coriolis or centrifugal matrix, $\boldsymbol{K_R}$ is the $2x2$ symmetric joint stiffness matrix (number of stiffness parameters is 3), θ_{R_0} is the equilibrium position vector (number of equilibrium angle parameters is 2), f_{RE} is the end-effector friction, f_{RJ} is joint frictions, and $\tau_R \, \epsilon \, \mathbb{R}^2$ is the applied robot motor torque. Note that friction terms were modeled using a continuous-velocity friction model in previous research [5]. The robot's inertia matrix, $\boldsymbol{M_R}(\theta_R)$, and Coriolis-centripetal matrix, $\boldsymbol{C_R}(\theta_R, \dot{\theta}_R)$, used in 3.6 are defined from [13, 19, 26] as:

$$\boldsymbol{M_R}(\theta_R) = \begin{bmatrix} \alpha_1 & \alpha_2 c_{12} + \alpha_3 s_{12} \\ \alpha_2 c_{12} + \alpha_3 s_{12} & \alpha_4 \end{bmatrix} \tag{3.7}$$

$$\boldsymbol{C_R}(\theta_R, \dot{\theta}_R) = \begin{bmatrix} 0 & (\alpha_2 s_{12} - \alpha_3 c_{12})\dot{\theta}_{R2} \\ (\alpha_3 c_{12} - \alpha_2 s_{12})\dot{\theta}_{R1} & 0 \end{bmatrix} \tag{3.8}$$

where

$$c_{12} = cos(\theta_{R1} - \theta_{R2}) \tag{3.9}$$

$$s_{12} = sin(\theta_{R1} - \theta_{R2}) \tag{3.10}$$

$$\alpha_1 = ({x_1}^2 + {y_1}^2)m_1 + m_3{l_1}^2 + ({x_4}^2 + {y_4}^2)m_4 + J_1 + J_4 \tag{3.11}$$

$$\alpha_2 = m_3 l_1 x_3 + m_4 l_2 x_4 \tag{3.12}$$

$$\alpha_3 = m_3 l_1 y_3 - m_4 l_2 y_4 \tag{3.13}$$

$$\alpha_4 = ({x_2}^2 + {y_2}^2)m_2 + m_4{l_2}^2 + ({x_3}^2 + {y_3}^2)m_3 + J_2 + J_3 \tag{3.14}$$

3.3 2D Human Arm Model

In order to develop an interaction model between the robot and the patient, a model of human arm first has to be derived. The user's human arm was modelled as a two-dimensional 2-DOF linkage (figure 3.3); it includes one degree of freedom for elbow flexion/extension and one degree of freedom for shoulder rotation.

3.3.1 Kinematics

From figure 3.3, the joint angles representing the generalized coordinates of the multibody dynamic human arm model can be defined using inverse kinematics:

$$\boldsymbol{\theta_A} = \begin{bmatrix} \theta_s \\ \theta_e \end{bmatrix} \tag{3.15}$$

Equation 3.16 below represents the forward kinematics of the human arm model, while equation 3.17 outlines the resulting human arm's geometric Jacobian.

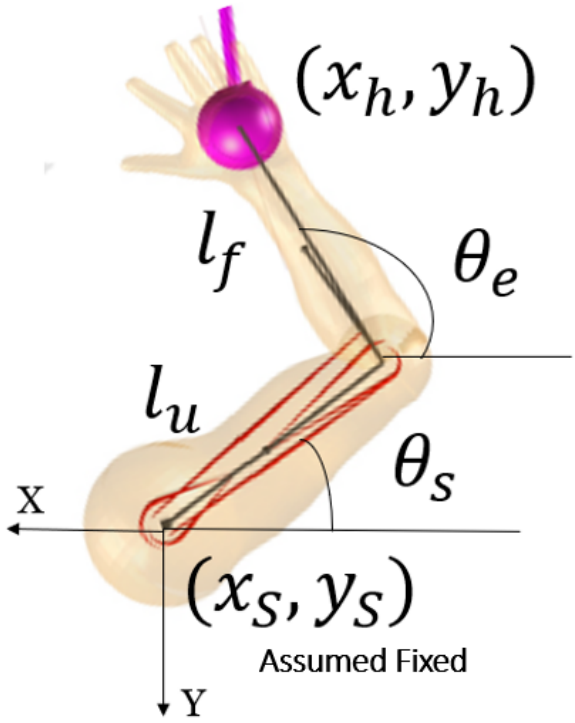

Figure 3.3: Planar human arm model (top view) with the coordinate system (shoulder joint location), link lengths, joint angles, and hand position displayed.

$$\boldsymbol{P_A} = \begin{bmatrix} x_h \\ y_h \end{bmatrix} = \begin{bmatrix} -l_u cos(\theta_s) - l_f cos(\theta_e) \\ -l_u sin(\theta_s) - l_f sin(\theta_e) \end{bmatrix} + \begin{bmatrix} x_s \\ y_s \end{bmatrix} \tag{3.16}$$

$$\boldsymbol{J_A} = \frac{\partial P_A}{\partial \theta_A} = \begin{bmatrix} l_u sin(\theta_s) & l_f sin(\theta_e) \\ -l_u cos(\theta_s) & -l_f cos(\theta_e) \end{bmatrix} \tag{3.17}$$

Similar to the planar model of the robot, the relationship between the human arm joint and the hand position in the workspace can now be derived below:

$$\dot{\boldsymbol{P}}_{\boldsymbol{A}} = \boldsymbol{J_A}\dot{\boldsymbol{\theta}}_{\boldsymbol{A}} \tag{3.18}$$

$$\ddot{\boldsymbol{P}}_{\boldsymbol{A}} = \dot{\boldsymbol{J}}_{\boldsymbol{A}}\dot{\boldsymbol{\theta}}_{\boldsymbol{A}} + \boldsymbol{J_A}\ddot{\boldsymbol{\theta}}_{\boldsymbol{A}} \tag{3.19}$$

3.3.2 Dynamics

The dynamic equation of motion (EOM) of this mechanism is shown below in equation 3.20, of which the actual terms were derived from MapleSim.

$$\boldsymbol{M_A}\ddot{\boldsymbol{\theta}}_{\boldsymbol{A}} + \boldsymbol{C_A}\dot{\boldsymbol{\theta}}_{\boldsymbol{A}} = \boldsymbol{\tau_A} \tag{3.20}$$

where $\boldsymbol{M_A}$ is the inertia matrix, $\boldsymbol{C_A}$ is the Coriolis or centrifugal matrix, and $\tau_A \, \epsilon \, \mathbb{R}^2$ is the human arm's motor torque.

The human arm's inertia matrix, $\boldsymbol{M_A}(\theta_A)$, and Coriolis-centripetal matrix, $\boldsymbol{C_A}(\theta_A, \dot{\theta}_A)$, used in 3.20 are defined as:

$$\boldsymbol{M_A}(\theta_A) = \begin{bmatrix} \beta_1 + 2\beta_2 + \frac{12817}{46875} m_u l_u^2 + l_u^2 m_f & \beta_1 + 2\beta_2 \\ \beta_1 + \beta_2 & \beta_1 \end{bmatrix} \tag{3.21}$$

$$\boldsymbol{C_A}(\theta_A, \dot{\theta}_A) = \begin{bmatrix} 0 & -\beta_3(\dot{\theta}_e + 2\dot{\theta}_s) \\ \beta_3 & 0 \end{bmatrix} \tag{3.22}$$

where

$$\beta_1 = \frac{8047}{30000} m_f l_f^2 \tag{3.23}$$

$$\beta_2 = \frac{43}{100} l_f m_f l_u cos(\theta_e) \tag{3.24}$$

$$\beta_3 = \frac{43}{100} l_f m_f l_u sin(\theta_e) \tag{3.25}$$

The subscripts f and u for the arm masses (m) and lengths (l) refer to the human's forearm and upper arm, respectively. Hence, the upper arm and forearm are assumed to be slender rods for computing the mass moments of inertia about the axis perpendicular to the horizontal plane.

Note also the absence of a stiffness term in equation 3.20. In order to obtain the stiffness parameters of specific patients, a biofidelic Muscle Torque Generator (MTG) [24] could be used. However, this would require the use of equipment such as a Biodex, which is not very portable and requires an extra time commitment from patients to acquire accurate stiffness parameter values. Hence, τ_A in equation 3.20 represents the net torque of the human joints, including the passive torque contributions in the joints.

3.4 Physical Human-Robot Interaction Model (Contact Force)

A post-stroke patient's human arm interacts with the robot by manipulating the end-effector, while the robot performs repetitive rehabilitation practices for the patient in the horizontal plane. A passive revolute joint on the end-effector was utilized to integrate the two systems by using the measured interaction contact force $\boldsymbol{F_I}$ between the rehab robot and the patient's human arm. Equations 3.26 and 3.27 below model this result of the human-robot interaction model, which in turn, results in an extra term in equations 3.6 and 3.20, respectively:

$$\boldsymbol{M_R\ddot{\theta}_R + C_R\dot{\theta}_R + K_R(\theta_R - \theta_{R_0}) + {J_R}^T f_{RE} + f_{RJ} = \tau_R - {J_R}^T F_I} \tag{3.26}$$

$$\boldsymbol{M_A\ddot{\theta}_A + C_A\dot{\theta}_A = \tau_A + {J_A}^T F_I} \tag{3.27}$$

This interaction model also results in the following kinematic constraints, where the position (equations 3.2 and 3.16), velocity (equations 3.4 and 3.18), and acceleration (equations 3.5 and 3.19) of both the end-effector and the patient's hand in the workspace are always equal. This is outlined below in equations 3.28-3.31:

$$\begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} l_2 cos(\theta_{R2}) + l_4 cos(\theta_{R1}) \\ l_2 sin(\theta_{R2}) + l_4 sin(\theta_{R1}) \end{bmatrix} = \begin{bmatrix} x_h \\ y_h \end{bmatrix} = \begin{bmatrix} -l_u cos(\theta_s) - l_f cos(\theta_e) \\ -l_u sin(\theta_s) - l_f sin(\theta_e) \end{bmatrix} + P_S \tag{3.28}$$

where

$$\boldsymbol{P_S} = \begin{bmatrix} x_s \\ y_s \end{bmatrix} \tag{3.29}$$

Since the shoulder joint location of the human is encouraged to be stationary by the operating instructions in the GUI (Figure A.1), the following constraint equations can be used:

$$\boldsymbol{J_R\dot{\theta}_R = J_A\dot{\theta}_A + \dot{P}_S = J_A\dot{\theta}_A} \tag{3.30}$$

$$\dot{\boldsymbol{J}}_{\boldsymbol{R}}\dot{\boldsymbol{\theta}}_{\boldsymbol{R}} + \boldsymbol{J}_{\boldsymbol{R}}\ddot{\boldsymbol{\theta}}_{\boldsymbol{R}} = \dot{\boldsymbol{J}}_{\boldsymbol{A}}\dot{\boldsymbol{\theta}}_{\boldsymbol{A}} + \boldsymbol{J}_{\boldsymbol{A}}\ddot{\boldsymbol{\theta}}_{\boldsymbol{A}} \tag{3.31}$$

To obtain the joint positions $\boldsymbol{\theta_A}$ of a subject's planar right upper arm, there were two avenues to achieve this objective. They were either pose estimation or regression techniques. Pose estimation has already been explored in section 2.4, but the regression technique relies on the assumption that the patient's shoulder joint position is known and does not move throughout the duration of the trial. Specifically, the regression technique involves solving equation 3.28 for θ_{R1} and θ_{R2}, while x_s and y_s are known constants. Using the relations defined in equations 3.30 and 3.31, the joint velocity and desired joint acceleration of the patient's arm can then be analytically estimated. Namely,

$$\dot{\boldsymbol{\theta}}_{\boldsymbol{A}} = \boldsymbol{J}_{\boldsymbol{A}}^{-\mathbf{1}}\boldsymbol{J}_{\boldsymbol{R}}\dot{\boldsymbol{\theta}}_{\boldsymbol{R}} \tag{3.32}$$

$$\ddot{\boldsymbol{\theta}}_{\boldsymbol{A}} = \boldsymbol{J}_{\boldsymbol{A}}^{-\mathbf{1}}(\dot{\boldsymbol{J}}_{\boldsymbol{R}}\dot{\boldsymbol{\theta}}_{\boldsymbol{R}} + \boldsymbol{J}_{\boldsymbol{R}}\ddot{\boldsymbol{\theta}}_{\boldsymbol{R}} - \dot{\boldsymbol{J}}_{\boldsymbol{A}}\dot{\boldsymbol{\theta}}_{\boldsymbol{A}}) \tag{3.33}$$

Singularities occur when the determinant of the human arm's geometric Jacobian is zero. This occurs either when θ_s or θ_e is zero (unlikely given the desired trajectory is set to avoid this), or when $\theta_s = \theta_e$ (i.e., straight arm). To avoid encountering singularities from which the inverse of the human arm's Jacobian matrix does not exist due to a subject's arm being straight, the Moore-Penrose pseudo-inverse of the Jacobian matrix is calculated instead of the exact inverse, essentially treating singular values as zero.

This estimation scheme of the joint angles was validated against the gold standards of a digital goniometer and a pre-trained pose estimation model [52], when performed on a healthy male subject for a total of 10 frames/poses. Specifically, the healthy male subject traversed the robot's end-effector to 10 different locations within the bounds of the ideal post-stroke patient workspace, from which an image of the pose was then taken by an overhead camera, as shown from the perspective in figure 3.4. The results are summarized below in table 3.4, along with an example video frame depicting pose estimation being performed on the subject in figure 3.4. Note also that for each row, the first error column is defined by subtracting the regression value from the measured goniometer value, and the second error column is defined by subtracting the regression value from the value obtained by the pre-trained pose estimation model.

Using the angular velocity and acceleration values computed from equations 3.32 and 3.33, equation 3.27 can then be used to calculate the human joint torque τ_A, in an effort to obtain the amount of exertion that is being exhibited by the patient. This quantitative

Table 3.4: Human joint angle validation results.

Frame #	**Joint**	**Regression**	**Goniometer**	**Error**	**Pose Estimator**	**Error**
1	Shoulder	69.8^o	69.3^o	0.52^o	69.1^o	0.77^o
1	Elbow	101^o	101^o	-0.56^o	99.7^o	1.03^o
2	Shoulder	76.4^o	75.8^o	0.56^o	77.3^o	-0.94^o
2	Elbow	92.6^o	93.2^o	-0.65^o	91.7^o	0.89^o
3	Shoulder	79.7^o	78.9^o	0.76^o	80.5^o	-0.87^o
3	Elbow	82.9^o	83.5^o	-0.59^o	82.0^o	0.88^o
4	Shoulder	74.6^o	73.9^o	0.69^o	75.5^o	-0.95^o
4	Elbow	78.3^o	79.0^o	-0.74^o	77.4^o	0.88^o
5	Shoulder	63.4^o	62.6^o	0.84^o	64.4^o	-0.97^o
5	Elbow	83.1^o	82.3^o	0.82^o	82.4^o	0.73^o
6	Shoulder	57.6^o	58.1^o	-0.48^o	56.8^o	0.80^o
6	Elbow	94.1^o	93.7^o	0.37^o	93.4^o	0.68^o
7	Shoulder	60.8^o	61.7^o	-0.95^o	61.7^o	-0.90^o
7	Elbow	106^o	105^o	0.60^o	106^o	-0.86^o
8	Shoulder	64.6^o	65.3^o	-0.74^o	63.8^o	0.77^o
8	Elbow	99.3^o	98.7^o	0.62^o	100^o	-0.89^o
9	Shoulder	70.7^o	70.0^o	0.72^o	69.9^o	0.82^o
9	Elbow	88.0^o	88.1^o	-0.12^o	88.8^o	-0.77^o
10	Shoulder	68.8^o	69.6^o	-0.77^o	69.7^o	-0.87^o
10	Elbow	80.3^o	79.7^o	0.61^o	80.6^o	-0.29^o
RMSE	-	-	-	0.66^o	-	0.84^o

evaluation metric can then be used by the therapist as a reason to adjust the trajectory parameters, such as the speed of the end-effector.

According to equation 3.27, the Mass matrix and Coriolis/Centripetal matrix are required for each specific patient, which in turn, depend on the subject's mass and length properties. These properties are needed to find the forearm and upper arm's moment of inertia and centre of mass. The means through which this is achieved is guided by Winter's regression equations, as shown below in figure 3.5 [50].

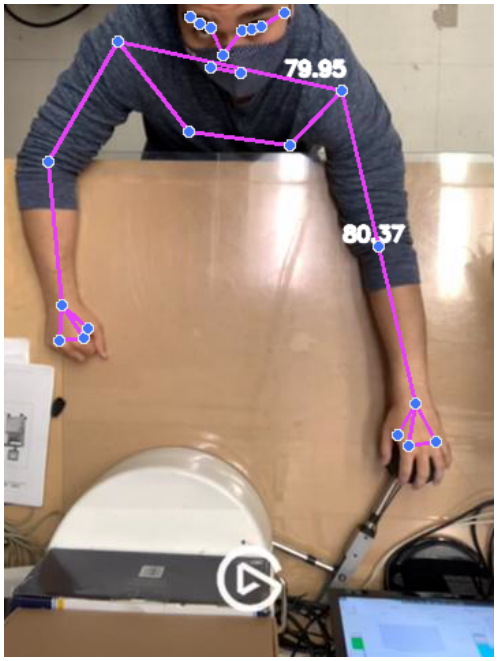

Figure 3.4: Pose estimate example.

3.5 Performance-Based Control Scheme

With the original intent to conduct trials on post-stroke patients who would use this planar rehab robot, EMG was decided not to be used on post-stroke patients for the reasons described in subsection 2.3. Hence, the interaction model to be used in making the robot controller subject-specific was the pHRI described in the previous section for the purpose of subject-specificity being performance-based.

Performance-based subject-specificity of the robot control was intended to be achieved by means of the force sensor. That is, if the interaction force F_I exerted on the end-effector results in the patient's hand following a direction that is within a small angular tolerance value (e.g., 20^o) of the desired trajectory, then the torque contribution of the robot can be reduced in adaptation with the human user's contribution. This, in turn, promotes adaptive performance-based (i.e., reduced) assistance. As opposed to adjusting the applied robot torque at every time instant relative to the direction of the subject's applied force, the reason for only adjusting the applied robot torque when the subject is within a small angular tolerance value of the desired direction is because the robot's controller already applies an aggressive corrective torque when the end-effector deviates

TABLE 4.1 Anthropometric Data

Segment	Definition	Segment Weight/Total Body Weight	Center of Mass/ Segment Length Proximal	Center of Mass/ Segment Length Distal	Radius of Gyration/ Segment Length C of G Proximal	Radius of Gyration/ Segment Length Distal	Density
Hand	Wrist axis/knuckle II middle finger	0.006 M	0.506	0.494 P	0.297	0.587	0.577 M	1.16
Forearm	Elbow axis/ulnar styloid	0.016 M	0.430	0.570 P	0.303	0.526	0.647 M	1.13
Upper arm	Glenohumeral axis/elbow axis	0.028 M	0.436	0.564 P	0.322	0.542	0.645 M	1.07

Figure 3.5: Snapshot of table 4.1 from Winter (2009) [50].

from the desired trajectory. It was determined from experiments on the robot that the resulting behavior can lead to excessive torques in high position-error scenarios, which can then lead to further discomfort for the post-stroke patient. Hence, only decreasing the applied robot torque based on the subject's performance was the adaptive control scheme that was adopted. A flowchart of this control scheme is depicted below in figure 3.6.

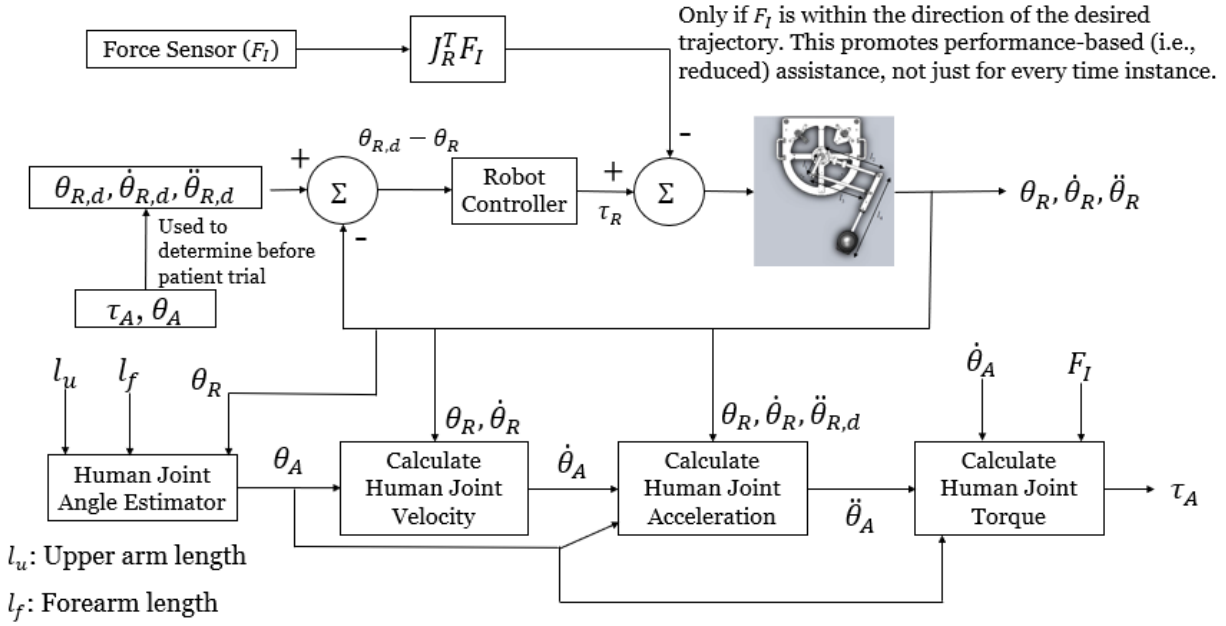

Figure 3.6: Subject-specific performance-based control scheme.

3.6 Experimental Results on a Healthy Subject

In the absence of post-stroke patients at Grand River Hospital, tests were performed on a healthy male subject (age: 24 years old, forearm length: 12 in., upper arm length: 12 in., weight: 91 kg, height: 180 cm) for the purpose of showcasing the potential of this subject-specific performance-based control scheme. The healthy subject was tasked with keeping their shoulder as still as possible to circumvent any undesired movements and allow for camera-less pose estimation when following a D-shaped reference trajectory. The reason a D shape reference trajectory was chosen is that in consultation with therapists' feedback, exercises that help to promote external rotation rather than internal rotation of the shoulder joint are more useful in rehabilitating a post-stroke patient's motor recovery. Post-stroke patients have a tendency to drop their elbows toward their bodies because of their lack of ability to lift their arms. The intention of the exercise is to promote more muscle contribution in the upper arm. The D-shape trajectory helps to ensure the patient is stretching their arm outwards to prevent the dropping of the arm.

The desired shoulder joint location was positioned at 0.12 m along the positive x-direction and 0.73 m along the positive y-direction, both from the origin with reference to the global frame coordinate system used in figure 3.2. The time to complete one full D traversal was set to 4 seconds. The radius of the half-circle defining the D shape was set to 0.1 m. The location of the centre of the left vertical line of the D shape was positioned at 0.3 m along the negative x-direction and 0.3 m along the positive y-direction, both from the origin with reference to figure 3.2. All of these parameter values were essentially inputted into the parameters tab of the GUI, as shown in figure A.7. And lastly, the robot itself was controlled by an SMC controller, which was experimentally proven to be the best-performing controller in terms of accurately following a desired trajectory [19].

3.6.1 Subject-Specific Trajectory Performance Results

Subject-specificity within this adaptive control scheme is largely based on the subject's mid-trial performance. As explained in section A.7, the trajectory performance of the subject is divided into two categories: positional performance and the subject's directional performance, which is a function of their force exertion on the robot's end-effector where the force sensor is located. The trajectory performance of the healthy subject described previously is shown below in figure 3.7. Note that for visualization purposes for both the post-stroke patient and the therapist, the positive axis directions shown in figure 3.7 are opposite to the direction used by the local coordinate system of the robot and thus should be taken into account for the discussion.

Figure 3.7: Trajectory performance of the healthy subject, including the 2D positional performance and directional performance.

Figure 3.7 shows that the healthy subject was able to achieve an RMSE of 8.68 mm against the desired trajectory in the x-direction and an RMSE of 7.02 mm against the desired trajectory in the y-direction. The subject also achieved 78% accuracy in applying a force on the end-effector that was within 20 degrees of the desired trajectory direction. This essentially means that for 78% of the trial duration, the force applied by the subject was within 20 degrees (arbitrarily chosen) of the desired trajectory direction. Note, however, that this angular tolerance value can be arbitrarily chosen in agreement with both the therapist and the patient, depending on how adaptive they want the robot controller to be for the patient. In accordance with the control scheme shown in figure 3.6 and the right side of equation 3.26, this translates into an adaptive robot controller (in this case, SMC) decreasing its applied torque by an amount defined by

$$\boldsymbol{\tau_{Subject}} = \boldsymbol{J_R}^{\boldsymbol{T}} \boldsymbol{F_I} \tag{3.34}$$

where $\boldsymbol{F_I}$ is the subject's applied interaction force on the end-effector. Note that the force sensor outputs the data in its local $X'Y'Z'$ frame. As a result, rotation transformation

was applied to acquire the global XYZ components (Figure 3.8):

$$F = [R(\theta_1)]F' \tag{3.35}$$

where $\theta_1 = \theta_{R1}$. Thus,

$$\boldsymbol{R}(\theta_1) = \boldsymbol{R}(\theta_{R1}) = \begin{bmatrix} cos(\theta_{R1}) & -sin(\theta_{R1}) \\ sin(\theta_{R1}) & cos(\theta_{R1}) \end{bmatrix} \tag{3.36}$$

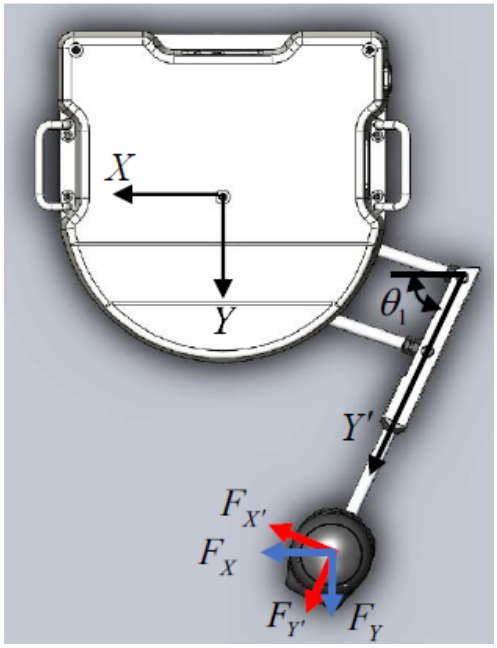

Figure 3.8: The local-global force relation [19].

To provide context to the healthy subject's 78% directional performance, below in figure 3.9 shows the time series of the applied force by the user on the end-effector in both the x- and y-directions, as well as the range of forces applied by the subject in both the x- and y-directions. In particular, the subject's range of force in the x-direction was 0.835 N and the subject's range of force in the y-direction was 2.528 N. Also note in figure 3.9 that the highlighted red lines indicate the time periods during which the subject's directional performance is low (i.e., the subject's hand is not following a direction that is within 20^o of the desired trajectory). Hence, blue indicates that the subject's hand is following a direction that is within 20^o of the desired trajectory. Judging from the performance from

figure 3.7, these are at the time periods when the subject is traversing the corners of the D shape since these are the only points when the force in the y-direction is expected to change. The plots also indicate that the point at which the worst performance occurs is the top point of the D shape in figure 3.7 because the longer red lines occur when the force transitions from the negative y-direction to the positive y-direction. The plot of the force in the x-direction also shows that the subject never applied a force on the end-effector in the positive x-direction (i.e., negative x-direction in figure 3.7). This can either mean that the robot controller may need to be improved for subsequent trials, such as increasing the tracking gains to ensure that adequate assistance is provided to accurately follow the desired trajectory, or the angular tolerance value may need to be increased, such that an environment is created where the improvement in their performance using the robot is in agreement with any improvement in their clinical performance metrics (e.g., Fugl-Meyer). A consistent improvement in performance also serves to maintain a high level of motivation in the post-stroke patient who would ideally be using the robot.

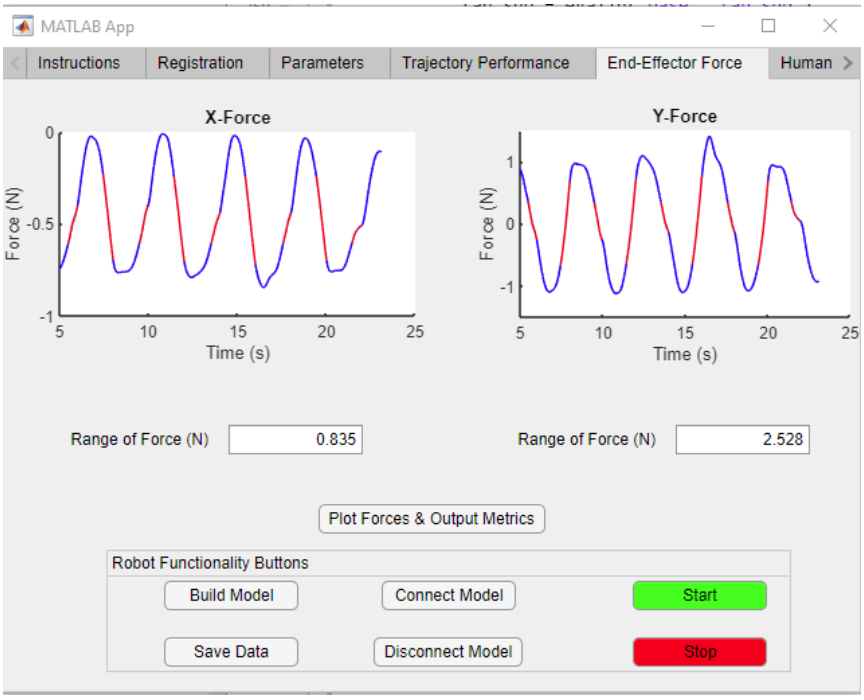

Figure 3.9: Time series of the applied force by the user on the end-effector in both the x- and y-directions.

Given a detailed muscle model, the force values of figure 3.9 may also then be correlated

against individual muscle activity levels since the dynamics of the human's arm is governed by equation 3.27. This potential capability is investigated in chapter 5.

3.6.2 Pose Estimation Results

As stated before, pose estimation is concerned with locating key points on a human's body in a video. Often, these key points represent the human body's joint locations. These key points can then be used to analytically determine the angles at these joint locations. In the context of a human subject using this rehab robot, the only human joints of concern are the subject's right elbow and shoulder.

The means through which the subject's elbow and shoulder joint angles are displayed to the subject (and therapist) is by the human joint angles tab on the GUI (Figure A.4 in appendix A). However, in keeping with the standards of the International Society of Biomechanics (ISB), the joint angles that are actually displayed by the GUI are depicted below in figure 3.10, instead of the joint angles represented by figure 3.3. Hence,

$$\theta_{s,disp} = 90^o - \theta_s \tag{3.37}$$

$$\theta_{e,disp} = \theta_e - \theta_s \tag{3.38}$$

where $\theta_{s,disp}$ is the ISB standard shoulder angle and $\theta_{e,disp}$ is the ISB standard elbow angle. As a result, figure 3.11 below shows the time series of the healthy subject's shoulder joint angle and elbow joint angle. In this tab, it shows that the healthy subject's range of motion in the shoulder joint was 42.17 degrees and the range of motion in the elbow joint was 30.65 degrees.

This type of information gives the therapist and post-stroke patient an indication of the current mobility of the patient's arm, in the absence of any pain or discomfort. If, for example, the therapist wanted to increase the subject's range of motion due to the subject's increased motor capability, the desired trajectory parameters can be adjusted accordingly (e.g., location and radius of D shape) to allow for greater mobility in subsequent trials. Thus, this is where the therapist could hypothetically use this provided information about what the subject is doing and how comfortable the subject is at performing the exercises, to ultimately tailor subsequent training exercises for each subject accordingly.

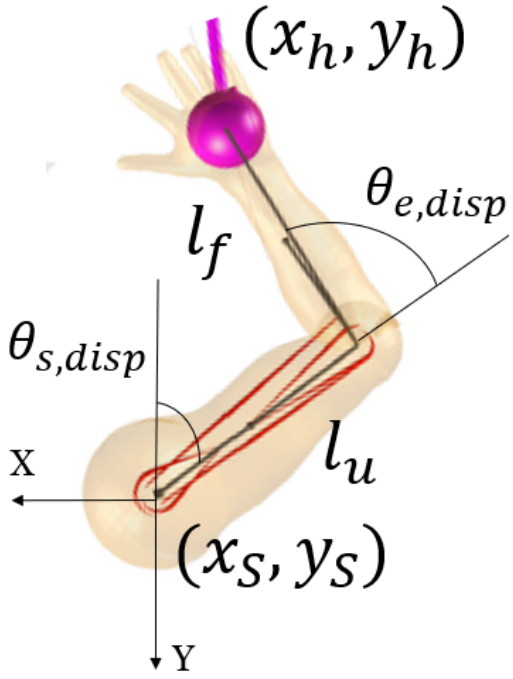

Figure 3.10: Planar human arm model (top view) with the ISB standard coordinate system (shoulder joint location), link lengths, joint angles, and hand position displayed.

3.6.3 Human Joint Torque Results

The joint torque τ_A (used in equation 3.27) that is experienced by a subject's shoulder and elbow joints can also be very indicative of a subject's motor recovery because τ_A is a function of the subject's muscle activation. If, for example, the robot requires the patient to traverse the desired trajectory at a very high speed, discomfort is likely to occur if the subject's muscles are not able to endure such strenuous movements. Thus, in the presence of a therapist, the subject's ability to endure a certain amount of torque at their shoulder and elbow joints might provide a reason for the therapist to recommend the patient traverse the trajectory at a higher speed, and see if the patient can seamlessly endure the higher experienced joint torques. This, again, highlights the use of the pHRI to tailor training exercises for each individual subject.

In regards to the healthy subject experiment with the robot, below in figure 3.12 is the human joint torques tab of the GUI displaying the raw joint torques exerted by the healthy subject's right shoulder and elbow. In this tab, it shows that the healthy subject's

Figure 3.11: Time series of the shoulder and elbow joint angles of the healthy subject.

range of torque by the shoulder joint was 4.79 N·m and the range of torque by the elbow joint was 0.817 N·m. Based on the level of comfort of the subject and expert advice from a therapist if the subject was a post-stroke patient, the goal may likely be to increase these torque values in subsequent trials to achieve motor control recovery.

The high frequency of the joint torque signals in 3.12 is due to the fact that numerical differentiation is used to obtain the robot's joint velocities and joint accelerations. If the signal were desired to be smoothened by the therapist/patient, options could include signal filtering (e.g., Butterworth), using spline fitting, or using some other curve to fit the joint torque data since smooth derivatives would likely remove the high-frequency.

3.7 Conclusion

In this chapter, we presented the rehabilitation robot being used in this study including its hardware specifications, software details, kinematic constraints, and dynamic model. We also presented the human dynamic model being used to model the subject who will

Figure 3.12: Time series of the shoulder and elbow joint torques of the healthy subject.

be interacting with the robot, including its kinematic constraints and dynamic model. We then presented the physical interaction model (pHRI), which describes how the robot and human user would physically interact with each other during a trial. A novel performance-based control scheme was introduced, which leveraged a force sensor mounted on the end-effector. The computer vision application of pose estimation was used as part of a performance validation study when compared against a regression method and a digital goniometer, such that real-time estimates of what the human was physically doing during their interaction with the robot could be reliably conducted. And finally, an experiment of this control scheme was conducted on a healthy human subject to show the potential of applying it to post-stroke subjects in a clinical setting.

Chapter 4

Using Machine Learning for Computed-Torque PID Control

4.1 Chapter Overview

As stated in section 2.5, the effectiveness of model-based control of a dynamic multibody system is reliant upon the accuracy of a model of the actual system. Computed-Torque PID control, in particular, is only applicable when an accurate enough model of the dynamic system is available. Otherwise, as the mismatch between the model of the system and the real dynamics increases, the outer-loop feedback PID term becomes increasingly crucial to correct for this increasing error. The derivation of computed-torque PID control on multibody robots is presented in the next section.

In this chapter, different machine learning control strategies are also explored. Chapter 3 aimed to derive an accurate model of the physical rehab robot system. The purpose of this chapter is to try to remove the requirement of deriving an accurate model of the physical rehab robot system, given that the input (i.e., trajectory parameters) and output (i.e., robot joint torque) data of the multibody system used to represent its interaction dynamics with a human arm model, can be reliably measured from previous trial traversals by its mounted sensors. Three machine learning architectures are explored in further detail in the following sections below.

4.2 Computed-Torque PID Control

Computed-Torque PID (CT PID) control is a special application of the feedback linearization of nonlinear systems; the nonlinear dynamics is cancelled by using its inverse dynamics in the inner-loop; the outer-loop feedback PID term is then utilized to correct the errors of the resulting decoupled linear system. The following steps derive the equation of the computed-torque control law.

If the desired trajectory $\boldsymbol{\theta_d(t)}$ is selected for the robot arm and the actual trajectory is represented by $\boldsymbol{\theta(t)}$, the tracking error is

$$\boldsymbol{e(t) = \theta_d(t) - \theta(t)} \tag{4.1}$$

By taking the first and second derivatives of the error, the following is obtained

$$\boldsymbol{\dot{e} = \dot{\theta}_d - \dot{\theta}} \tag{4.2}$$

$$\boldsymbol{\ddot{e} = \ddot{\theta}_d - \ddot{\theta}} \tag{4.3}$$

According to the feed-forward equations of 3.6 and 3.20, $\ddot{\theta}$ can then be given by

$$\boldsymbol{\ddot{\theta} = M(\theta)^{-1}(\tau - C(\theta, \dot{\theta})\dot{\theta} - f)} \tag{4.4}$$

where $\boldsymbol{f}$ can be friction or stiffness terms. Substituting equation 4.4 into equation 4.3 results in

$$\boldsymbol{\ddot{e} = \ddot{\theta}_d + M(\theta)^{-1}(C(\theta, \dot{\theta})\dot{\theta} + f - \tau)} \tag{4.5}$$

From equation 4.5, the joint torque $\boldsymbol{\tau}$ can be given by

$$\boldsymbol{\tau = M(\theta)(\ddot{\theta}_d - \ddot{e}) + C(\theta, \dot{\theta})\dot{\theta} + f} \tag{4.6}$$

Since the goal is to control and minimize $\boldsymbol{\ddot{e}}$, the feedback control signal can be a PID term as such

$$\boldsymbol{u_{PID} = -K_p e - K_i \int_0^t e(t)\, dt - K_d \dot{e}} \tag{4.7}$$

By substituting equation 4.7 into equation 4.6, the computed joint torque signal $\boldsymbol{u}$ becomes

$$\boldsymbol{u} = \boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{\theta})(\ddot{\boldsymbol{\theta}}_{\boldsymbol{d}} + \boldsymbol{K_p e} + \boldsymbol{K_i} \int_0^t \boldsymbol{e}(\boldsymbol{t})\, \boldsymbol{dt} + \boldsymbol{K_d}\dot{\boldsymbol{e}}) + \boldsymbol{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{f} \tag{4.8}$$

Figure 4.1 is a diagram of the outlined Computed-Torque PID control scheme, which takes in the desired trajectory parameter $\boldsymbol{\theta_{R_d}}$ as input, features the feed-forward inverse dynamics block to compute $\boldsymbol{\tau}$ or equivalently $\boldsymbol{u_{Model}}$, and features the feedback PID control block to compute $\boldsymbol{u_{PID}}$, all eventually being super-imposed to compute the final control signal $\boldsymbol{u(t)}$.

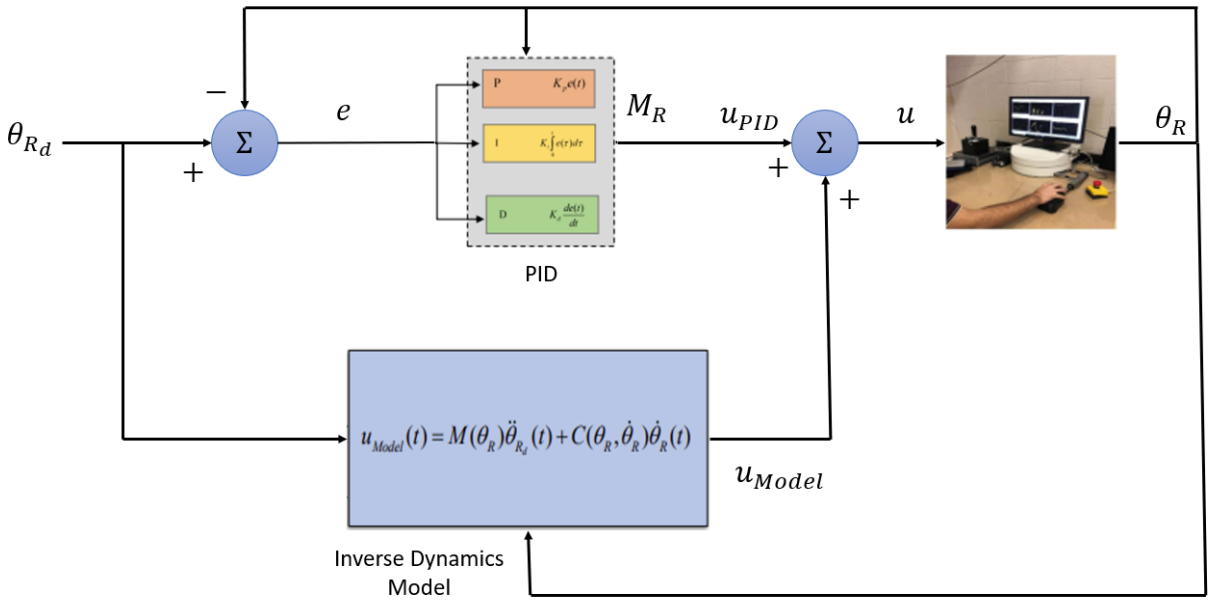

Figure 4.1: Computed-Torque PID block diagram.

4.3 Training Data

Three experiments were conducted on the robot by one healthy subject, in which three reference trajectories with different trajectory locations were set, and consequently traversed by the robot, while controlled by an SMC controller [19]. Data acquired from each

experiment included the joint angles, joint velocities, desired joint accelerations, and total applied joint torques (i.e., including the performance-based torque contribution from the force sensor) by the robot's SMC controller. The three reference trajectories, as well as the positional and directional performance of the robot end-effector following these trajectories, are shown in the trajectory performance tab of the GUI (Figures 4.2-4.4).

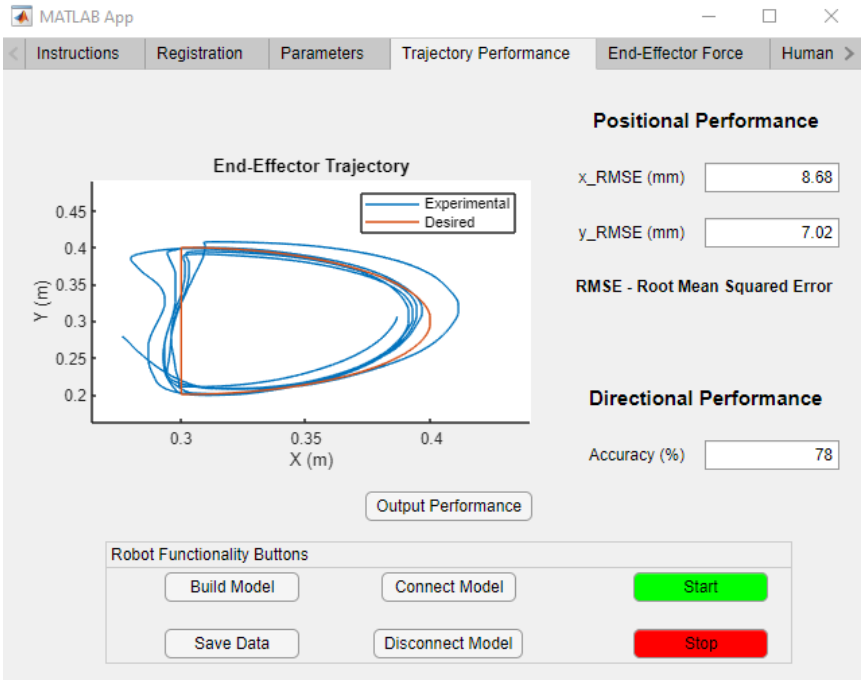

Figure 4.2: First reference trajectory.

Testing the trained networks on newly implemented computed torque PID controllers deployed on the physical robot could damage the physical system if the model represented by the neural network is severely inaccurate. Thus, the goal is to use the data from one of these three trajectory experiments as training data for each of the neural networks and then test how the trained neural networks performed on the other two trajectory datasets. The results of these simulations are described in the following section.

4.4 Simulations Results

The following subsections describe in detail the accuracy of the neural networks in modelling the so-called "actual" dynamics of the multibody robot, by means of how accurate

Figure 4.3: Second reference trajectory.

these controllers are in following the desired trajectories, and in the presence of a healthy subject applying a 2D force on the end-effector.

4.4.1 Deep Lagrangian Network (DeLaN)

The implementation of a DeLaN on this robot proved to not be successful because the joint torque signals applied to the robot joints were not super-imposed signals that were functions of the joint angles, joint velocities, and desired joint accelerations, as outlined by equation 3.26 (including the performance-based contribution of the force sensor). Thus, the relationship between the joint angles, joint velocities, desired joint accelerations, and joint torques is as follows:

$$\boldsymbol{M_R\ddot{\theta}_{R,d} + C_R\dot{\theta}_R + K_R(\theta_R - \theta_{R_0}) + J_R{}^T f_{RE} + f_{RJ} = \tau_R} \tag{4.9}$$

To prove that the kinematic data measured by the robot's motor encoders did not satisfy this relationship, let us first assume that the Mass matrix and Coriolis matrix are

Figure 4.4: Third reference trajectory.

those stated in equations 3.7 and 3.8 since these were validated by previous researchers, and the real/actual parameter values, are likely not very different. In doing so, the only terms of equation 4.9 to be modelled are the last three on the left side of the equation, namely, the robot stiffness, end-effector friction, and joint friction. Figures 4.5 and 4.6 show the contribution of the super-imposed Mass torque and Coriolis torque to the robot joint torques for the first trajectory (Figure 4.2).

It is evident from figures 4.5 and 4.6 that if equation 4.9 does in fact describe the dynamics of the actual rehab robot, it can be deduced that the majority of the torque contribution is from the robot stiffness, end-effector friction, and joint friction. If the robot stiffness, end-effector friction, and joint friction can be modelled as functions of the input variables of the joint angles, joint velocities, and desired joint accelerations, then it can be reliably concluded that a DeLaN should enable an accurate controller. This is tested by training a simple feed-forward neural network on the first trajectory (Figure 4.2) with the input variables being the joint angles, joint velocities, and desired joint accelerations, and the output variable being the total applied robot joint torques minus the Mass and Coriolis torque contributions. Below in figures 4.7-4.9 shows the accuracy of the feed-forward neural

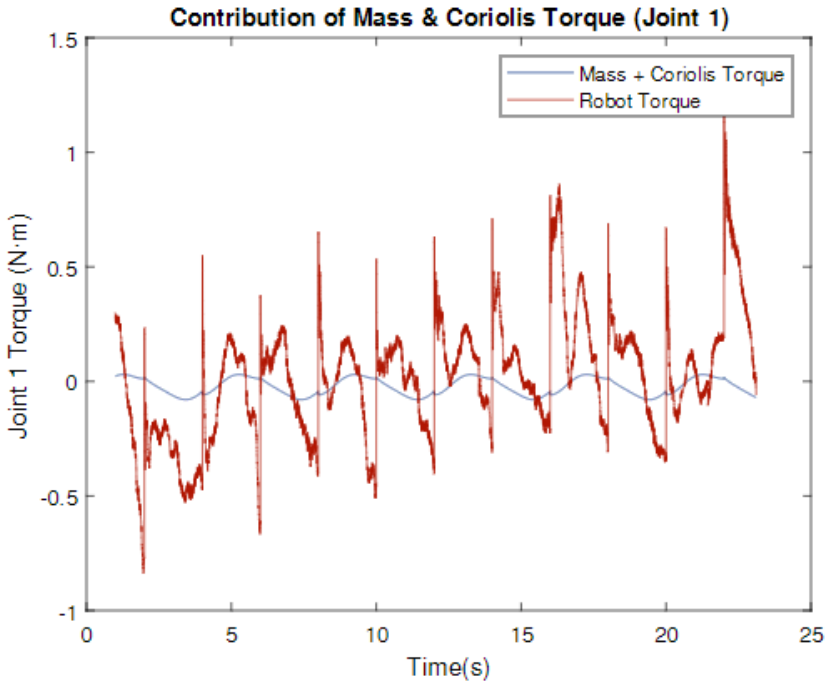

Figure 4.5: Joint 1: Mass Torque + Coriolis Torque.

network on all three trajectories.

It is evident from figures 4.7-4.9 that the simple feed-forward neural network expectedly performed really well on the first trajectory's total applied robot torques minus the Mass and Coriolis torques since it was trained on this data, but it performed really poorly on the second and third trajectories' total applied robot torques minus the Mass and Coriolis torques. Therefore, the implementation of a DeLaN on the physical robot would not be very successful because the dynamics governed by equation 4.9 are never learned.

4.4.2 Feed-forward Neural Network (FNN)

Similar to how an FNN could not provide a general model of the robot's stiffness and friction torque terms in the last subsection, an FNN also cannot provide a general dynamic model of the actual robot. Another simple FNN is trained on the first trajectory (Figure 4.2) with the input variables being the joint angles, joint velocities, and desired joint accelerations, and the output variable is the total applied robot joint torques. Below in figures 4.10-4.12

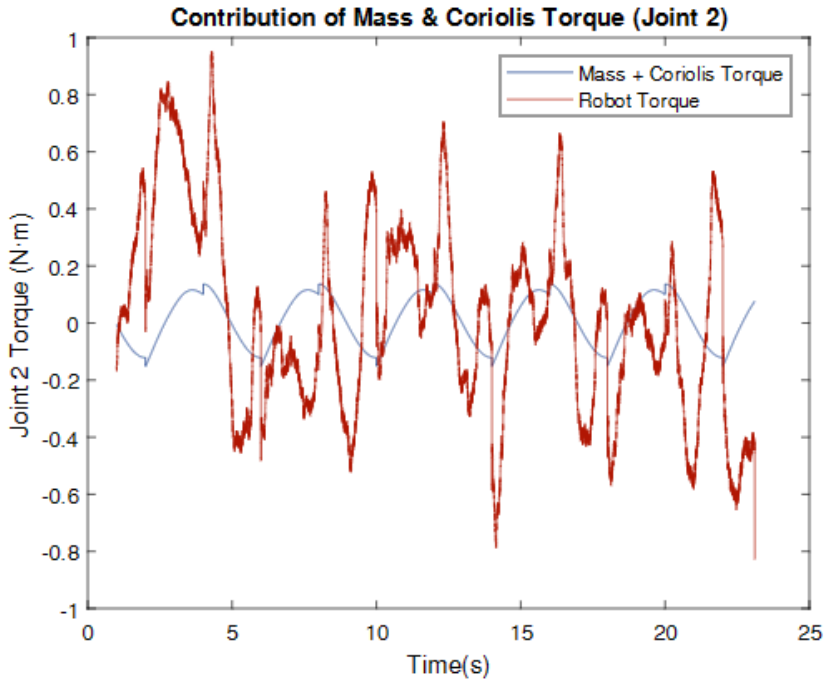

Figure 4.6: Joint 2: Mass Torque + Coriolis Torque.

shows the prediction performance of the FNN on all three trajectories.

It is evident from figures 4.10-4.12 that the FNN performed well on the first trajectory since it was trained on this data, but it performed poorly on the second and third trajectories. Hence, the training of an FNN on robot kinematic inputs to predict the torque applied to this physical robot proved to be unsuccessful. For an FNN model to be generalizable and thus be applied to all desired trajectory parameters, this would require the inclusion of a data set that would ideally encompass the entire end-effector workspace being spanned at multiple velocities and accelerations. However, this is infeasible due to the very high number of combinations of desired trajectory parameters (i.e., different trajectories being traversed at different speeds and accelerations) that would have to be traversed in order to generate the necessary training data. It may be stated that the shape of the predicted and actual torque signals are similar with only some scaling needed, but the inclusion of more training data may help to incorporate this necessary scaling factor.

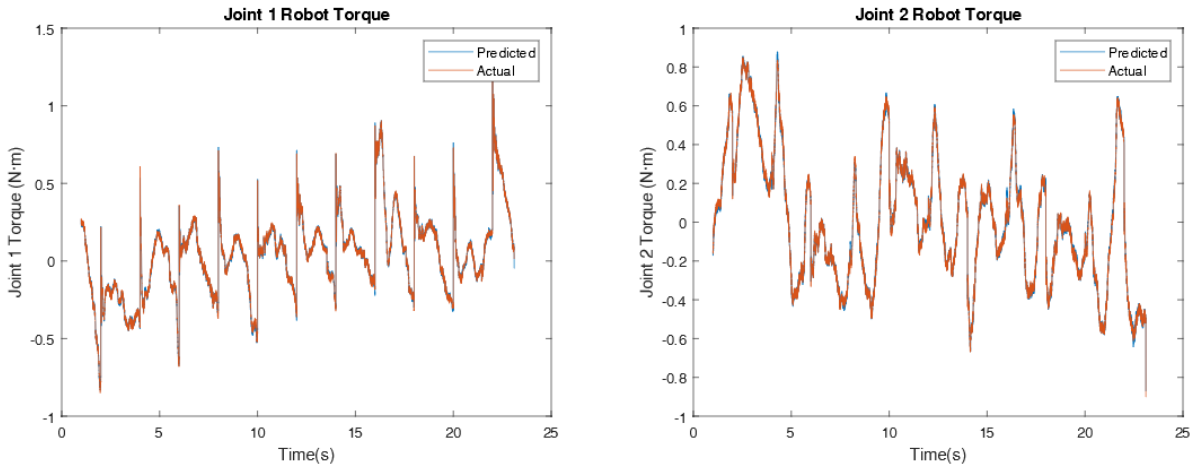

Figure 4.7: FNN performance on the first trajectory, when trained on the first trajectory and the output set as the total applied robot torques minus the Mass and Coriolis torques.

4.4.3 Recurrent Neural Network (RNN)

An RNN may produce the same inaccuracies as a shallow FNN due to the fact that the source of the problems encountered by the FNN in the previous subsection was the data itself, and not the network architecture. But another problem with using an RNN was encountered. A nonlinear autoregressive with external input (NARX) network in MATLAB was used to train on the data of the first trajectory. Due to the fact that the robot's control algorithms are run on MATLAB/Simulink for Quarc, one of the requirements to use a trained neural network in Simulink, including a simple shallow FNN, is that a Simulink block with a sample time must be generated for the shallow neural network simulation. This is achieved by MATLAB's *gensim* command. The sample time for dynamic networks such as a NARX is required by MATLAB to be 1 second or greater due to unknown reasons, while the sample time of the robot is set to the recommended 0.002 seconds. Designing an accurate trajectory controller with a minimum sample time of 1 second would likely be difficult when the joint angles change considerably every 0.002 seconds, as shown by simulations.

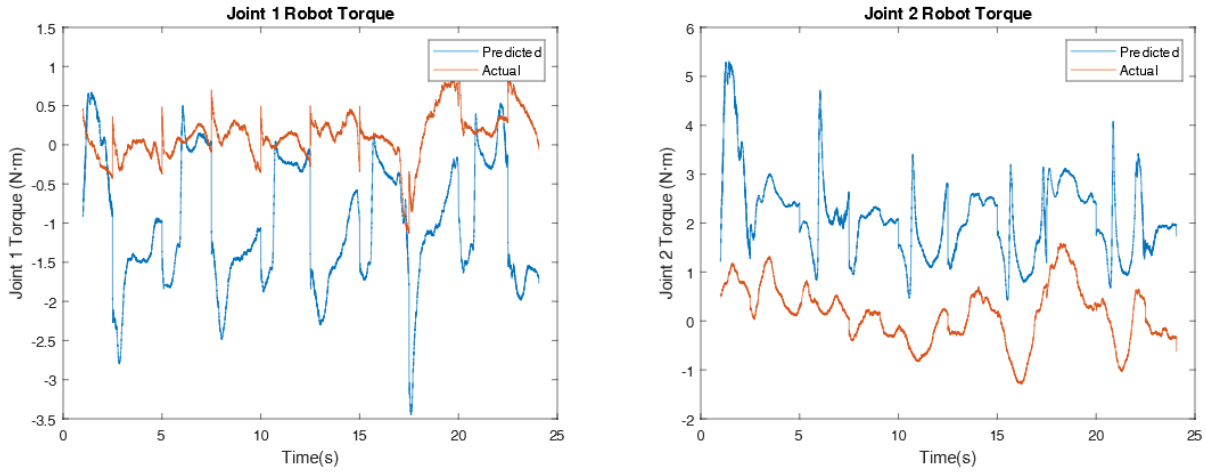

Figure 4.8: FNN prediction performance on the second trajectory, when trained on the first trajectory and the output set as the total applied robot torques minus the Mass and Coriolis torques.

4.5 Conclusion

In this chapter, we presented the training data used for training the neural networks of a DeLaN, FNN, and RNN, and then we presented the simulation results of these networks when implemented on the data gathered from the actual rehabilitation robot. It was concluded that none of these neural network architectures provided any reliable generalization capability to be deployed on the actual rehabilitation robot since this would likely result in spasmodic robot motions, robot motor damage, and potentially further injuries to the post-stroke patient.

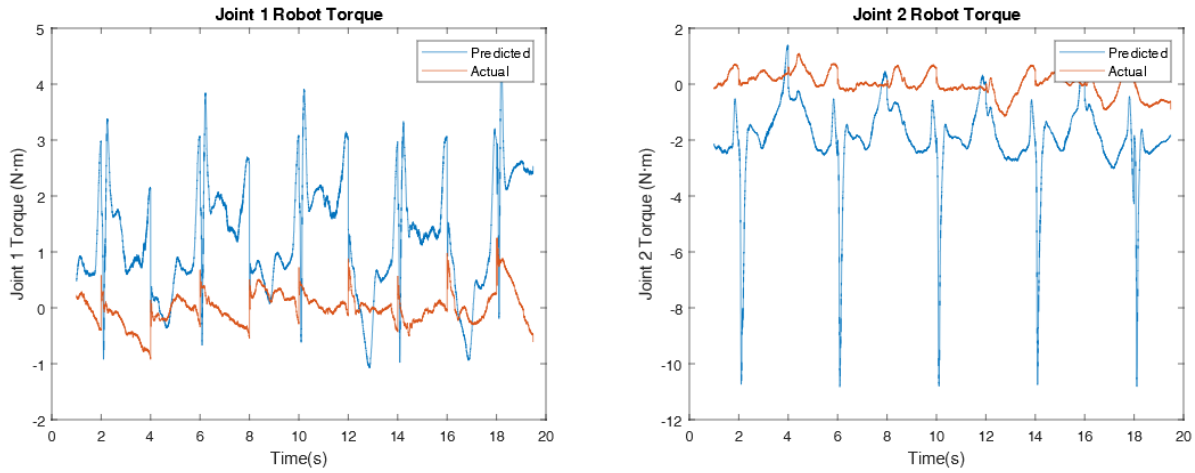

Figure 4.9: FNN prediction performance on the third trajectory, when trained on the first trajectory and the output set as the total applied robot torques minus the Mass and Coriolis torques.

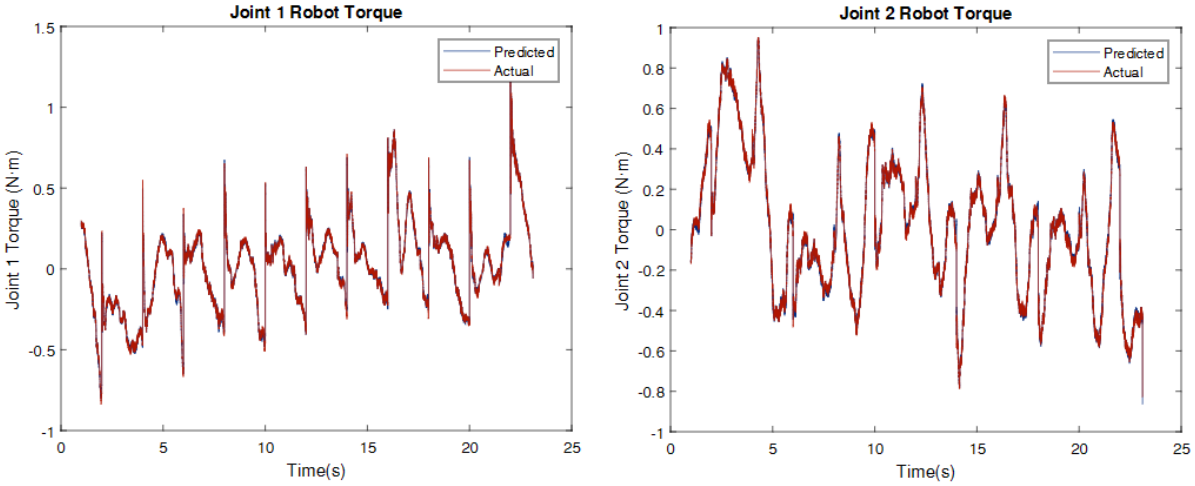

Figure 4.10: FNN performance on the first trajectory, when trained on the first trajectory and the output set as the total applied robot torques.

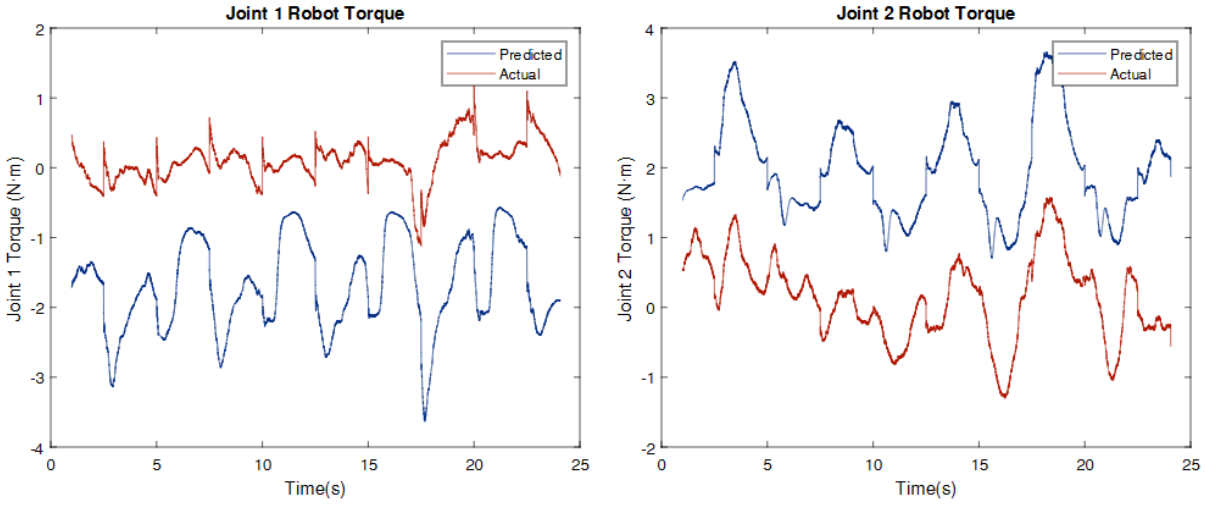

Figure 4.11: FNN prediction performance on the second trajectory, when trained on the first trajectory and the output set as the total applied robot torques.

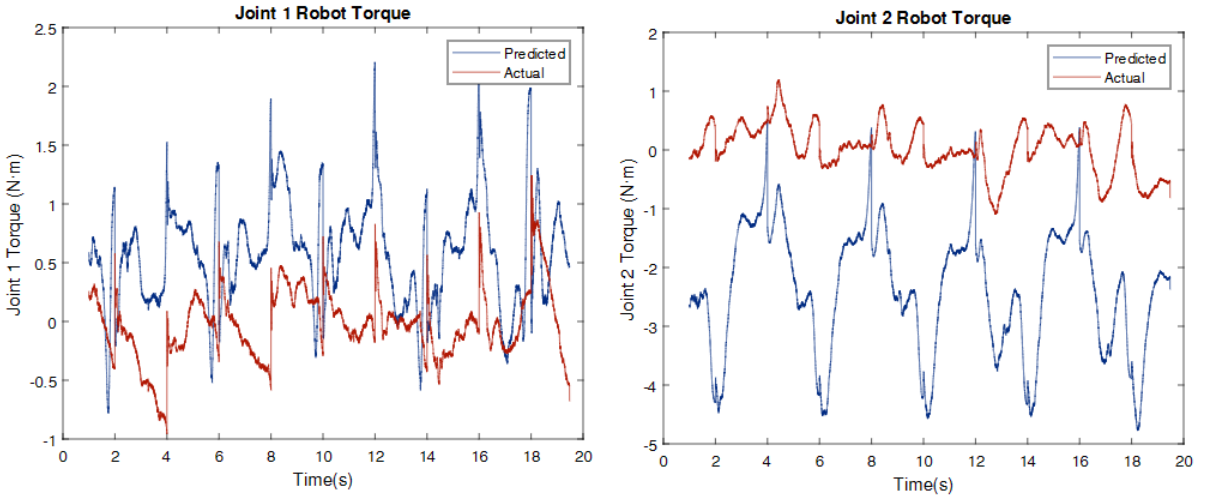

Figure 4.12: FNN prediction performance on the third trajectory, when trained on the first trajectory and the output set as the total applied robot torques.

Chapter 5

EMG-Based Assistive Control

5.1 Control Scheme Overview

In this chapter, experiments were conducted on four healthy subjects in an effort to determine whether Surface Electromyography (sEMG) data can be used to predict the subject's joint torques at the elbow and shoulder. sEMG measures the electric potential difference triggered by the human nervous system to command muscle contraction. In doing so, it has demonstrated capability in illuminating human motion intention for controlling biomechatronic devices as well as being used to identify aberrant neuromuscular functions [32]. One or more probes, commonly known as electrodes, are initiated over the desired muscle. A base recording unit is then used to display the electrical activity that the electrodes pick up (a monitor that amplifies and displays electrical activity in the form of waves). The electrical activity of muscles during rest, a light contraction, and a strong contraction can all effectively be measured by EMG.

The subject-specific performance-based control scheme discussed in chapter 3 uses a pHRI model (i.e., force sensor) to adjust the robot's applied torque when the subject is applying a force in a direction that is within a tolerance value of the desired direction at a certain time instant. Research has already widely been conducted on EMG-based assistive control when a subject is using a planar end-effector-based robot for the upper extremity [6, 8, 47]. What hasn't been widely explored, however, is predicting the applied force direction on the robot end-effector using EMG. This prediction model, which defines the newly formed cognitive Human-Robot Interaction (cHRI) model, can then be used to adjust the robot's applied joint torques (i.e., level of assistance) in accordance with the subject-specific performance-based control scheme of chapter 3, but without a pHRI model.

It was mentioned in subsection 2.3 that EMG tends to not be as well-indicative of post-stroke patients' movement intentions as it can be for predicting healthy subjects' intentions. This is due to the fact that improper placement of electrodes on post-stroke patients who are often overweight is much more prevalent, post-stroke patients often have higher movement variability when using a robotic device due to the inter-subject variability in neuromuscular system damage, and there exists a higher intra-subject variability of muscle activation patterns amongst post-stroke patients, all of which makes it more difficult for robot controllers to accurately predict a post-stroke subject's motion intention [6, 8]. Hence, only healthy subjects were recruited for conducting experiments involving sEMG and the rehab robot, whereas using the force sensor as part of a pHRI model would ideally be designated for post-stroke patients.

The other reason for exploring this potential capability in predicting a subject's directional performance via the force sensor is that most planar end-effector-based rehab robots currently being used in clinical environments are not equipped with mounted force sensors on the end-effector. As is the case with the planar end-effector-based rehab robot used in this project, force sensors are often an augmentation to these types of robotic devices. Given the already widespread use of EMG in laboratory settings and despite its minor drawbacks, its ability to measure motion intent in healthy subjects [6, 8, 47] is a motivating factor in predicting the subject's joint torques, such that subject-specific active control can be implemented without using a force sensor. In the following sections, details about the experiments conducted as well as the results will be discussed.

5.2 Experiments

Equation 3.27 is repeated below as equation 5.1, but with the arm joint torque τ_A term explicitly expressed as a function of muscle activation u_A. This is because the torque exerted by the subject's right arm during their interaction with the rehab robot is hypothesized to be a function of only the activation level exerted by the muscle groups that are responsible for moving the subject's right arm, and which was also used in simulations by [19]. The process that is often used to measure a muscle group's activation level is by initially measuring the subject's raw EMG signal, and then normalizing this raw EMG signal by the subject's maximum voluntary contraction (MVC) of the desired muscle group. The reason for normalizing the EMG data using the specific subject's MVC is to be able to reliably compare EMG activity in the same muscle group between different days, between different individuals (e.g., healthy subject vs. post-stroke patient), and between different muscle groups. The reliable comparison of EMG activity in the same muscle group between

different days also requires the collection of lots of data to address intra-subject variability.

$$M_A\ddot{\theta}_A + C_A\dot{\theta}_A = \tau_A(u_A) + {J_A}^T F_I \tag{5.1}$$

A traditional way to convert normalized EMG signals into the appropriate muscle joint torques τ_A is to use a biomechanical muscle model. The most commonly used model is the three-component model, which takes inspiration from the lumped-parameter model developed by Hill for active and passive muscle tension behavior [51]. However, including a general muscle model within a multibody model introduces various drawbacks including muscle redundancy, specified complex musculoskeletal geometry such as wrapping pathways, difficult-to-fit parameters for each muscle, and parameter sensitivity. One method that may serve to alleviate the complex muscle geometry, redundancy, and interpretation of EMG signals within a control framework could be a machine learning model (i.e., neural network) trained by experimental data.

Therefore, the inputs of a neural network to predict the human joint torque as calculated using equation 5.1, was only the normalized EMG signals (via dividing by the subject's isometric MVC) to account for the muscle activations (u_A). The muscle groups of interest that were selected to be measured during these experiments were the right arm's biceps brachii, triceps brachii, and deltoid. Hence, the neural network consists of three inputs representing the three EMG signal variables; and two outputs representing the two human joints of the elbow and shoulder.

The sample size determined to be sufficient to assess the accuracy of the human joint torque predictability scheme using EMG data was set to four subjects. After analyzing the normalized EMG data of these participants, this sample size was deemed sufficient to assess whether a diverse range of normalized EMG data could accurately predict the joint torque of a human subject using the rehab robot. The following subsections discuss the procedure followed to conduct these experiments as well as the results obtained.

5.2.1 Procedure

The procedure encompassed two steps of data collection:

1. Pre-treatment assessments,
2. Robotic treatment assessments.

Pre-treatment Assessments

After receiving informed consent, the subject's general health information was reviewed by the researcher in a meeting with the subject. The collected data included:

- Mass
- Upper extremity anthropometric data: upper-arm and forearm lengths

Robotic Treatment Set-up and Assessment

The researcher attached the EMG probe (Trigno Wireless System, Delsys incorporated, MA, USA) to the user's right arm. Recording the participant's motion and myoelectric (majority of the study visit) would be done at a 3-meter distance from the researcher. There was no overlap between participants (1 participant per day).

The skin overlying the participant's target muscles was cleaned before the researcher attached self-adhesive EMG electrodes over the muscles of interest. After the preparation, the researcher attached the electrodes to the skin overlying each muscle of interest, which were the right arm's bicep brachii, tricep brachii, and deltoid.

Each of the four subjects performed two sessions of trials. Each subject's isometric maximum voluntary contraction (MVC) from a maximum root mean square (RMS) envelope of the EMG signal using a moving window was recorded for each session. The MVC value was then used to normalize the EMG signals collected during the session's trials. Note that for the normalization procedure, the RMS window length was set to 0.125 seconds and the RMS window overlap was set to 0.0625 seconds. The output was displayed as a percentage of the MVC value. For the first session, the participant moved the robot's end-effector along a desired D-shaped trajectory with no robot assistance or resistance for two trials. For the second session, the participant moved the robot's end-effector along the same desired D-shaped trajectory with no robot assistance or resistance for only one trial. Thus, there were 12 trials altogether.

5.2.2 Results

Once all of the necessary trial data was obtained from the subject experiments, only a feed-forward neural network (FNN) was trained on the data due to the reasons described in subsection 4.4.3 pertaining to using RNNs in MATLAB/Simulink. The data was split

such that the training set contained 11 trials and the testing set contained the fourth subject's trial during their second session.

Many feed-forward neural network architectures were not able to achieve acceptable results on the testing data set when trained on the 11 trials that comprised the training data set. Hence, an investigation into why the three muscle activation signals could not be successfully mapped to the elbow and joint torques was conducted. Below in figures 5.1-5.2 are six scatter plots that depict the relationship between each of the muscle activation signals and the joint torques.

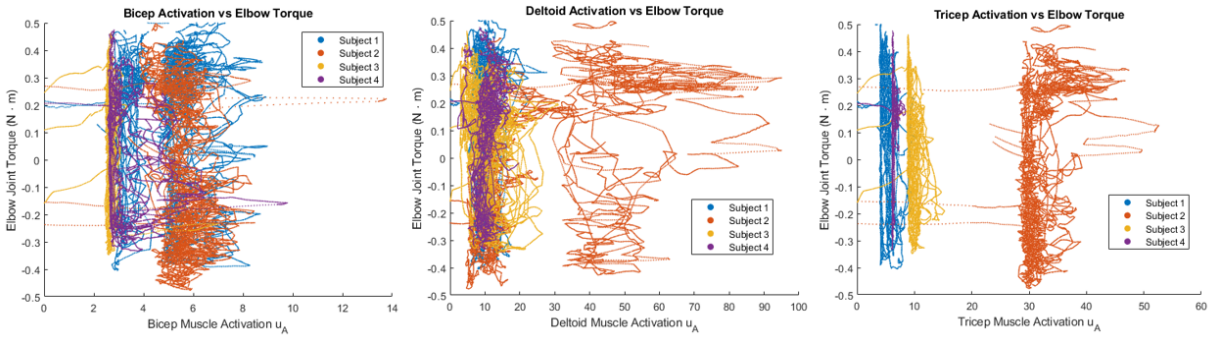

Figure 5.1: Elbow Torque vs. Muscle Activations.

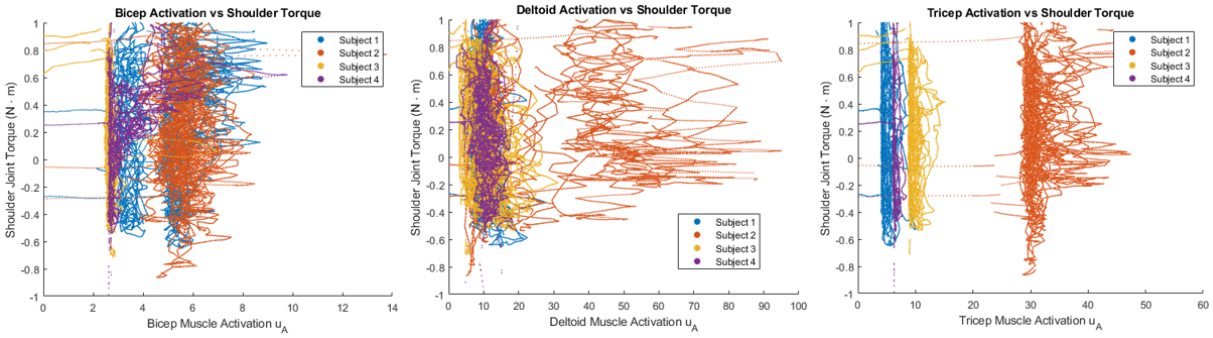

Figure 5.2: Shoulder Torque vs. Muscle Activations.

It is clear from the vertical clusters in figures 5.1-5.2 that muscle activations of relatively equal value correspond to elbow and shoulder joint torque values that can span the entire spectrum range of all four subjects. This explains why different neural network architectures are unable to accurately predict the joint torques given the muscle activations as the sole input. This leads to the conclusion that for this particular application of the

rehab robot, experimental results show that using muscle activations as an indicator for predicting the subject's joint torques to eventually replace the force sensor, is not reliable. A possible reason why there is not a specific 1-to-1 mapping from EMG to joint torque might be the fact that not enough muscles were probed for electrical activity during the exercises. For Nasr's MuscleNet [32], surface EMG signals were measured from 11 sites over muscles of the right upper limb: the Serratus Anterior (SERR), Middle Deltoid (MDEL), Supraspinatus (SUPR), Infraspinatus (INFR), Posterior Deltoid (PDEL), Pectoralis Major (PECC), Latissimus Dorsi (LATS), Anterior Deltoid (ADEL), Middle Trapezius (MTRA), Upper Trapezius (UTRA), and Lower Trapezius (LTRA). Delsys Trigno's EMG software did not provide these specific muscle groups, but having the muscle activity of these specific muscle groups might have produced a 1-to-1 mapping from EMG to joint torque.

One might also be curious about subject 2's large range of muscle activation, especially in their deltoid while even reaching up to about 90% muscle activation as subject 2 traversed the desired D-shaped trajectory. This anomaly is most likely attributed to recording an erroneous MVC for the subject at the beginning of the session, in which subject 2 likely did not exert their full muscle contraction in their deltoid.

5.3 Conclusion

In this chapter, we presented a cognitive human-robot interaction model (cHRI), in which measuring a subject's muscle activation via sEMGs was hypothesized to predict the elbow and shoulder joint torques exerted by the subject. This would then enable the prediction of the subject's contribution by using equation 5.1 to calculate the force on the end-effector F_I, such that the novel performance-based control scheme introduced in chapter 3 could then be used. Unfortunately, many neural network architectures were not able to achieve acceptable results on the testing data trial when trained on the 11 trials that comprised the training data set. Experimental results showed that using muscle activations as an indicator for predicting the subject's joint torques to eventually replace the force sensor, was deemed not reliable.

Chapter 6

Conclusion and Future Work

6.1 Thesis Summary

This thesis introduced a novel control scheme for a stroke rehabilitation robot to enable more subject-specific adaptability, examined the role of machine learning in identifying a rehab robot's unknown dynamic model for model-based control, and investigated whether combining biomechanical data with EMG data could accurately predict a user's applied force on the robot's end-effector since many rehab robots already on the market are not equipped with a force sensor on the end-effector.

To make the control scheme subject-specific, the goal for this robot was to adjust its joint torque contribution in accordance with the user's force-directional performance. In particular, two performance metrics were used: (1) the root mean squared error against the desired trajectory in order to measure the user's positional accuracy, and (2) the percentage of the trial duration during which the user's applied force on the robot's end-effector stayed within a small tolerance of the direction of the trajectory to measure directional performance. These two performance metrics contribute to the subject-specificity that is used by the robot controller to adapt its applied torque values in real-time. The potential of using this control scheme was showcased on a healthy subject. The robot was initially intended to be used on post-stroke patients, such that this novel subject-specific performance-based control scheme could be tested over a longer duration of time to see if any correlations between the performance metrics described and standard clinical metrics could be drawn. But COVID and other reasons prevented us from testing on post-stroke patients.

Model-based control of a dynamic multibody system is reliant upon the accuracy of a model of the physical system. Computed-Torque control, in particular, is only applicable

when a perfect (or accurate enough) model of the dynamic system is available. Otherwise, as the mismatch between the model of the system and the real dynamics increases, the outer-loop feedback (e.g., proportional-integral-derivative (PID) control) term becomes increasingly crucial to correct for this increasing error. Different machine learning control strategies were explored to try to remove the requirement of finding an accurate model of a dynamic system, given that the input and output data of the multibody system used to represent its dynamics, could reliably be measured from previous trials using sensors on the robot. However, the results of simulating these machine-learning control strategies (DeLaN, FNN, RNN) to this particular rehab robot were not promising.

A cognitive human-robot interaction model (cHRI) between the rehab robot and a subject user was presented since it was hypothesized that a subject's muscle activations (bicep brachii, tricep brachii, and deltoid) measured via sEMGs could map to the elbow and shoulder joint torques exerted by the subject. A traditional way to convert normalized EMG signals into the appropriate muscle joint torques τ_A is to use a biomechanical muscle model, such as the three-component model by A.V. Hill. However, including a general muscle model within a multibody model introduces various drawbacks including muscle redundancy, complex musculoskeletal geometry such as wrapping pathways, difficult-to-fit parameters for each muscle, and parameter sensitivity. One method to alleviate the complex muscle geometry, redundancy, and interpretation of EMG signals within a control framework was a machine learning model (i.e., neural network) trained by experimental data. Reliant upon accurate predictions of the subject's elbow and shoulder joint torques, this would enable the prediction of the subject's force contribution on the robot's end-effector, such that the novel subject-specific performance-based control scheme already introduced could be used. However, many neural network architecture configurations were not able to achieve acceptable results on the testing data trial when trained on the 11 trials that comprised the training data set. Experimental results showed that using muscle activations as an indicator for predicting the subject's joint torques to eventually replace the force sensor, was deemed not reliable.

6.2 Recommendations and Future Work

The following topics are recommended for future research:

- There were some limitations encountered with using Quanser's Quarc real-time control software, which is a MATLAB software that generates real-time code directly from Simulink-designed controllers and runs it in real-time on the Windows target.

Using its library of function blocks, Quarc is able to capture video images from a device, such as an iPhone camera. However, Quarc is unable to conduct real-time pose estimation on these captured images because Simulink is unable to generate stand-alone C/C++ code for pose estimator models written in both MATLAB and Python. It is recommended that in future studies of post-stroke robot-assisted rehabilitation, research should be conducted in partnership with Quanser about designing a new Simulink function block that performs pose estimation, such that a trained and accurate pose estimation model is embedded into a new function block that is part of Quarc's overall block library.

- The default sample time of the control structure in Quarc is 0.002 seconds. The iPhone camera that was intended to be used as part of the stroke patient trials limited the video frame rate to 30 frames per second. Due to these differences in sample rates between the robot and the video capture, 1D interpolation would have to be performed on the pose estimation estimates of the human joint angles, such that the sample rate would match the rate of the robot. Other metrics such as the patient's exerted joint torques could also be calculated for each 0.002-second time step. It is recommended that in future studies of post-stroke robot-assisted rehabilitation, higher frame rate cameras should be used, such that the camera could capture images at a frame rate of at least 500 frames per second (1/0.002 seconds).

- The calculated torque values displayed in figure 3.12 are inherently estimates of the subject's joint torque when using the rehab robot. To ensure that the values displayed are accurate, it's recommended that the protocols of potential future studies with this rehab robot incorporate a requirement for the subject to wear an upper limb exoskeleton that can measure torque via a torque sensor. This way, the muscular torque of the subject can be estimated by first measuring the applied external torque at each joint of the exoskeleton, and then removing the inertial, Coriolis, and gravitational torque contributions from the subject's upper limb.

References

[1] O.I. Abiodun, A. Jantan, A.E. Omolara, K.V. Dada, N.A. Mohamed, and H. Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):200–230, 2018.

[2] O.I. Abiodun, A. Jantan, A.E. Omolara, K.V. Dada, N.A. Mohamed, and H. Arshad. Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, 3(2):334–340, 2018.

[3] R. Bertani, C. Melegari, C. Maria, A. Bramanti, P. Bramanti, and R.S. Calabro. Effects of robot-assisted upper limb rehabilitation in stroke patients: a systematic review with meta-analysis. *Neurological Sciences*, 38(9):1561–1569, 2017.

[4] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. *2009 IEEE 12th International Conference on Computer Vision*, pages 1365–1372, 2009.

[5] P. Brown and J. McPhee. A continuous velocity-based friction model for dynamics and control with physically meaningful parameters. *Journal of Computational and Nonlinear Dynamics*, 11(5):1–6, 2016.

[6] B. Cesqui, P. Tropea, S. Micera, and H.I. Krebs. Emg-based pattern recognition approach in post stroke robot-aided rehabilitation: a feasibility study. *Journal of NeuroEngineering and Rehabilitation*, 10(75), 2013.

[7] B.C. Csáji. Approximation with artificial neural networks. *Master's Thesis*, page 11, 2001.

[8] L. Dipietro, M. Ferraro, J.J. Palazzolo, H.I. Krebs, B.T. Volpe, and N. Hogan. Customized interactive robotic treatment for stroke: Emg-triggered therapy. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13(3):325–334, 2005.

[9] S. Dupond. A thorough review on the current advance of neural network structures". annual reviews in control. *Annual Reviews in Control*, 14:200–230, 2019.

[10] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[11] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[12] Centers for Disease Control and Prevention. About stroke, 2022.

[13] B. Ghannadi. Model-based control of upper extremity human-robot rehabilitation systems. *UWSpace*, 2017.

[14] B. Ghannadi, R. Sharif Razavian, and J. McPhee. Upper extremity rehabilitation robots: A survey. *Handbook of Biomechatronics*, page 319, 2018.

[15] I. Goodfellow, Y. Bengio, and A. Courville. 6.5 back-propagation and other differentiation algorithms. *Deep Learning. MIT Press.*, page 200–220, 2016.

[16] C. Gowland, P. Stratford, M. Ward, J. Moreland, W. Torresin, S. Van Hullenaar, J. Sanford, S. Barreca, B. Vanspall, and N. Plews. Measuring physical impairment and disability with the chedoke-mcmaster stroke assessment. *Stroke*, 24:58–63, 1993.

[17] CSA Group. http://shop.csa.ca/en/canada/applications-of-electricity-in-\health-care/cancsa-iso-14971-07-r2012/invt/27016072007, 2023.

[18] M.P. Gunasekara, R.C. Gopura, T.S.S. Jayawardane, and S.W.H.M.T.D. Lalitharathne. Control methodologies for upper limb exoskeleton robot. *IEEE/SICE International Symposium on System Integration*, pages 19–24, 2012.

[19] A. Hashemi. Trajectory planning and subject-specific control of a stroke rehabilitation robot using deep reinforcement learning. *UWSpace*, 2021.

[20] E. Haug. Computer aided kinematics and dynamics of mechanical systems. *Allyn and Bacon Series in Engineering*, 1:64–68, 1989.

[21] Heart and Stroke Foundation of Canada. Arms and legs. https://www.heartandstroke.ca/stroke/recovery-and-support/physical-changes/arms-and-legs, 2023.

[22] Z.-G. Hou. Passive and active control for rehabilitation robotics. https://ewh.ieee.org/conf/wcci/2016/document/tutorials/ijcnn6.pdf.

[23] S. Huang, S. Cai, G. Li, Y. Chen, and L. Xie. Variable robot-resistance rehabilitation for upper limb based on an semg-driven mode. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 814–818, 2019.

[24] K.A. Inkol, C. Brown, W. McNally, C. Jansen, and J. McPhee. Muscle torque generators in multibody dynamic simulations of optimal sports performance. *Multibody System Dynamics*, page 435–452, 2020.

[25] S. Johnson and M. Everingham. Learning effective human pose estimation from inaccurate annotation. *CVPR 2011*, pages 1465–1472, 2011.

[26] P. Khoshroo. Control system and graphical user interface design of an upper-extremity rehabilitation robot. *UWSpace*, 2020.

[27] H.I. Krebs. Twenty + years of robotics for upper-extremity rehabilitation following a stroke. *Elsevier Ltd.*, 2018.

[28] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *ICLR Conference*, 2019.

[29] L. Marchal-Crespo and D.J. Reinkensmeyer. Review of control strategies for robotic movement training after neurologic injury. *Journal of NeuroEngineering and Rehabilitation*, 6(1):1–15, 2009.

[30] W. McNally, K. Vats, A. Wong, and J. McPhee. Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation, 2022.

[31] L.M. Mooney, E.J. Rouse, and H.M. Herr. Autonomous exoskeleton reduces metabolic cost of human walking during load carriage. *Journal of NeuroEngineering and Rehabilitation*, 11(1):1–11, 2014.

[32] A. Nasr, S. Bell, J. He, R.L. Whittaker, C.R. Dickerson N. Jiang, and J. McPhee. Musclenet: mapping electromyography to kinematic and dynamic biomechanical variables by machine learning. 2021.

[33] A. Nasr, B. Laschowski, and J. McPhee. Myoelectric control of robotic leg prostheses and exoskeletons: A review. *ASME 2021 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2021.

[34] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation, 2016.

[35] E. Odemakinde. Human pose estimation with deep learning – ultimate overview in 2023, 2023.

[36] J.L. Patton. Robot-assisted adaptive training: Custom force fields for teaching movement patterns. *IEEE Transactions on Biomedical Engineering*, 51(4):636–646, 2004.

[37] J.L. Patton, M.E. Stoykov, M. Kovic, and F.A. Mussa-Ivaldi. Evaluation of robotic training forces that either enhance or reduce error in chronic hemiparetic stroke survivors. *Experimental Brain Research*, pages 368–383, 2005.

[38] D. Pinto-Fernandez, D. Torricelli, M. del Carmen Sanchez-Villamanan, F. Aller, K. Mombaur, R. Conti, N. Vitiello, J.C. Moreno, and J.L. Pons. Performance evaluation of lower limb exoskeletons: a systematic review. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(7):1573–1583, 2020.

[39] F. Pistoia, S. Sacco, C. Tiseo, D. Degan, R. Ornello, and A. Carolei. The epidemiology of atrial fibrillation and stroke. *Cardiology clinics*, 34(2):255–268, 2016.

[40] D. Ramanan. Learning to parse images of articulated bodies. *Proceedings of the 19th International Conference on Neural Information Processing Systems, NIPS'06*, pages 1129–1136, 2006.

[41] F. Rezazadegan, J. Gengb, M. Ghirardi, G. Menga, S. Murèb, G. Camuncolib, and M. Demichela. Risked-based design for the physical human-robot interaction (pHRI): an overview. *Chemical Engineering Transactions*, 43(1):1–6, 2015.

[42] Z. Song, J. Yi, D. Zhao, and X. Li. Multilayer feedforward networks are universal approximators. *Pergamon Press*, 2:359–366, 1989.

[43] Z. Song, J. Yi, D. Zhao, and X. Li. A computed torque controller for uncertain robotic manipulator systems: Fuzzy approach. *Fuzzy Sets Syst.*, 154:208–226, 2005.

[44] J. Tompson, A. Jain, Y. LeCun, and C. Bregle. Joint training of a convolutional network and a graphical model for human pose estimation. *2014 IEEE Conference in Computer Vision and Pattern Recognition (CVPR)*, 2014.

[45] Engineering ToolBox. Friction and friction coefficients, 2004.

[46] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, page 1653–1660, 2014.

[47] E. Washabaugh, J. Guo, C. Chang, D. Remy, and C. Krishnan. A portable passive rehabilitation robot for upper-extremity functional resistance training. *IEEE Transactions on Biomedical Engineering*, 66(2):496–508, 2019.

[48] L.M. Weber and J. Stein. The use of robots in stroke rehabilitation: A narrative review. *NeuroRehabilitation*, 43(1):99–110, 2018.

[49] B. Williams, M. Toussaint, and A.J. Storkey. Modelling motion primitives and their timing in biologically executed movements. *Advances in Neural Information Processing Systems*, 20:1609–1616, 2008.

[50] D.A. Winter. *Biomechanics of Motor Control of Human Movement*. John Wiley and Sons, Inc., 2009.

[51] J.M. Winters. Hill-based muscle models: A systems engineering perspective. *Multiple Muscle Systems*, pages 69–93, 1990.

[52] B. Xiao, H. Wu, and Y. Wei. Simple baselines for human pose estimation and tracking. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[53] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures of parts. *2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1385–1392, 2011.

[54] A. Zell. Simulation of neural networks. *Addison-Wesley*, (1):73, 1994.

APPENDICES

Appendix A

Graphical User Interface

The purpose of the graphical user interface (GUI) is to provide the therapist, patient, and researcher, with a simple interactive tool to control the robot as they desire. The layout of the GUI is described in the following sections.

A.1 Instructions Tab

Below in figure A.1 is the first tab of the GUI. These steps outlined are aimed to provide the therapist and patient with enough information to undergo a patient trial.

A.2 Registration Tab

Below in figure A.2 is the second tab of the GUI. This is where the patient's first and last names are recorded, as well as the session and trial number. Note that a trial was defined as the time between the pushing of the start button and the pushing of the stop button. A session was defined to be a distinct sitting that could consist of multiple trials.

A.3 Robot Functionality Buttons

Below in figure A.3 is the group of push buttons that control the functionality of the robot being driven by a Simulink control model. This button group is shown in every single tab

of the GUI, including the registration tab (figure A.2), such that if the patient or therapist wants to stop the trial at any time, the delay in carrying out this desire is not due to a tab that is currently being displayed not having this functionality.

The build model button generates the real-time C/C++ code for the Simulink model to be automatically downloaded by the robot hardware. The connect button connects the Simulink model to this real-time code on the robot via external mode. The disconnect button disconnects the Simulink model from the real-time code on the robot. The start button (in green) starts the Simulink in external mode. The stop button (in red) stops the Simulink in external mode. And the save data button simply saves the real-time recorded MATLAB data under the file name identified by the information provided in the registration tab.

A.4 Human Joint Angles Tab

Below in figure A.4 is the sixth tab of the GUI. This tab depicts the range of motion of the patient during their trial, as part of a post-trial report that can be seen by the therapist and patient.

A.5 Parameters Tab

A.5.1 Pose Estimation Using Camera

In the scenario when a camera is selected to be used for pose estimation, the idea is that any pre-trained computer vision pose estimator can be used on a video; once a video of a patient trial is finished recording, the joint angle estimates can then be obtained by applying a pre-trained pose estimator on that recorded video. The reason that pose estimation can only be performed after the entire video is recorded is that Simulink is unable to generate stand-alone C/C++ code for the pose estimator models (both MATLAB and Python) and all of the MATLAB commands that would record a video in real-time with the robot. Hence, another GUI is essentially used in parallel on a separate MATLAB session, in order to start and stop the recording of a video. Below in figure A.5 is the video tab of this GUI, which essentially shows the real-time frame of the camera video. In an effort to synchronize the time instances of the robot data and video, the start and end time stamps are recorded for both the robot data and video.

It is also important to note that the frame rate of the video is 30 frames per second, and the sample rate of the data acquisition in Simulink is 0.002 seconds. Due to these differences in sample rates between the robot and the video capture, 1D interpolation has to be performed on the pose estimation estimates of the human joint angles, such that the sample rate matches the rate of the robot. Other metrics such as the patient's exerted joint torques can also then be analytically calculated for each 0.002-second time step. In the following subsection is the corresponding parameters tab.

Below in figure A.6 is the parameters tab, which is the third tab of the GUI. In this tab, the user inputs the radius of the half-circle trajectory (or D trajectory), the time to complete one half-circle, and the centre of the straight line of the half-circle. The human parameter inputs are necessary inputs for the human joint velocity (equation 3.31), joint acceleration (equation 3.32), and joint torque calculations (equation 3.20). These include the mass of the patient, the length of their forearm, and the length of their upper arm.

A.5.2 Pose Estimation Without Camera

Sometimes the patient or therapist may be uncomfortable having a camera phone over the patient's head. Therefore, the option to have pose estimation be performed on a patient without the use of a camera is presented. This requires the use of the regression technique. The regression technique relies on the assumption that the patient's shoulder joint position is known and does not move throughout the duration of the trial. Specifically, the regression technique involves solving equation 3.28 for θ_{R1} and θ_{R2}, while x_s and y_s are known. In the following subsection is the corresponding parameters tab.

Below in figure A.7 is the parameters tab, which is the third tab of the GUI. In this tab, the user inputs the radius of the half-circle trajectory (or D trajectory), the time to complete one half-circle, and the centre of the straight line of the half-circle. The human parameter inputs are necessary inputs for the human joint velocity (equation 3.31), joint acceleration (equation 3.32), and joint torque calculations (equation 3.20). These include the mass of the patient, the length of their forearm, and the length of their upper arm. The shoulder joint location inputs are required for the reasons outlined before.

A.6 Human Joint Torques Tab

Below in figure A.8 is the seventh tab of the GUI. This tab depicts the range of torques exerted by the patient during their trial, as part of a post-trial report that can be seen by

the therapist and patient.

A.7 Trajectory Performance Tab

The performance of the patient is split into two categories: positional performance and the subject's directional performance, which is a function of their force exertion on the robot's end-effector where the force sensor is located. The reason to split a patient's performance into these categories is that there exists a scenario when the patient's positional performance is very good, but the magnitude level of force they exert on the end-effector is very low; the patient is not contributing any effort, but rather essentially "just going for the ride". The reverse may also happen in which the patient's positional performance is very poor, but the magnitude level of force they exert on the end-effector can at times be very high. This scenario may indicate that the patient is exhibiting more spasmodic behaviour, given that the patient is attempting to follow the desired trajectory. Therefore, splitting the performance report into these two complimentary categories should provide a more comprehensive evaluation of the motor recovery of the patient.

Below in figure A.9 is the fourth tab of the GUI. This tab depicts the positional performance of the patient's hand in both the x and y directions of the planar surface, as well as the patient's directional performance, which is a function of their force exertion on the robot's end-effector where the force sensor is located, with respect to the desired trajectory direction for each time instance. Also included in this tab is a graph that, at the end of the trial, will show the trajectory performance of the patient with respect to the desired trajectory (D shape/half circle).

A.8 End-Effector Force Tab

Below in figure A.10 is the fifth tab of the GUI. This tab depicts the range of forces exerted by the patient on the end-effector in the x and y directions during their trial, as part of a post-trial report that can be seen by both the patient and their therapist.

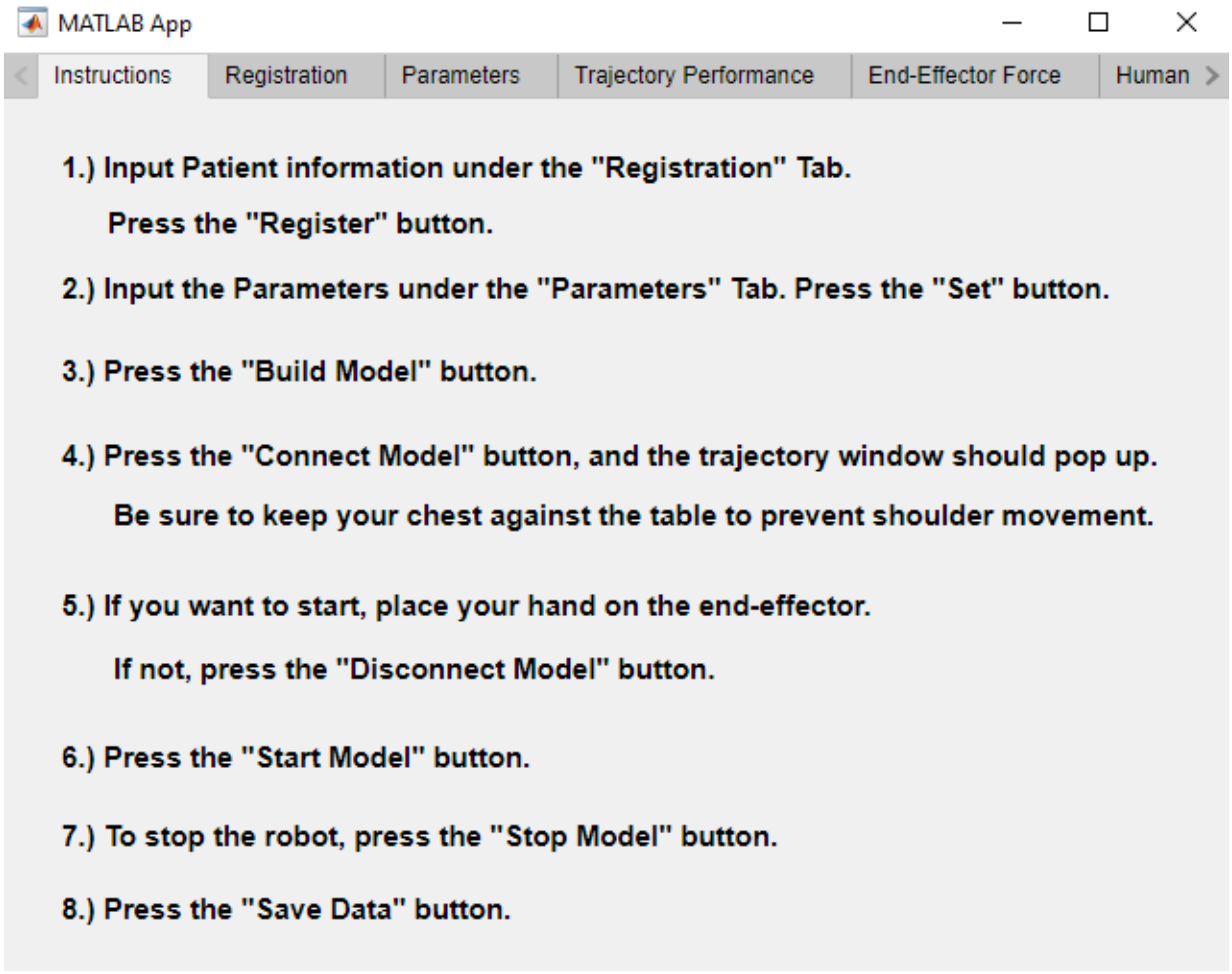

Figure A.1: GUI tab showing the instructions for the patient and therapist to follow.

Figure A.2: GUI tab showing the registration fields to be filled out by the patient.

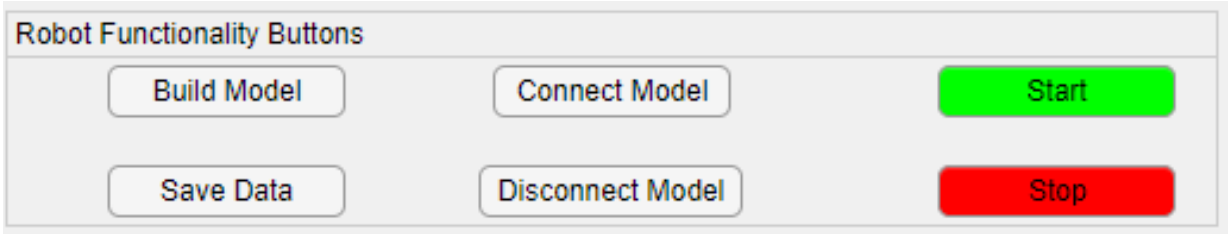

Figure A.3: Push buttons that control the functionality of the robot being driven by a Simulink control model.

Figure A.4: Human joint angles tab, which shows the time series of the joint angles and the corresponding range of motion.

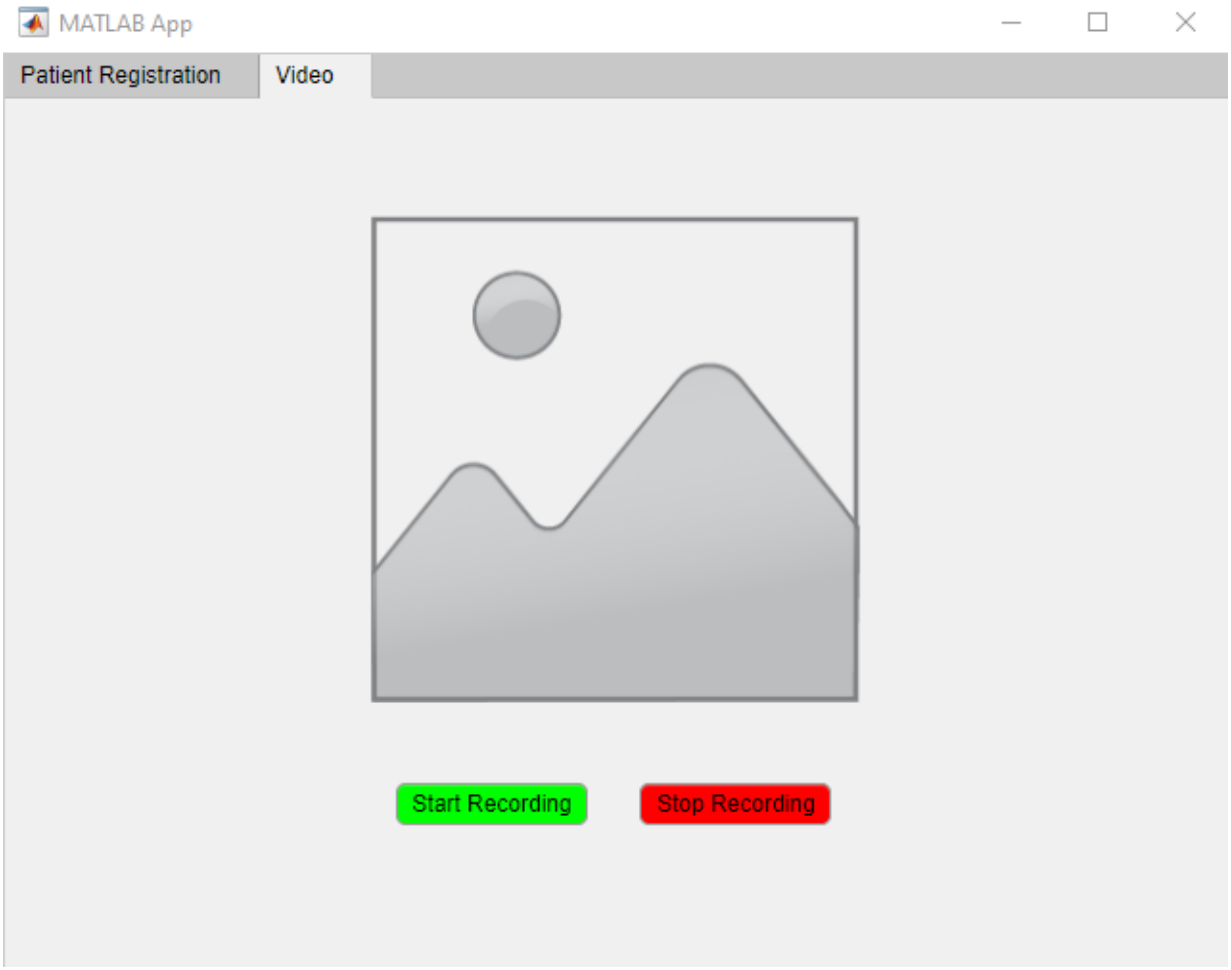

Figure A.5: Video Tab of the Recording GUI.

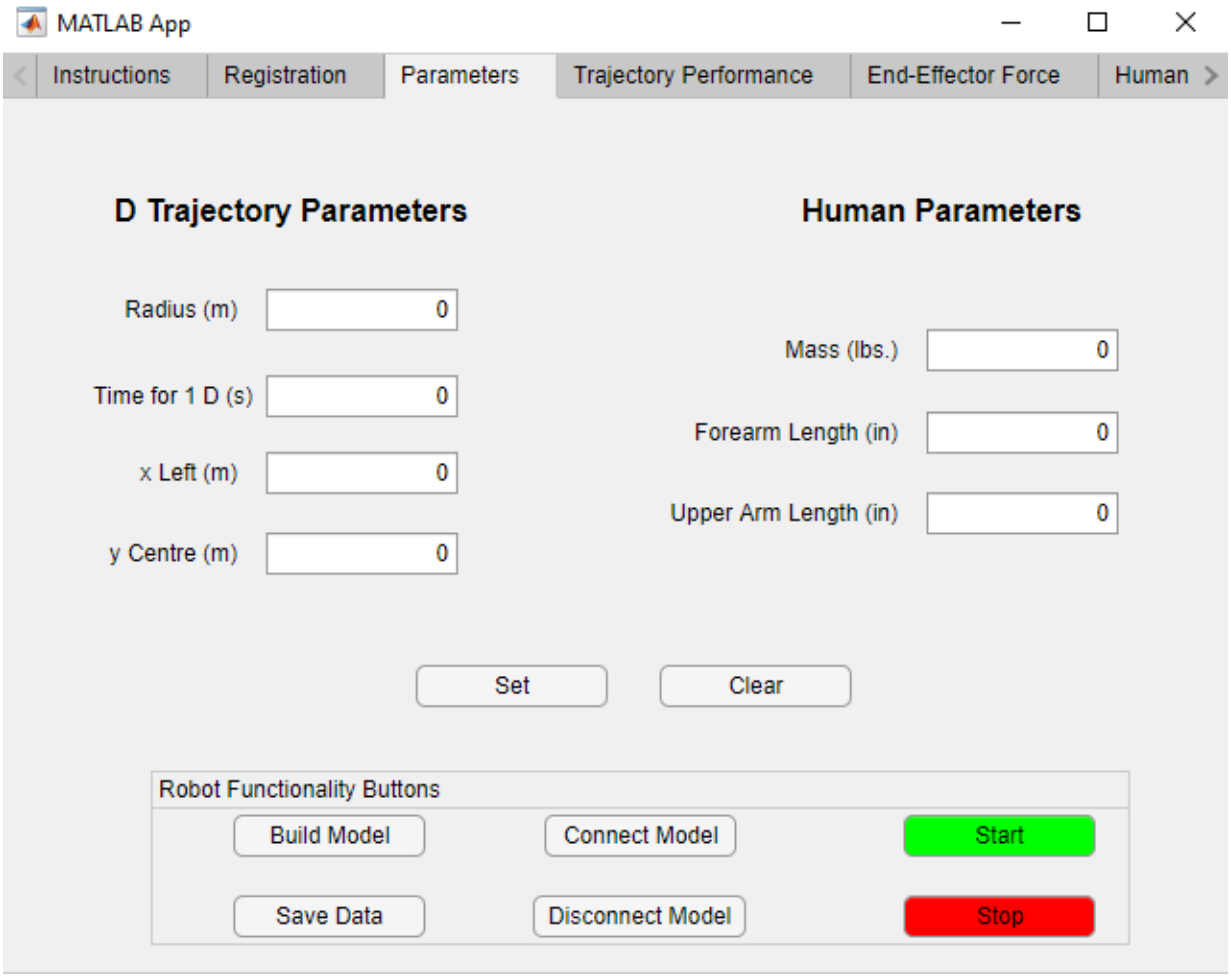

Figure A.6: Parameters tab using a camera.

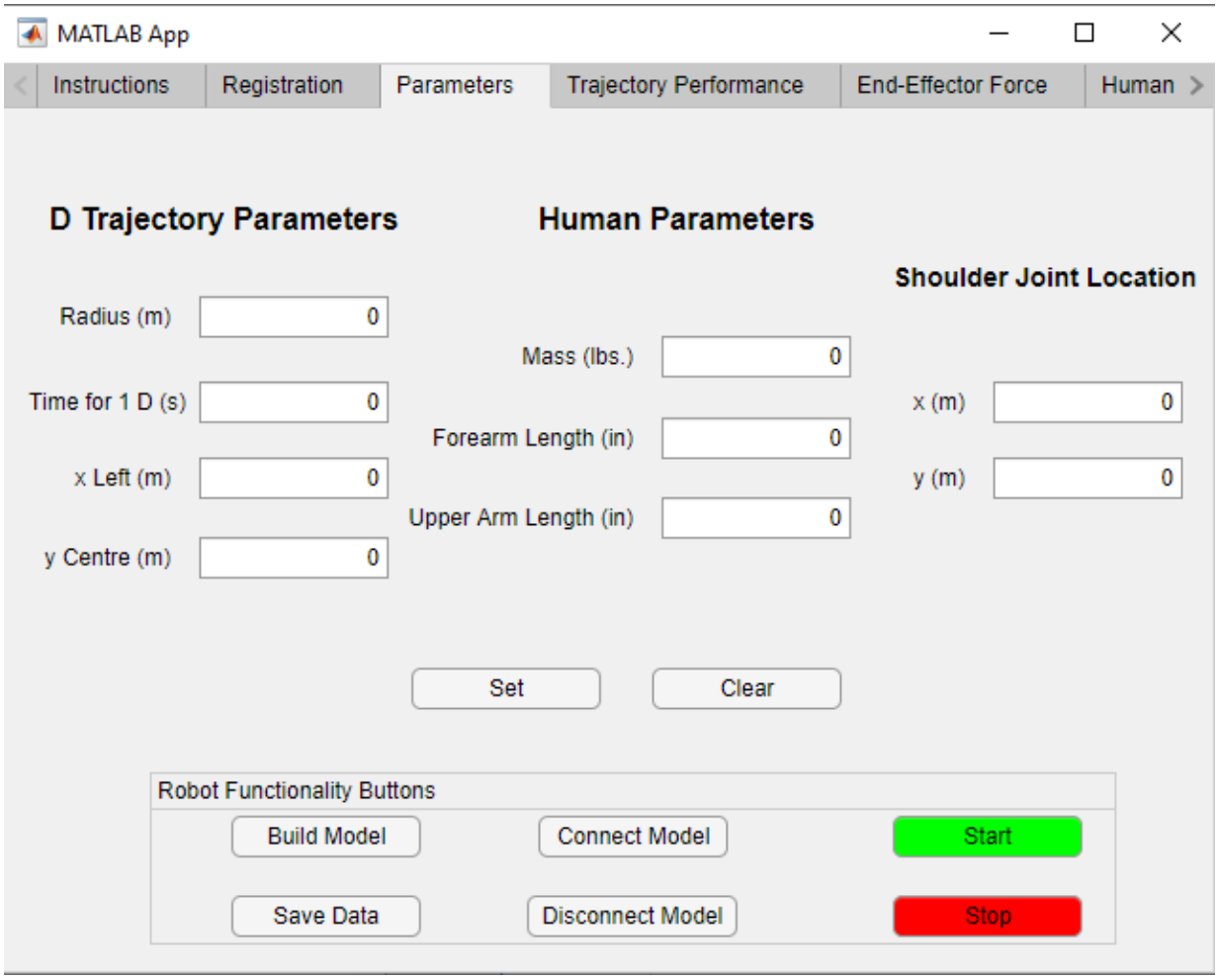

Figure A.7: Parameters tab using no camera.

Figure A.8: Human joint torques tab, which shows the time series of the joint torques and the corresponding range of torques exerted by the patient.

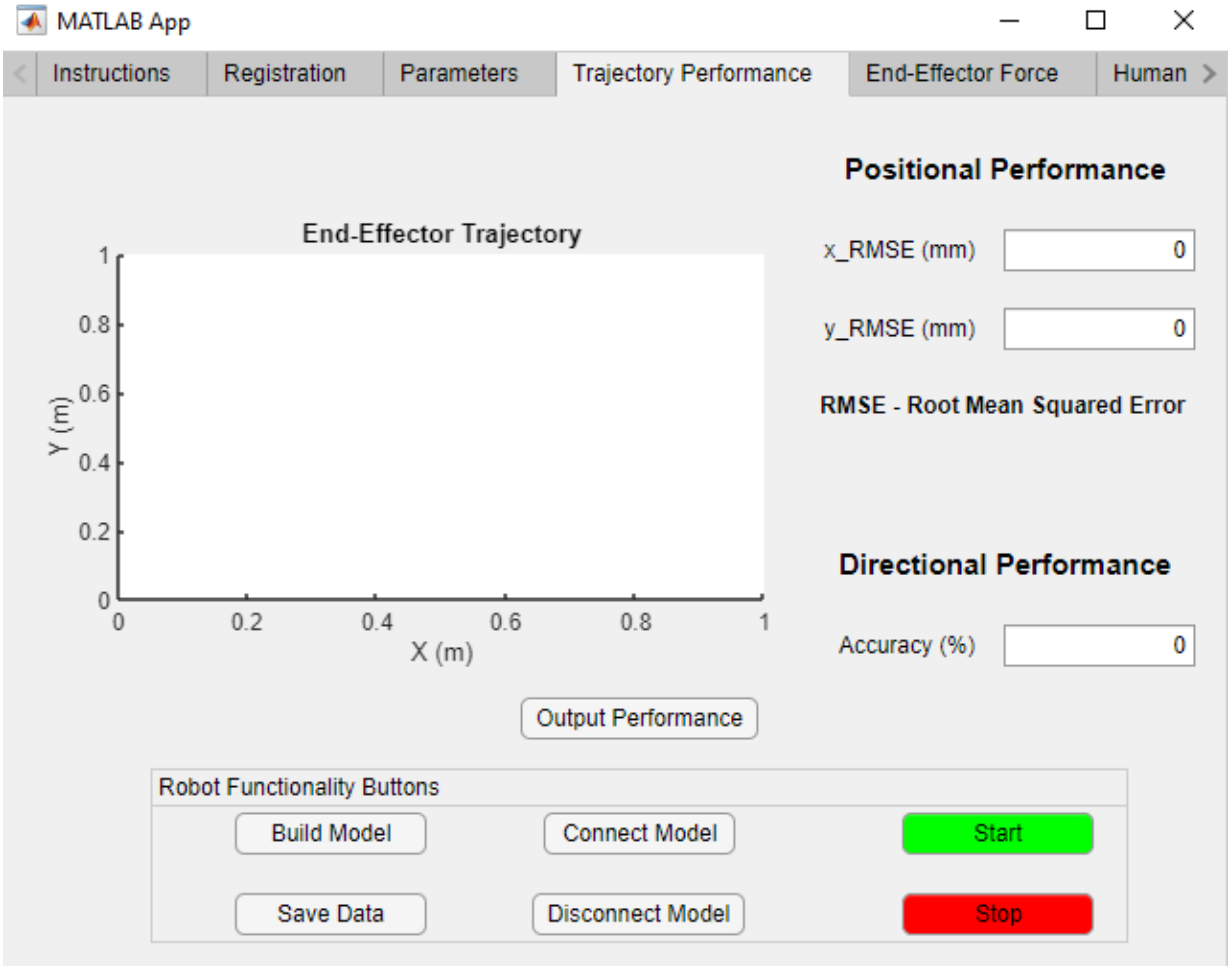

Figure A.9: Trajectory performance tab, which shows both the positional and directional performance of the user during their trial.

Figure A.10: Force performance tab, which shows the time series of the 2D force exerted by the patient on the end-effector and the corresponding range of forces.