Adaptive Optimal Control

in Physical Human-Robot Interaction

by

Rebecca C. Bell

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2019 by the
Graduate Supervisory Committee:

Wenlong Zhang, Chair
Erin Chiou
Daniel Aukes

ARIZONA STATE UNIVERSITY

May 2019

ABSTRACT

What if there is a way to integrate prosthetics seamlessly with the human body and robots could help improve the lives of children with disabilities? With physical human-robot interaction being seen in multiple aspects of life, including industry, medical, and social, how these robots are interacting with human becomes even more important. Therefore, how smoothly the robot can interact with a person will determine how safe and efficient this relationship will be. This thesis investigates adaptive control method that allows a robot to adapt to the human's actions based on the interaction force. Allowing the relationship to become more effortless and less strained when the robot has a different goal than the human, as seen in Game Theory, using multiple techniques that adapts the system. Few applications this could be used for include robots in physical therapy, manufacturing robots that can adapt to a changing environment, and robots teaching people something new like dancing or learning how to walk after surgery.

The experience gained is the understanding of how a cost function of a system works, including the tracking error, speed of the system, the robot's effort, and the human's effort. Also, this two-agent system, results into a two-agent adaptive impedance model with an input for each agent of the system. This leads to a nontraditional linear quadratic regulator (LQR), that must be separated and then added together. Thus, creating a traditional LQR. This new experience can be used in the future to help build better safety protocols on manufacturing robots. In the future the knowledge learned from this research could be used to develop technologies for a robot to allow to adapt to help counteract human error.

DEDICATION

I would like to dedicate this to my parents for always pushing me further.

ACKNOWLEDGMENTS

I would like to express my thanks of gratitude to my professor, Dr. Zhang, and Ph.D. student, Yiwei Wang, who gave me the golden opportunity to do this wonderful project on the topic of Adaptive Control. This permitted me to do an immense amount of research and I came to learn about a great deal of new things. I would also like to give a special thanks to Dr. Oakes and Dr. Isom for helping me in understanding the complex mathematical side of this project.

Finally, I would like to thank my parents and friends who assisted in finalizing this project within the limited time frame.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

Figure Page

PREFACE

Each day there are military troops and individuals suffering from accidents around the world, leaving people with injuries that will affect them for the rest of their lives. When my god-father was deployed in Iraq, he was medically discharged after sustaining a broken pelvis. He underwent a traditional hip replacement and received a femoral implant, which left him with a limp and some physical limitations. In the future, there may be a possibility for a new type of implant or robotic joint that could improve the results of joint replacement such as this. More research regarding physical human-robot interaction must be conducted for this to become a reality.

Robotics engineering covers a multitude of topics including industry, prosthetics, rehabilitation machines, and even education. Considering the types of engineering and the technology currently available, robots could help humans improve both their efficiency in the workplace and their quality of life.

Robots can be utilized in industrial settings to reduce takt time and physical strain on workers. Currently, prosthetics and rehabilitation machines are being engineered with motors and actuators. One day they may also include artificial intelligence in order to minimize the patient's struggles by assisting with the patient's body and muscles. In schools, children with autism interact with robots to learn acceptable social behaviors and understand social cues provided by the robot [1]. This type of interaction can be seen in Figure 1. Before any of these robots can be implemented into society, there must be a fundamental understanding of the interactions between humans and robots, which may be the future of the robotic engineering field. This thesis is based on research regarding the optimization of the physical relationship between a robot and a human. Simulations have

established the idea that interaction forces are created when humans and robots work together. This in turn can cause the robot to recognize alterations to the plan of action and adapt quickly with minimal strain on the human.

Figure 1 - Children with autism working with robotic doll, learning social skills [1]

CHAPTER 1

INTRODUCTION

1.1 Literature Review and Background

When a person goes through physical therapy, they are put through different movements with resistance bands. When my grandma shattered her shoulder, she had a metal plate put in. After she was healed enough to be able to move her arm again, she was given stretches to strengthen her shoulder. One of these exercises was to grab both ends of a rope that is hanging from a pully and use the good arm to pull the other arm up so that it could stretch her shoulder. Due to the type of stretches, she had to do them on her own. Yet, because she did not want to push herself too far, she never was able to regain full motion of her arm.

What if there was a robot that could help her stretch, by helping her raise her arm and push her to reach a little higher each time? Then maybe she could regain the whole range of motion to that shoulder thanks to the help of that robot. To do this, the robot would have to have a slightly different desired goal and path than the human. For example, the human wants to be able to reach the whole area of the table but can only reach a small circle within the table due to an injury. With the robot's help and desire to reach the whole area of the table, it gives some pull on the human that helps the human reach further than it could before, but also be adaptive so that it would not go beyond the human's physical limits. This could be done with a robot that has adaptive control within its programming.

Role adaptation control can be used in physical human-robot interaction (HRI) for rehabilitation (as seen in [2]) and social development, like working with children with

autism (as seen in [3]). Another way robots have been implemented is in home settings, using a service robot [4] or an industrial setting, using a robot designed to carry heavy objects [5]. As technology advances, interactions between humans and robots will become more frequent, presenting a need to understand the coordination and safety aspects of their relationships. One way that was looked at to understand the coordination aspect of HRI is in article [6], using an adaptive optimal control in a nonlinear system like the robots used in manufacturing. Then we must understand the importance of the safety aspect. For instance, in article [7] a safety protocol is used to access the surrounding area of the robot to ensure there is not any danger within the area. These aspects will be necessary to educate robot users about safety protocols concerning the utilization of robots in many industries [8]. Research in the field of physical HRI specifically looks at continuous time role adaptation and the dynamics of shared control [9]. When performing a single task, research has shown that humans and robots have different goals, which allows researchers to understand the adaptive shared control between the human and robot [10].

The importance of role adaptation is to understand the models of the human-robot relationship so that it can be adapted into a model that is able to assist people. Current techniques of robot utilization include social learning for children diagnosed with autism [11] and physical therapy [12]. One way of understanding the role of adaptation is using motion capture cameras to examine human movements [13] and behaviors when interacting with robots [14] as well as other humans [15]. Prior to looking at a system using optimal control [7], the robot must be modeled to be able to analyze how it interacts with its environment. At this point, the model can then be altered to fit any need,

including but not limited to developments in safety [16], rehabilitation and therapy [17], and even dance instruction [18].

In industry there are several different reasons to incorporate a robotic system into the work place. For example, the Kuka robot seen in Figure 2, made for the assembly line to lower takt time and increase the accuracy of the machined parts. These robots can be used to manufacture parts more precisely [19] or help workers handle tools [20]. Role assignment policies have become an important research topic when it comes to robots in industry [21]. In article [21], it shows that there are several issues that need to be resolved before humans and robots can work together seamlessly. These issues are cues that a robot can understand: social factors, competition between dyads, mechanical impedance, asymmetric relations, and psychological aspects [21]. Therefore, understanding how a human and a robot interact and how to better this interaction is very important in the industry. If people and robots can interact seamlessly, then manufacturing plants become a more productive and promote a higher output of product.

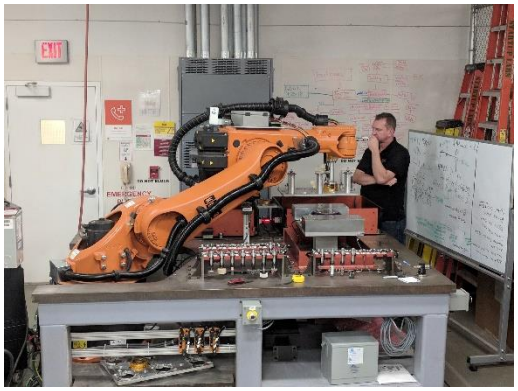

Figure 2 - Kuka Robot being setup to run on the assembly line - ASU 2018 Capstone with Raytheon

Robots are being used in rehabilitation using a nonlinear adaptive impedance model with haptic feedback [22]. The research provides a stable impedance model that

can be used in the Cartesian space [22]. Therefore, this research can provide a basis to build upon for recreating a stable system that will allow robots to help in rehabilitation and physical therapy. There is also research that has investigated exoskeletons to help strength joints, like an exoskeleton upper arm to help strengthen and move the upper arm. [23]. Also, built upon the idea of exoskeletons, there is a new type of exoskeleton known as soft robotics. It can be used as any other exoskeleton, but instead of actuators and motors it has inflatable bags like seen in article [24] when used on the elbow to help movement. Then soft robotics is also being used in HRI as seen in [25] to allow for easier collaboration between the robot and human when the exoskeleton is on the human.

People interact every day, but they never analyze how to interact with someone else. When a robot interacts with a human, there is a great amount of thought that goes into how the robot should interact with humans, specifically socially [26]. There has been research done that investigates how to look at the conceptual model. For example, how to articulate, intentionality, interpret, and evaluate the problem at hand [27]. This investigation into how to model HRI includes looking at the problem from the view of the robot acting as a human, just like an actor becomes a character when acting in a movie [27]. With this way of thinking, it could lead to a new way in how robots are programmed to interact like a human, like how robots are being programmed to work with humans in a more human-like way [28]. Just like how article [29] investigates how to calculate the optimal grasping and manipulation forces needed to control and use the robotic hand properly.

Another type of research going on in HRI is having robots socially interact with children to help build social bonds [30]. This is done by having the children work with

their robot companion, virtual and physical. The result is that the children respond best with robots that adapt to their behavior, especially when there is physical interaction from the robot instead of virtual [30]. This is important because if physical robots can be used to help children learn best and can be used for helping children and adults with autism understand social situations [31]. Yet, these robots do not have to be fancy, there is some research that is looking into the benefits of Lego robots when used to help children with autism and their social behavioral growth [32]. For this physical interaction can happen, there must be a foundation of how robots and humans interact.

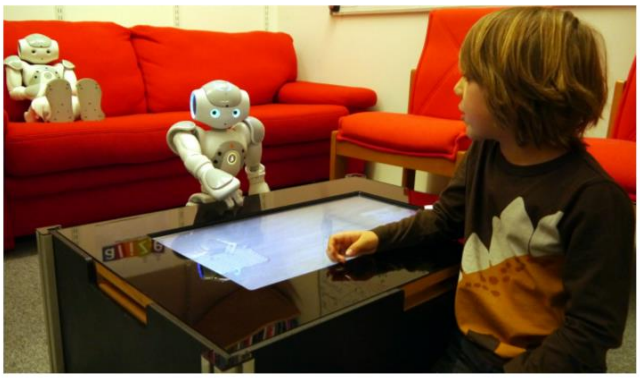

Figure 3 - Social interaction between a child and robot, using a touchscreen to interact [30]

The common link between the different types of HRI is the physical aspect. These interactions are mostly based on force and how the robot responds to the human's input. Thus, if the interaction between the human and robot is going to be smooth, then the robot needs to be able to respond to the human's interaction force in a certain way. This thesis looks at an algorithm that adapts to the interaction force being inputted into the system as well as how this system can be adapted to minimalize the amount of force is needed to change the direction of the robot's path. The relationship between the human

and robot can be improved in the future to allow for the use in the medical, industrial, and social fields. The basis of the derived equations and cost functions will come from [9], [6], [33], and [34] to build the basic model.

1.2 Motivation

The motivation to do this project is to examine and develop an adapt robotic control method to improve physical HRI, such as prosthetics to help disabled veterans and to grow the adaptiveness of robots that could work with children with disabilities to improve their life experience.

1.3 Organization of Paper

This thesis is organized using six different chapters. Chapter 1 provides and literature review and background of the research that has been done in the pass and motivation for this thesis. While Chapter 2 is setting up the system that is being analyzed. Next, Chapter 3 establishes all the critical equations needed for running the algorithm as well as the assumptions made. Once the equations have been established the algorithm is constructed in Chapter 4 before going onto the tests seen in following chapter. In Chapter 5, the constants of the system are changed to check the system is working properly before testing the adapting part of the algorithm. Thus, three techniques are conducted to update the weight within the system that makes it adaptive. Then, Chapter 6 is about the limitations and challenges of the system, the applications of this algorithm, and the future work that is required to be able to use this algorithm. Lastly, at the end of this thesis there are five appendices with the different codes that are used in this thesis.

CHAPTER 2

THE SYSTEM: TWO-AGENT SHARED CONTROL

Research in physical human-robot interaction has mostly been on the leader-follower role adaptions [35]. This thesis will look at how a continuous role adaption algorithm could be adopted into a robot control that allows the robot to be able to teach their human partner to do something new. If a robot can adapt to a human's action based on force, then a robot could teach a person how to dance [18] or teach a person how to walk again after surgery. If an algorithm could be configured to control a robot to be able to adapt based on the human's actions and allow for the robot to be able to teach the human without pushing the human passed his or her limits [9].

The human-robot system is a two-agent take on game theory, with the cost function of each agent with the system set-up almost the same as in [9]. The system includes the human and the robot as their own entities, working together with the same goal in mind, but different ways of achieving this goal. The way this is done is the human and robot work together to draw a path, but the human has a different desired path than the robot. To solve this problem, game theory needs to be introduced. Game theory is looking at how a human and robot can best work together if their cost functions or goals, are different [36]. Not only can game theory be used to help understand this relationship, but it can also help analyze how this relationship will affect each agent [37] and understand the social aspect of the system [38].

This thesis is looking at reconstruction of the system seen in article [9] and reproduce the results, however due to unseen circumstances, this thesis looks at the reconstruction of the system and different techniques it could use to adapt. In this system,

the human will be working with the robot to draw a desired path. The overall goal is to work together to move the object with the robot's goal to move the block along the red path and the human's goal is to follow the blue path, as seen in Figure 4. Yet neither knows each other's desired path goal. The robot's adaption is controlled by the interaction force between the human and the robot, with the desire of the optimal control producing the minimal interaction force while keeping the desired path of the human. This is done by a computer simulation of the robot and human interaction in order to understand how the robot can adapt to the human's actions.

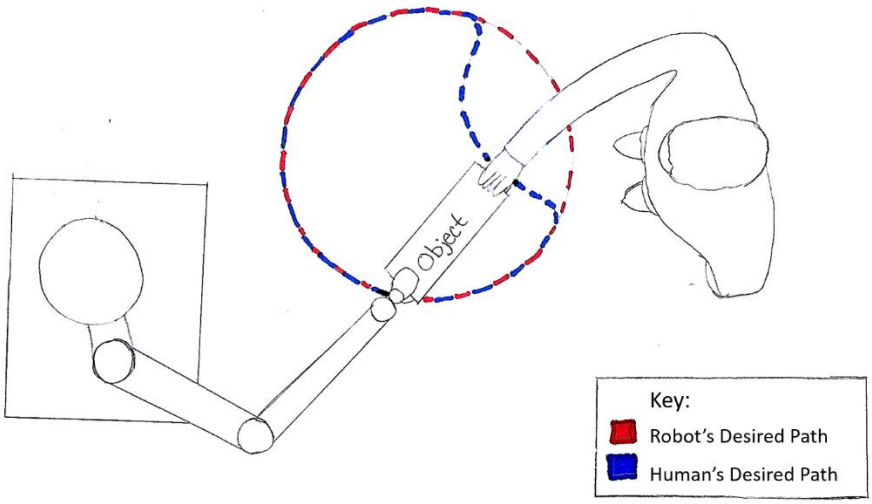

Figure 4 – Illustrated scenario of the system

CHAPTER 3

METHOD

3.1 System Variables and Initial Values

The system being tested is a two-input system: one input from the robot, and the other from the human. In the simulation, the input from the human is fixed while the input from the robot changes based on how the human acts while working with the robot. These equations are based on the equations seen in article [9].

Equation (1) looks similar to a traditional steady-state system equation with one exception: there are two input matrices and two input vectors. This system has the human's and the robot's reaction effecting the system. Similarly, in this system the B_1 matrix is the same as B_2 matrix in this model. This allows the system updates and the robot reacts, the adjustments the robot makes in response to the human is directly proportional to the action of the human.

$$\dot{\bar{z}}(t) = \bar{A}\bar{z}(t) + \overline{B_1}u(t) + \overline{B_2}f(t) \quad (1)$$

With an input from the human (*f(t)*) and an input from the robot (*u(t)*) can have different input matrices depending on what is desired from the system. The values for these matrices seen in (1) and the rest of the equations can be found in Appendix A. Equation (1) does not give the desired path results as seen in [9], instead it must be changed to (1a) before the outcome of the graphs will look like the results on page 677 of [9]. This is because the force should be affecting the system in a negative way in the position plot. However, this does not happen. If the human's input of the system is changed to a negative, then the system reacts as desired, specifically in the position plots of the system reactions. These differences can be seen later in Chapter 4.

$$\dot{\bar{z}}(t) = \bar{A}\bar{z}(t) + \overline{B_1}u(t) - \overline{B_2}f(t) \tag{1a}$$

$$\bar{z} = \begin{bmatrix} x \\ \dot{x} \\ \omega \end{bmatrix} \tag{2}$$

Now $\bar{z}$ is formatted like (2) because the first state is the position vector of the system in the x and y direction. Then the next state is the velocity vector of the system in the x and y directions. Finally, the third state is the auxiliary state of the system in the x and y directions. All these states are only in the x and y directions since this system is only working in a 2D plane. Thus, this can easily be changed into a 3D system by changing the state's dimensions. This formatting of the state matrix is also seen on page, 673 in article [9].

$$\dot{\omega} = U\omega$$

$$x_d = V\omega \tag{3}$$

To acquire the initial auxiliary state equation (3) can be used. This auxiliary state is based on the desired position state, x_d, and a constant matrix, *V*. This desired path is the path the robot wants to take, a circle about the center of (0,0) with a radius of 0.2, as seen in (4). This path was chosen because a circle is easiest to simulate over time. This trajectory is then used in this reconstruction of article [9] to find the auxiliary state ω, $x(0)$, $\dot{x}(0)$, and $\bar{z}(0)$.

$$x_d = \begin{bmatrix} -0.2\cos(\frac{\pi}{5}t) \\ 0.2\sin(\frac{\pi}{5}t) \end{bmatrix} \tag{4}$$

The first assumption when solving for $\dot{x}(0)$ is that it can be differentiated from x_d as seen in (5). This assumption is based on the initial value of x_d used for the initial value of *x*, so if x_d is differentiated to change the position to the velocity, then x_d can also be

used to find the initial velocity. But, from then on, according to Algorithm 1 on page 675 in [9], $\dot{x}$ can be solved from equation (1).

$$\dot{x_d} = \begin{bmatrix} 0.2\sin(\frac{\pi t/_5}{25}) \\ -0.2\cos(\frac{\pi t/_5}{25}) \end{bmatrix} \tag{5}$$

One of the ways to understand how the system is reacting is to look at the gain of the system. Meaning that K_1 and K_2 need to be calculated and looked at as the system runs to see how the robot and human parts of the system are reacting. If the system is reacting properly, K_1, the robot side, should be changing proportional to K_2, the human side, but in the opposite direction. The way these values are calculated is found in article [9] and are shown below in (6) and (7). Equations (6) and (7) are used based on separating the system into 2 parts, the robot system part (6) and the human system part (7). When the norm of these two equations are graphed over time, the graphs show how the system is behaving. This is important to note because the systems should look alike if the R_2 is updating correctly. The gain plots will be looked over in more depth in Chapter 4 and 5.

$$K_1 = -\frac{1}{2}{R_1}^{-1}\overline{B_1}^T P\bar{z} \tag{6}$$

$$K_2 = -\frac{1}{2}{R_2}^{-1}\overline{B_2}^T P\bar{z} \tag{7}$$

On the other hand, to be able to calculate the gains of the system, P must first be calculated. To accomplish this, a Riccati equation can be used. This equation is not a simple Riccati equation because this is a two-agent system. The structure of this complex Riccati equation can be found in article [9], equation (8). This is a LQR of a 2-input system which makes a normal LQR solver almost impossible to use unless the equation is

manipulated some. In order to be able to solve for *P* in the LQR of (8), it first needs to be separated into 2 separate Riccati equations. From here they can be added together and then divided by 2 so that the resulting LQR looks like (9). Another way to solve for *P* in the LQR, (8) is referencing the book, [33], that sets up the problem to solve a two-agent LQR as a traditional LQR on pages 280-284. Resulting in equation (9). The only difference between what is in [33] and how (9) is formatted is the *B* part of (9) needs to be divided by 2. This is due to if (8) is separated into two separate equations, then added together, every part is multiplied by two except the *B* part of the equation. However, when simplified so that the whole equation is divided by two, the *B* part gets divided by two, resulting in equation (9). Due to this technique, the **first assumption is that (8) = (9)**.

$$\bar{A}^T P + P\bar{A} + Q - P\overline{B_1}R_1^{-1}\overline{B_1}^T P - P\overline{B_2}R_2^{-1}\overline{B_2}^T P = 0_{mxm} \quad (8)$$

$$\bar{A}^T P + P\bar{A} + Q - P\frac{(\overline{B_1}R_1^{-1}\overline{B_1}^T + R_2^{-1}\overline{B_2}^T)}{2}P = 0_{mxm} \quad (9)$$

Leading to where Q, $\bar{A}$, $\overline{B_1}$, R_1, $\overline{B_2}$, and R_2 came from. Each one has their own set of values. Some of the variables are constants in the system while other variables change over time as the system adapts. Now, to get $\bar{A}$, $\overline{B_1}$, and $\overline{B_2}$ values for *A*, *B₁*, and *B₂* need to be understood. These values can be seen in (10). The *A*'s and *B*'s matrices are formatted like a traditional *A* and *B* matrices of a steady state system, based on the mass and damping constants of the system. Since this system does not have a spring constant, this constant is neglected from the *A*'s and *B*'s matrices of this system. These values can be seen in (10) with $\bar{A}$, $\overline{B_1}$, and $\overline{B_2}$ in (12).

$$A = \begin{bmatrix} 0_{m\times m} & I_{m\times m} \\ 0_{m\times m} & -M_d^{-1}C_d \end{bmatrix}$$

$$B = B_1 = B_2 = \begin{bmatrix} o_{m\times m} \\ -M_d^{-1} \end{bmatrix} \tag{10}$$

These matrices in (10) are used for the state-space form seen in (11) of the dynamic system.

$$\dot{z}(t) = Az(t) + B_1u(t) + B_2f(t) \tag{11}$$

However, this cannot be used for an optimal tracking problem, so the system must be transformed. Equation (1) is the new augmented system. Thus, (10) must also be augmented, which can be seen in (12). These matrices are based on the three states in $\bar{z}$. So, to augment *A* and *B*, the auxiliary state needs to be considered. For this reason, the *A* and *U* matrices are in $\bar{A}$, so that it updates the system the same way as a traditional A matrix. Similarly, $\bar{B}$ is formatted.

$$\bar{A} = \begin{bmatrix} A & 0_{2m\times l} \\ 0_{l\times 2m} & U \end{bmatrix}$$

$$\bar{B} = \overline{B_1} = \overline{B_2} = \begin{bmatrix} B_1 \\ 0_{l\times m} \end{bmatrix} \tag{12}$$

Yet, there is a slight issue with the annotation of the dimensions within (12). The *l*×*2m*, *2m*×*l*, and *l*×*m* are supposed to be zero matrices that are the proper dimensions, but in B_1 is a 4×2 matrices, which means the *l*×*m* should be 2×2 matrix to make it a 6×2 matrix. This assumption is due to $\bar{A}$ being a 6×6 matrix. So, this mean *l*=*m*. Therefore, this will also solve the confusion for $\bar{A}$, since *l*×*2m* and *2m*×*l* are supposed to be 2×4 and 4×2 matrices to make the overall matrix square and match the dimension made by *A* and *U*.

Since *l*=*m* must be true, then this also helps with the dimension issues of *V* and *U*, since on page 673 of article [9] it states $V\epsilon\mathbb{R}^{n\times l}$ and $U\epsilon\mathbb{R}^{l\times l}$, but on page 677 when article [9] explains the values used for the simulation, the results are in (13).

$$V = \frac{1}{\pi}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

$$U = \begin{bmatrix}0 & \frac{\pi}{5}\\ -\frac{\pi}{5} & 0\end{bmatrix} \tag{13}$$

This means that *n* must also equal *l*, leading to **Assumption 2: *m*=*n*=l**. So, if this is the case than when building the matrix *Q*, the dimensions also work out. *Q* can be seen in (15), but to understand the dimensions, Q_1 and Q_2 need to be understood. Q_1 and Q_2 are shown below in (14).

$$Q_1 = 50\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

$$Q_2 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \tag{14}$$

The matrices values in (14) are the values used for the simulation in [9]. Q_1 is the weight of the robot's side while Q_2 is the weight of the human's side. These values must be inputted into *Q*, (15), to be able to use the values in the LQR.

$$Q = \begin{bmatrix}Q_1 & 0_{m\times m} & -Q_1V\\ 0_{m\times m} & Q_2 & 0_{m\times l}\\ -V^TQ_1 & 0_{l\times m} & V^TQ_1V\end{bmatrix} \tag{15}$$

As seen in (15) *m* and *l* must be equal according to Assumption 2, otherwise the 6×6 matrix of *Q* would have holes that are unexplainable. Under Assumption 2, these zero matrices are 2×2 in (15), (12), and (10), then the overall row dimension for the augmented system is 6, meaning *P* will be a 6×6 solved from the LQR and $\bar{z}$ is a 6×1 matrix. *m*, *l*, and *n* must all equal each other for these equations to work. One could make

them not equal each other, but then the matrices will not compute properly since x_d is a 2×1 as given in (4), so $\dot{x}$ must be a 2×1 and ω calculates to a 2×1, making $\dot{\bar{z}}$ a 6×1. Understanding this then leads to understanding u, the control input of the robot.

$$u = -\frac{1}{2}{R_1}^{-1}\overline{B_1}^{T}P\bar{z} \tag{16}$$

$$u^{*} = -\frac{1}{2}{R_1}^{-1}\overline{B_1}^{T}P\bar{z}^{*} \tag{17}$$

Under the assumptions made, u should be a 6×1 matrix, solved using (16). This is considered the actual control input of the robot during the simulation or during a physical interaction between human and robot. While (17) is considered the optimal control input of the robot, based from the optimal state of the optimal system (18). The optimal state of the system is updated per (18). The way (18) is formatted is the same as (1), to ensure the system updated the same, but using the optimal inputs instead of the actual inputs.

$$\dot{\bar{z}}^{*}(t) = \bar{A}\bar{z}^{*}(t) + \overline{B_1}u^{*}(t) + \overline{B_2}f^{*}(t) \tag{18}$$

Once the equations that define the system are in place, the equations that change over time within the system must be implemented. This is seen in the next section.

3.2 Simulation Study

The difference between the actual state and the optimal state for this system is the force matrices. The actual force, (19), is kept the same as what is seen in article [9], to keep a solid reference point when comparing plots of how the system is reacting later. Thus, the forces seen in (19) will not change throughout all the tests done on this system.

The optimal force is updated just like the optimal input of the robot but using the weights of the human instead of the robot's weights. This can be seen in equation (20).

$$f_x = \begin{cases} 0N, & t \leq 4s \\ 0.5N, & 4s < t \leq 5s \\ 0.1N, & 5s < t \leq 7s \\ 0 & t \geq 7s \end{cases} \quad (19)$$

$$f^* = -\frac{1}{2} {R_2}^{-1} \overline{B_2}^T P \bar{z}^* \quad (20)$$

When f^* is solved, f^* becomes a 2×1 matrix while f_x is a scalar in (19). Hence why that force is noted with an x, meaning this force in only affecting the x direction of the system. So the y direction of the force is noted as $f_y = 0N$ for $t \in [0,10]s$, which changes the force to become dependent on an x and y values, making it a 2×1 matrix, $f(t) = \left[f_x(t), f_y(t)\right]^T$. This means the human's input force is only in the x direction, however, as seen in the system's position plots, this force will affect the system in both the x and y directions.

$$R_1 = 0.5 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (10)$$

$$R_2 = 20 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (21)$$

The final piece of the puzzle for the equations are the values for R_1 and R_2. These values are shown above in (21). R_1 is the weight of the robot system input while R_2 is the weight of the human system input. These weights determine how the robot perceives who is considered leader and who is follower. But, if these weights are adapted properly, it could allow the robot to understand how to be a teacher with the human being the student. Therefore, the system must have a way to change R_2 overtime depending on the interaction force the human has applied to the system. Due to this system being an adaptive control system, R_2 is ever changing. To see how R_2 changes, the equations on page 674-675 in [9] must be understood. These equations are shown below, with $\alpha =$

5000. Equation (22) looks at the force error of the system with the hope it can be used to help minimize the force of the system. Then equation (23) is how R_2 is updated dependent on the partial differential of the force error multiplier, $E = 0.5e_f^T e_f$. However, to make the arithmetic easier, (23) is augmented to be equivalent to (24). Therefore, (25) is needed to be able to update r_2. Equation (25) is the partial differential of equation (22) based on r_2 which due to the force, *f*, being the set as seen in (19). Equation (25) then becomes dependent on equation (20). Then equation (27) is the partial differential of *P* with respect to r_2 from equation (8). From here, the partial differential of $\bar{z}^*$ with respect to r_2 needs to be found so that R_2 can be updated. This can be done by using equation (26).

$$e_f = f - f^* \tag{22}$$

$$\dot{R_2} = -\alpha \frac{\partial E}{\partial R_2} \tag{23}$$

$$\dot{r_2} = -\alpha \frac{\partial E}{\partial r_2} = -\alpha e_f^T \frac{\partial e_f}{\partial r_2} \tag{24}$$

$$\frac{\partial e_f}{\partial r_2} = \frac{1}{2r_2^2}\overline{B_2}^T P\bar{z}^* - \frac{1}{2r_2}\overline{B_2}^T \frac{\partial P}{\partial r_2}\bar{z}^* - \frac{1}{2r_2}\overline{B_2}^T P \frac{\partial \bar{z}^*}{\partial r_2} \tag{25}$$

$$\dot{\bar{z}}^* = \dot{r_2} \frac{\partial \bar{z}^*}{\partial r_2} \tag{26}$$

$$\frac{\partial P}{\partial r_2} = \frac{1}{2}(\bar{A} - \left(\overline{B_1}R_1{}^{-1}\overline{B_1^T} + \overline{B_2}R_2{}^{-1}\overline{B_2^T}\right)P)^{-T} * (\frac{1}{2r_2^2}P\overline{B_2}\overline{B_2^T}P) \tag{27}$$

To update R_2, (27) is inputted into (25) then (24) and (25) are put into (26) to solve for $\frac{\partial \bar{z}^*}{\partial r_2}$. Then this new value can be used to solve (25) and then (24) to finally solve the updated value of R_2. In (28), the way (26) is solved is shown, this is with **Assumption 3: $\dot{\bar{z}}^*$ is not directly related to $\frac{\partial \bar{z}^*}{\partial r_2}$ when it is differentiated with respect to r_2.**

$$\dot{\bar{z}}^* = -\alpha e_f^T \left(\frac{1}{2r_2^2}\overline{B_2}^T P\bar{z}^* - \frac{1}{2r_2}\overline{B_2}^T \frac{\partial P}{\partial r_2}\bar{z}^* - \frac{1}{2r_2}\overline{B_2}^T P\frac{\partial \bar{z}^*}{\partial r_2}\right)\frac{\partial \bar{z}^*}{\partial r_2}$$

$$\dot{\bar{z}}^* = -\alpha e_f^T \left(c - d - v\frac{\partial \bar{z}^*}{\partial r_2}\right)\frac{\partial \bar{z}^*}{\partial r_2} = \left(-C + D + V\frac{\partial \bar{z}^*}{\partial r_2}\right)\frac{\partial \bar{z}^*}{\partial r_2}$$

$$\frac{dy}{dx}\left(\dot{\bar{z}}^* = (C + D)\frac{\partial \bar{z}^*}{\partial r_2} + V\frac{\partial \bar{z}^{*2}}{\partial r_2}\right) \rightarrow 0_{6\times 1} = (C + D) + 2v\frac{\partial \bar{z}^*}{\partial r_2} \tag{28}$$

The goal of this robot's control is to minimize the cost function in (29), to allow for optimal control between the human and the robot.

$$\Gamma = \int_0^\infty c(t)dt$$

$$c(t) = (x - x_d)^T Q_1 (x - x_d) + \dot{x}^T Q_2 \dot{x} + u^T R_1 u + f^T R_2 f \tag{29}$$

This cost function shown in (29), shows the tracking error in the first part, the speed in the second part, the robot's input in the third, and the human's input in the fourth. Therefore, as R_1 and R_2 changes, the leader and follower roles change. If R_2 becomes the bigger of the two weights in the system, then the human becomes the leader. The same is true for the opposite, when R_1 becomes larger, then the robot becomes the leader.

CHAPTER 4

RECONSTRUCTION

After the assumptions and equations are understood, the simulation in MATLAB was reconstructed based on the algorithm process from [9]. In Appendix B, the MATLAB code for the simulation of this system can be seen, with annotations and comments from where in [9] the equation came.

Before the system from [9] can be reconstructed, the understanding of how an impedance system works needs to be done. This understanding of the impedance system comes from article [34]. Article [34] looks at an impedance control model between robot and the environment it is within. Specifically, this model looks at a mass-damping stiffness model and a damping stiffness model. Their cost function that they derived for their models involve speed (for the mass-damping stiffness model), tracking error, and interaction force. Also, in article [34], the k variable that is used for optimal control is a value that changes over time instead of staying constant like some systems. Their models were simulated within MATLAB while also looking at the Linear Quadratic Regulator (LQR) and the desired trajectory compared to the results of the model's trajectory. The results showed that the adaptive impedance model did have aspects that they desired but would need more testing and changes since it was not an ideal model for the robot that was tested, which lead to their next article, [9].

The first part of the system setup is making sure all the constants are the correct dimensions and inputted into the code in the correct order. Ensuring that the model is setup for the specific model that is being looked at. This means that when it comes to setting up the dimensions of x, it needs to be a 2×1 matrix since the end effector is

moving in a 2D plane. At this point, $\dot{x}$ dimensions can be found, being 2×1 as well due to the velocity being in the same plane as the position vector. From here ω can be calculated by (3), which results in a 2×1 vector. Now, the force matrix can also be concluded as a 2×1 since it is also from the 2D plane. This creates $\bar{z}$ to become a 6×1 and since $\dot{z}$ is the future version of $\bar{z}$, that makes $\dot{z}$ a 6×1 as well. Using (1), the dimensions of $\bar{A}$, $\overline{B_1}$, and $\overline{B_2}$ can be determined. From (1), $\bar{A}$ must be a 6×6 with $\overline{B_1}$ and $\overline{B_2}$ being 6×2, so that the dimensions multiply out correctly. Consequently, when looking at (12), *l*×*2m*, *2m*×*l*, and *l*×*m* are said to be zero matrices that are the proper dimensions, but in B_1, a 4×2 matrices, which means the *l*×*m* should be 2×2 matrix to make it a 6×2 matrix. Assuming this is due to $\bar{A}$ being a 6x6 matrix. So, this mean *l*=*m*. Since *l*=*m*, then this also helps with the dimension of *V* and *U*, since on page 673 of article [9] it states $V = \mathbb{R}^{n\times l}$ and $U = \mathbb{R}^{l\times l}$, but looking at (13), *V* and *U* are 2×2 matrices. Leading to Assumption 2: *n*=*l*=m. Now, looking at later works by the authors of [9], the dimensions are *m*×*m* in article [10], fixing this problem of having to figure out the different values of *l*, *m*, and *n*.

Next, the initial values of the time dependent variables can be calculated so that when the for loop runs, there is reference variables for the initial time step. This also tells the system where it is starting. It is important to have these initial values initialized with the desired values, so that the system responds properly. For example, if the initial value of M_d or c_d is changed, then the system will respond different. This can be seen later in this chapter.

Once these values are initialized, the *for* loop can be implemented with all the time dependent variables and the updating of R_2 within this loop. The *for* loop allows the system to run for a set amount of time with a predetermined time step. This simulation is

set to run for 10 seconds with a time step of 10 milliseconds. When this for loop is ran, all the time dependent variables and the system is updated then saved for each time step. In these 10 seconds, the robot is trying to draw a desired path, (in this simulation a circle), while the human is impacting the robot's path by applying a force. As a result, a path that makes the circle look like someone took a bite from the circle. As represented in the graph in Figure 5. The blue path is showing what would happen in real life if the human exerted the same force as seen in (19) while the red path is showing the path the robot desired to take before it adapts to the human's actions. Yet, if the system is to run optimally, with the minimal amount of force excreted by the human, then the path of $\bar{z}^*$ would be the result, as seen in Figure 6. The optimal path is to allow the best human-robot interaction experience with little strain on the person working with the robot.

After all these values are collected, graphs are made to plot the values of the position values of the actual and optimal response data, the force plot, and the gain plots of the system's response. The first set of position graphs are shown in Figure 6. The first force plot that is looked at containing the actual force, optimal force, and force error can be seen in Figure 8. Then the first gain plot of the norm of K_1 and K_2 can be seen in Figure 10.

Yet, there is an issue when trying to re-simulate the results of [9] of the desired path and actual path generated from their simulation. As seen in Figure 5, when the same equations are running as in [9] and the desired path and actual path are looked at, there is a huge difference in the actual paths across the figures until (1) is changed to (1a). This update between (1) and (1a) can be seen in Figure 6.

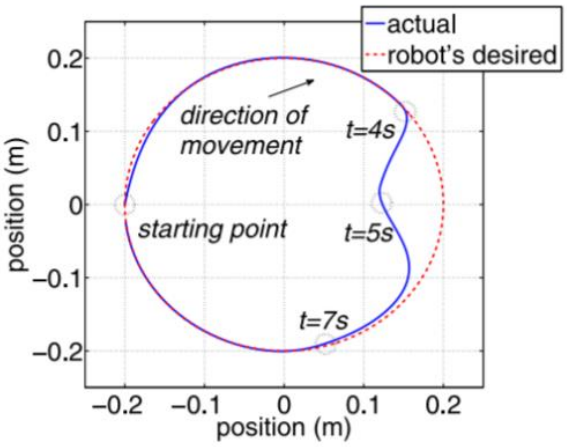

Figure 5 – (Above) *The expected outcome of the graphs from [9]*

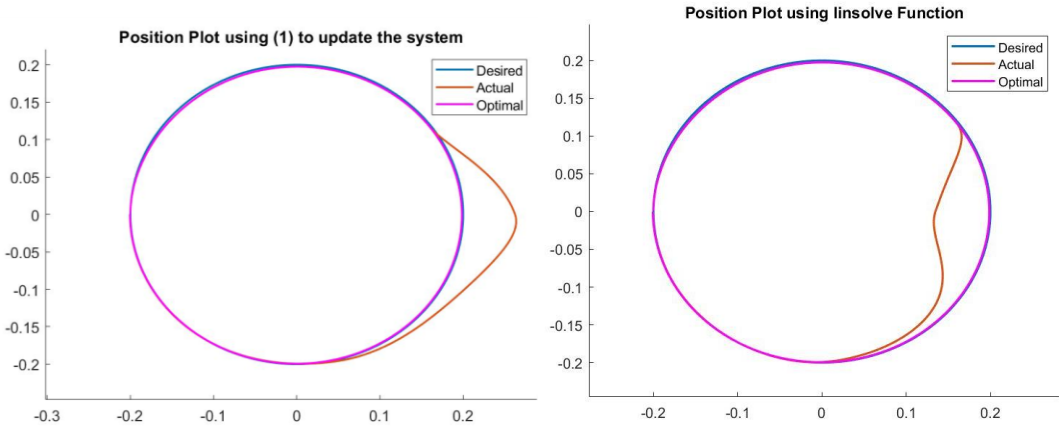

Figure 6 – Position Plot based on equations (1) vs (1a)

These differences raise the question of why the force is pushing to the right instead of the left. In [9], there is no reference as to if the force provided is pushing into the circle or pushing out of the circle when it is applied to the system. However, as seen in Figure 1, the intended goal was to have the force pushing the circle in, to cause a dent-looking effect to the circle. Yet when the information from [9] is applied as shown, the results are not consistent with what is being said. Therefore, the change from (1) to (1a) is made. Due to this change, the resulting figure from the model is shown in Figure 6, and when compared to the desired outcome in Figure 5, the results are consistent with what

was expected. Switching the sign from positive to negative may seem small, but as seen in Figure 6, this is a huge change in the expected results.

If the change made in (1a) is correct, then the force plots in Figure 7 should look exactly like Figure 8. The actual force is given in (19), which is a nonchanging variable. Yet, the optimal force as seen in equation (20), is ever changing. This is due to how R_2 is updating. R_2 updated based on equations (22) through (27). In article [9], it does not specifically state or reference how to solve (26) once (24) and (25) are substituted in. To ensure the minimal force error between the actual force and the optimal force, and to ensure that the optimal system has the minimal force needed to obtain the same path as the actual path, does not have a detailed way of how to update the system. The difficulty that arises, causing this difference in how the optimal system responds, is how (26) is being solved. The equation becomes a quadratic-like equation with matrices when (24), (25), and (27) are substituted into (26), resulting (26) having multiple ways to compute the desired value that is then used to update R_2.

In this thesis paper, one way (26) is solved, is by differentiating it. This is under the assumption that $\dot{\bar{z}}^*$ is only remotely related to $\frac{\partial \bar{z}^*}{\partial r_2}$ when the partial differential of $\bar{z}^*$ is taken in respect to r_2. Under this assumption, (26) can be differentiated and then rearranged to be able to solve using the *linsolve* function in MATLAB. The process of how this differential of (26) can be seen in (28). The results of this technique to update R_2 is seen in Figure 6, 8, and 10. However, this is not the same as the results from [9] as seen in Figures 5 and 7, because there are several differences between the figures due to the information lost when the equation of differentiated. Meaning that the optimal position plot is not following the actual position plot, and the force error is extremely high while

the optimal force is extremely low. Resulting in the optimal system following the robot's desired position path because the system is acting like the human is not inputting any force into the system, which is also seen in Figure 8.

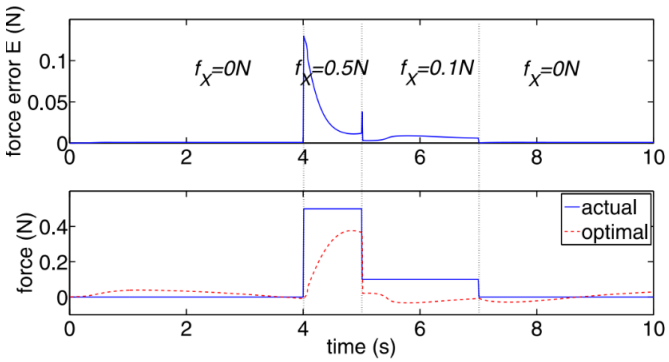

Figure 7 – Results from [9] of Interactive Force and the Force Error

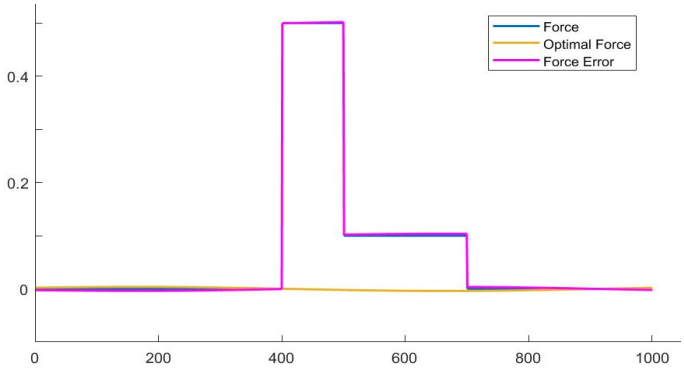

Figure 8 – Interactive Force and Force Error based on differentiating (26)

Another part of this system is looking at the norms of the K_1 and K_2 values. These values show how the gain of each agent of the system is responding based on how R_2 is updating. The values of K_1 and K_2 can be found in equations (6) and (7). The plots of the gain, of K_1, show how the robot responds to the human's force and how the robot adjusts its stiffness based on equation (6). The same can be seen with the gain plot of K_2, which is the response of the human based on the updated value of R_2, this value is calculated in equation (7). As more tests are run, the gain plots become more important as it shows

how the system responds based on the techniques used to update R_2. In [9], the gain plot can be seen in Figure 9, while the system of this paper can be seen in Figure 10. These two plots have major differences, which is due to how the system is updating R_2 and how the robot is responding to the updated weight of the human's actions in the optimal system.

The last plot that is looked at is the plot of the values of R_2 as this variable update. This plot shows how the system is reacting to the interaction force by changing the values of R_2 based on who is leading or following. Due to the robot's weight being set at 0.5, the human's weight, R_2, needs to be lower than 0.5 to allow the robot to head. However as seen in Figure 12, this is not the case with this system. As the system adapts, the R_2 value slowly declines, then spikes when the human applies force at 400 milliseconds before slowly decreasing again. Yet, when Figure 12 is compared to Figure 11 from article [9], the system in Figure 11 is not adapting the system enough to allow the robot to lead the system. As seen in Figure 11, the system allows the robot to lead except when the human applies a force that changes the direction of the robot's desired path. When this happens, the result is the human leading the robot the whole time, making the system not adaptable since it does not change who is leading and who is following based on who needs to lead. For a system to be adaptive, it must automatically change the weight of R_2 to a proper value so that when there is no force from the human, then the robot is considered the leader and when the human does apply force, the system can adapt to allow the human to lead. This way the path the system draws is the path desired by the human instead of the robot. Yet, the way this system is updating R_2, there is no adaption by the system taking place that would allow the robot to lead. This is why when Figure 11 is looked at, the

system can be considered adaptive because the weight of R_2 goes done low enough at times to allow the robot to be considered the leader of the relationship. This happens when the path of the human and the robot are the same.

Therefore, this system is not adapting to the human's actions, because as seen in Figure 8, the optimal force of the human's actions is almost zero. Also, as seen in Figure 6, the optimal system is following the robot's desired path instead of the actual path. Plus, as seen in Figure 12, the human's weight is not adapting to allow the robot to lead. Thus, this technique is not adapting properly and R_2 is not updating using the best possible method. So, other techniques are tested for solving (26) later in Chapter 5.

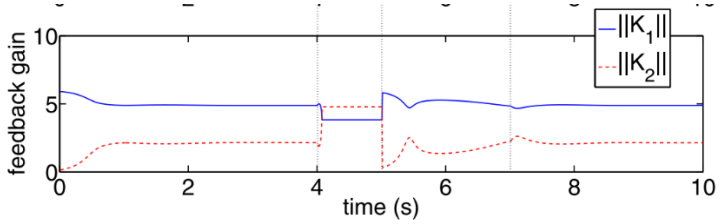

Figure 9 - Article [9] Gain Plot

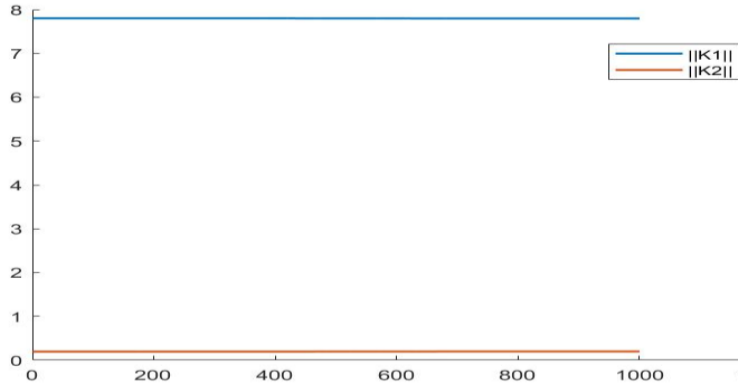

Figure 10 - Gain Plot using (28) to update R_2

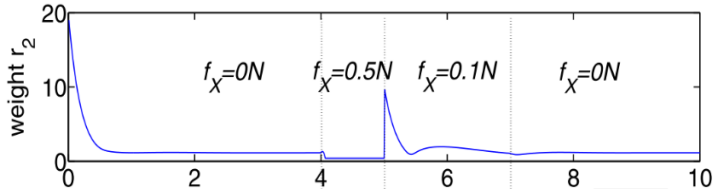

Figure 11 - R_2 Plot from article [9]

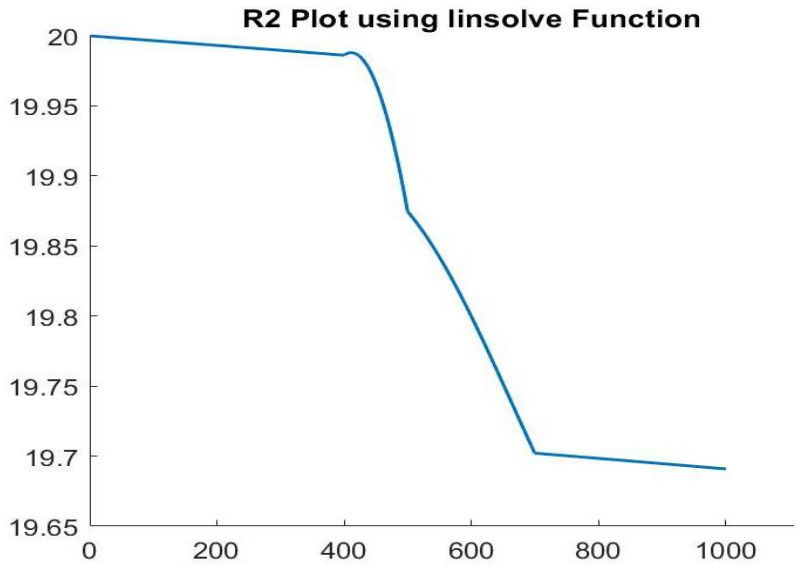

Figure 12 - R_2 Plot as the system adapts

CHAPTER 5
SIMULATIONS

5.1 System Check – Changing the System Constants

Once the simulation is working, keeping how R_2 is updating the same, other tests were performed. These tests mainly looked at the effects of changing the system's mass and damping constants. The first test was halving the mass of the system. The next test halved the damping constant of the system before testing the effects of multiplying the damping constant by 2. These results can be seen below in Figures 13-24.

5.2 Changing the Mass Constant

When the mass of the system was halved, it did not have a huge effect on the force plot, as seen in Figure 14, but there was a noticeable impact to the position plot seen in Figure 13. Figure 13 shows this position plot, and if it is compared to Figure 6, there is a noticeable difference in the path of the actual and optimal plot. The reason the mass change to the system only showed a major change in the position graph and not the force graph is due to how easily it is to move with the robot. Since there is less impedance when the human works with the robot, the path this system takes is greatly affected by the same amount of force the human applied to the robot. Thus, the indent in the circle in Figure 13 is much larger, the force is the same but the human and robot must move only half the weight. The same can be seen with the optimal plot in Figure 13, however there is not as much of a noticeable impact as seen in the actual plot in Figure 13. This is also why the force plot in Figure 14 does not show a great impact, since the actual force being applied is not changed. The only effect the mass has on the force is when calculating the optimal force, and that effect is minor. When the gain plot is looked

at in Figure 15, the norm values of K_1 and K_2 are halved when compared to Figure 10. Meaning that the mass constant is affecting the system as expected, because when the gains are calculated, it is calculated based on the $\bar{B}$ matrices., which was halved when M_d is halved.

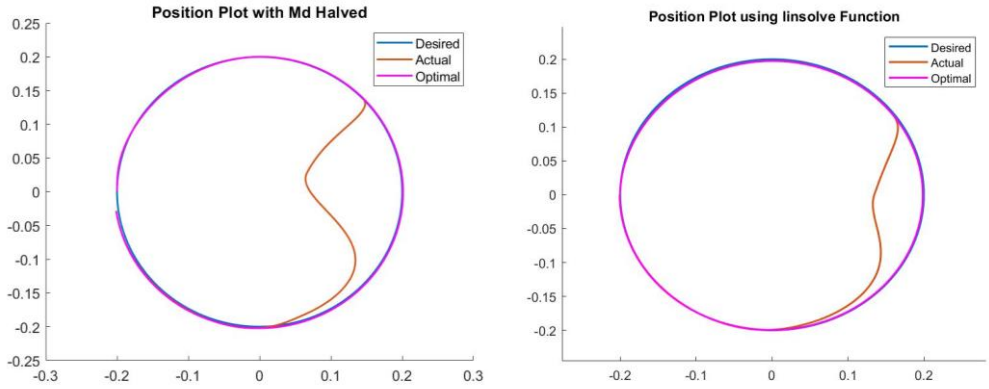

Figure 13 – Position plot when M_d is halved vs original system

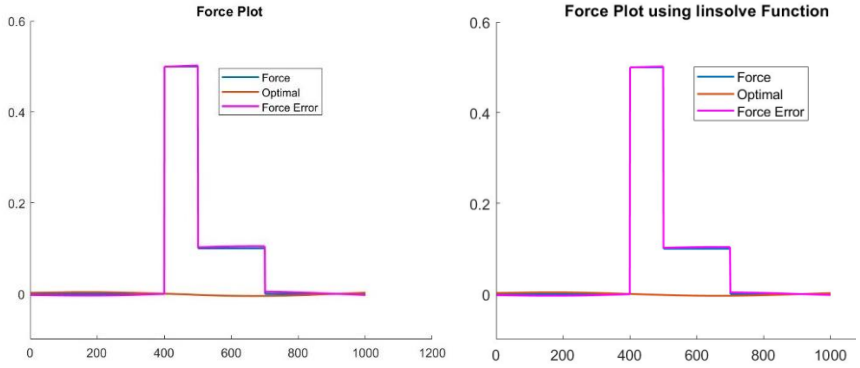

Figure 14 – Force plot when the M_d is halved vs original system

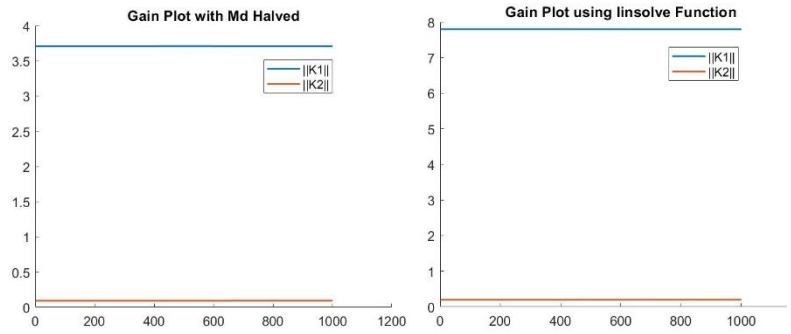

Figure 15 - Gain plot when M_d is halved vs original system

5.3 Changing the Damping Constant

The next test conducted was to see how the damping constant affects the system. In Figures 16 and 17, the position plot and force plot can be seen when the damping constant is halved. When Figure 16 is compared with Figure 6, there is not a noticeable change, the same with when Figure 17 is compared to Figure 8. Then if the gain plot of the system when C_d is halved, there is not a noticeable difference when compared to Figure 10.

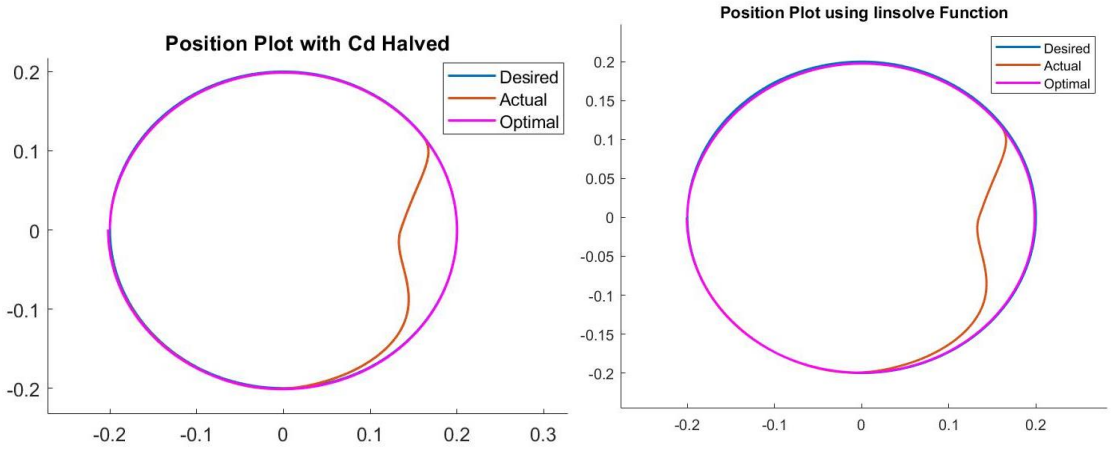

Figure 16 - Position plot with C_d halved vs original system

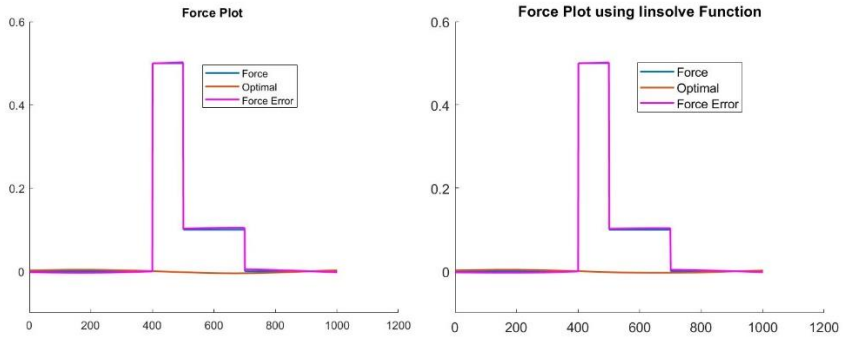

Figure 17 - Force plot with C_d halved vs original system

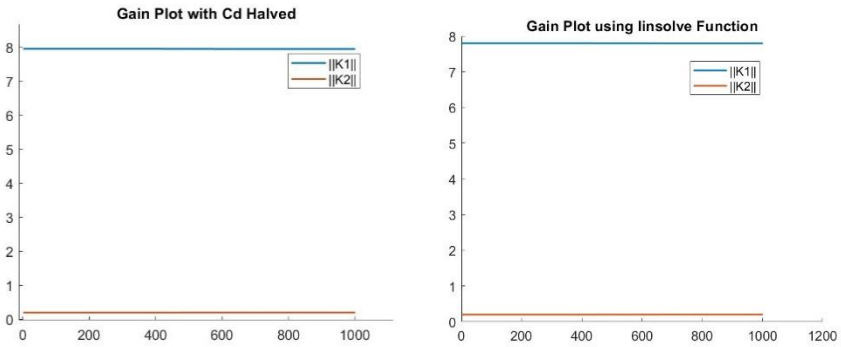

Figure 18 - Gain plot with C_d halved vs original system

Yet, when the C_d value is doubled, as seen in Figures 19, 20, and 21 are compared to Figures 6, 8, and 10, there is a noticeable difference. This is due to the damping constant has an exponential type of effect on the system. If the constant is only halved, then there will not be a big change in the plots, but if there is a larger change in the damping constant, like doubling it, then there is a more noticeable difference.

When a damping constant of a system is changed, it affects how smoothly the system will respond and how quickly. This could be explained like a car shock. When the shock is working properly, then when the car goes over a bump, there is little bounce before the car settles back to equilibrium. However, if the shock is not working properly, and the car goes over the bump, there is a longer period the car spends bouncing before returning to the equilibrium point.

As seen in Figure 19, when the damping constant is doubled, the effect of this change to the position plot is greater. This is due to the system wanting to oscillate more so than normal, resulting in larger change in position. The affects the force plot, since it now requires more force to keep the system on the desired path. This can be seen in the optimal plot line in Figure 20. The optimal force is growing, creating a smaller force error. But as before the actual force the human is applied was not changed. Allowing the

force to be used a constant variable in all the tests. Equally, when the gain plot is looked at in Figure 21, there is only a slight change in how K_1 is calculating at. The change is not huge, like how the mass constant changed this plot, but it is affected because the gain is affected by how R_2 is updating which is also affected by the value of C_d.

These tests show how changing the constants in the system will affect how the robot and human work together. It could affect the force required to work with the robot, or it could affect the overall path of the system. These settings for a system are critical to understand, so that the experience when working with a robot will be the best possible for the human and the desired outcome.

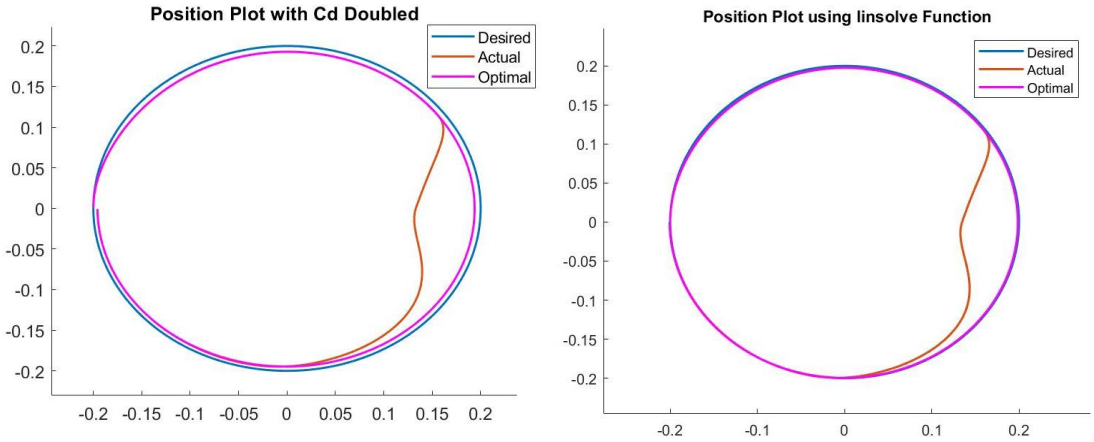

Figure 19 - Position plot with C_d doubled vs original system

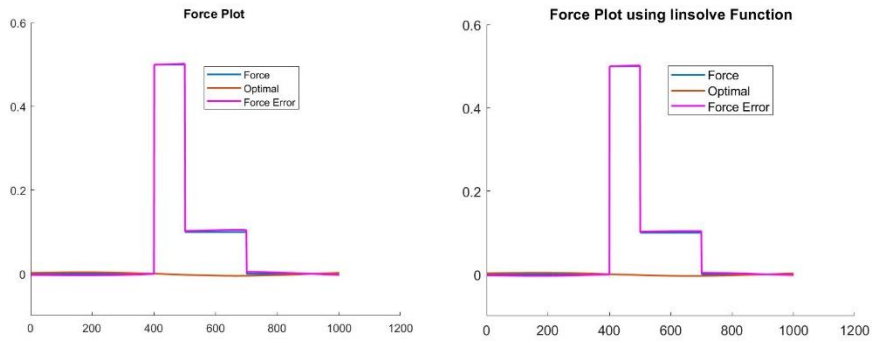

Figure 20 - Force plot with C_d doubled vs original system

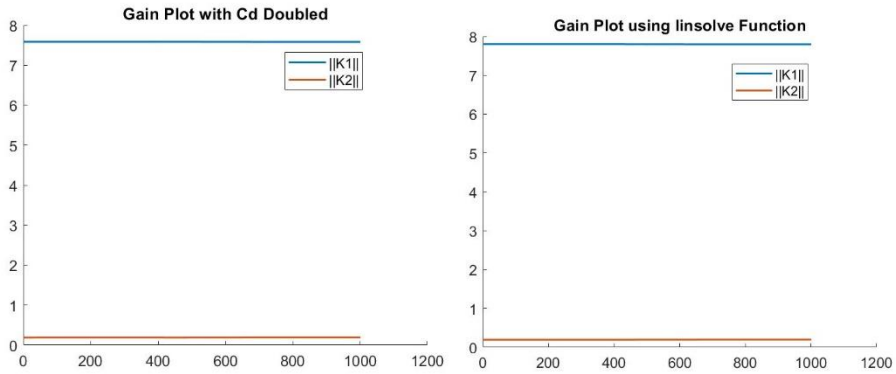

Figure 21 - Gain Plot with C_d doubled vs original system

5.4 Techniques Tested to Update R_2

After these tests are done to ensure the system is working properly, more techniques of updating R_2 are tested.

5.5 Care and nonlinear *fsolve* functions

The next technique that was tried was using the *care* function to solve $\frac{\partial P}{\partial r_2}$, to then use to solve for $\frac{\partial \bar{z}^*}{\partial r_2}$ in equation (26) with (24) and (25) substituted in. (26) was then solved by using the nonlinear *fsolve* function. This code can be found in Appendix C and with the function code needed to use the nonlinear *fsolve* function can be found in Appendix D. The outcome of this way of updating R_2 when compared to Figures 6, 8, and 10, are the same. However, the R_2 plot is completely different from Figure 12. Instead of R_2 slowly decay until a force is applied, the weight of R_2 does not change until the force from the human is applied at 400 milliseconds. This change is due to using *fsolve* function instead of the substitution technique shown in Chapter 4. Yet, this plot does not allow the robot to lead either, as seen in Figure 22.

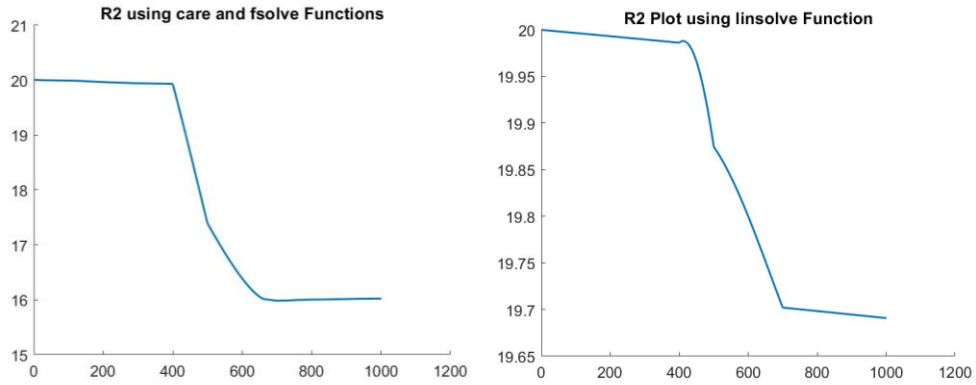

Figure 22 - R_2 Plot using care and fsolve function vs linsolve function

5.6 Basic *fsolve* function

Due to this technique not changing the outcome of the plots, another technique was used. This technique, the traditional *fsolve* function was tried with using the original way of solving for $\frac{\partial P}{\partial r_2}$ as seen in (27). This code can be found in Appendix E. These results show the same results as seen in Figures 6, 8, 10, and 22. So this technique is not the technique used to update R_2.

5.7 Difference Technique using Equation (30)

The next technique that's tested is using a way to update $\frac{\partial \bar{z}^*}{\partial r_2}$ that was recommended by the author of article [9]. Instead of using substitution of equations (24), (25), and (27) into (26), it was suggested to use difference when solving (26) to solve for $\frac{\partial \bar{z}^*}{\partial r_2}$. The equation that is used is (30). The code for this technique is the same as Appendix C, but with equation (30) uncommented and the *fsolve* function commented out.

$$\frac{\partial \bar{z}^*}{\partial r_2} = \frac{\dot{\bar{z}}^*}{\dot{r_2}} \tag{30}$$

The results from this technique of updating R_2 is the same as seen in Figures 6 and 8, showing that this is also not the best way of updating R_2. Yet, when the Gain Plot, Figure 23, is compared to Figure 10, there is a noticeable difference, but not huge. Also, the R_2 plot is different than Figures 11, 12, and 22, this can be seen in Figure 24. The first 400 milliseconds of this system is the same as Figure 22, but then this is a drastic drop before slowly increasing past the original value of R_2 and then settling to the weight of 24. Due to the weight of R_2 becoming larger rather than smaller, the system never allows the robot to lead, meaning this system cannot be considered adaptive. Thus, this technique is not the technique used in article [9] and this is not the best way to adapt the control of the system to minimize the required interaction force and the force error. So, another technique is needed to be able to adapt this system in such manner that the optimal force and force error is minimal, and the optimal path follows the path the human desires instead of the robot's desire. This way the robot is adapting to the human's actions in the best possible way with this type of adaptive control.

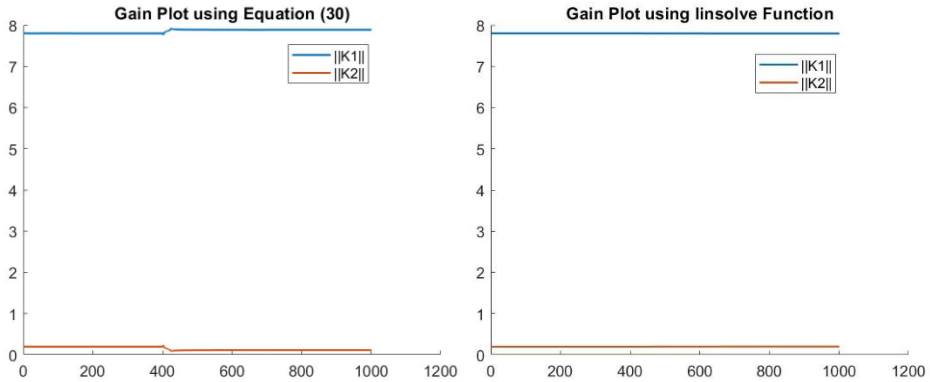

Figure 23 - Gain Plot - Updating R_2 using equation 30 vs linsolve function

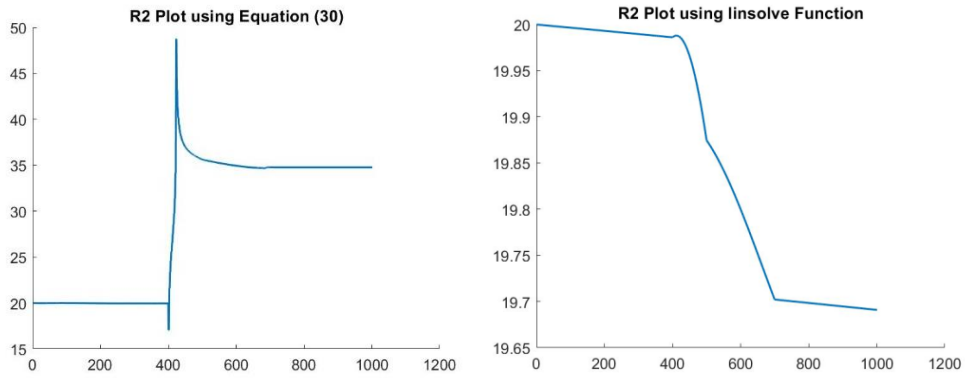

Figure 24 - R_2 Plot using equation 30 vs linsolve function

CHAPTER 6

CLOSING

6.1 Limitations and Challenges

As discussed in Chapter 2, the robot's adaption is based on the interaction force between the human and the robot. This leads to limitations of this type of control. These limits are that this type of control can only be used when it involves the robot using the interaction force in the system. Due to the game theory system setup, the robot has its own task it wants to achieve and does not change if the human is moving with the robot or in the system at all, but if the human does decide to work with the robot and apply force to the robot, it will adapt based on the force applied.

For example, if a manufacturing robot is putting the door panels onto a truck, and there is a manager standing in the regular path of the robot. The robot could be pushed away by the manager to tell the robot to go around him and has the abilities to adapt to go around that manager because it of its type of system control it has in its safety protocol programming. However, there are limits to this example because it must have some type of haptic feedback to trigger the adaption, such as the force applied by the manager. This means that the manager had to have seen the robot and pushed it away for this type of system adaption to work. If the manager had not seen the robot and pushed the robot away, the robot could have hit the manager, causing safety issues. So, any system that rely on haptic feedback or does have this type of two-agent game theory will be able to use this type of control. But if the system uses any other type of feedback control or is not setup as a game theory system, then this algorithm can not be used for that system.

Also, the system needs to be calibrated to the agents in the system, because if the human applies extreme force to move the robot, it could throw the robot pass the path the human wanted because the weights, Q's and R's, were not calibrated properly to that more forceful human. Thus, to stray from coordination issues, the system must also be calibrated before use, to ensure that the system works properly with the robot being used and the human working with the robot.

Due to the complexity of how R_2 is updated, there are multiple ways of solving (26) and updating R_2 based on the force that the human is putting on the system. Several techniques were tested, however due to the complexity, none of these techniques made the system respond as seen in [9]. As seen in all the position plots, the paths are almost the same as what is seen in [9], but the force plots have a huge discrepancy between the optimal force and the force error. This is due to how the R_2 is updating and making the system respond to the human's actions. This is because, the gain plots from the system also do not match the gain plots from [9]. With each new technique that is tested the gain plots start to show a larger response than before to the human's actions. As a result, each test is making progress towards the technique that is used in [9]. To be able to have the system respond like seen in [9], further research will be needed.

6.2 Applications

As stated above, this type of adaptive control can be used with robots that are dependent on interaction force. It is limited to only being able to be used with force dependent systems and systems that have at least two agents working together. Nonetheless, there are multiple applications for which this type of control can be used. Once this algorithm is updating properly with minimal force error and smooth response

to the human's actions, this algorithm could be used in physical therapy, industry, and for teaching.

This type of control could be implemented in physical therapy robots that are working with people to help retrain muscle control, just like what is tested in article [39]. If a person hurt their shoulder or broke their rotator cuff, they would be required to go to physical therapy weekly and stretch daily. If the person is not strong enough to pull the one arm up with the other arm, then a machine or robot could be used to help that person. When a situation like this happens, a robot could react to the person in a similar way to what would happen with this algorithm. This would become useful to recovering individuals, since the algorithm could be adapted to help the person reach their recovery goals.

Another way this algorithm could be used is in industry. When a robot is used on an assembly line, it may need to move a large object to a specified location like seen in [40]. Unknowing to it, there could be an obstacle in the way. If the robot has this algorithm as part of its safety protocol, when a human or another object puts force on the system, the robot would adapt to either stop or move in the opposite direction of the force applied, avoiding the obstacle in its way.

The third application of this type of control is that it could be used for teaching a person. This type of application is very broad. Adaptive control could be used in robots to teach people how to dance (as seen in article [18]) or could be used to teach children with disabilities how to regain their ability to walk. The only limits of this algorithm are that it is dependent on the interaction force and being a two-agent system.

6.3 Future Work

Work will be needed in the future to continue to find a more suitable way to update R_2, (24)-(27). Future work will be required on physical experiments using a algorithm to adapt a robot's response to working with a person. For example, using a UR5 robot working with a human to testing this algorithm by testing different desired paths for each agent within the partnership. As well as seeing if the system adapts to the human's path with minimal amount of force needed. These findings could prove that this algorithm can be used in industry to allow for better ergonomics for human workers or to help in physical therapy. There needs to be more research done with this algorithm before it is ready to use.

6.4 Conclusion

In conclusion, physical HRI can have a positive impact in several areas in life. The greatest impact will come from how the system is defined and therefore how the robot will respond to the human relationship. Once the flaws are smoothed out, the algorithm can be improved, and the best results will be seen. In this research, it was learned that in the future a robot can be programmed to adapt to its surroundings and respond accordingly based on the feedback within the adaptive control system. However, if this system is not done properly, then the robot will not adapt to its surroundings. Thus, more research is needed before this type of control should be implemented.

REFERENCES

[1] B. Robins, K. Dautenhahn, R. Boekhorst and A. Billard, "Robotic assistants in therapy and education of children with autism: can a small humanoid robot help encourage social interaction skills?", *Universal Access in the cxcxvInformation Society*, vol. 4, no. 2, pp. 105-120, 2005.

[2] R. Chemuturi, F. Amirabdollahian and K. Dautenhahn, "Adaptive training algorithm for robot-assisted upper-arm rehabilitation, applicable to individualised and therapeutic human-robot interaction", *Journal of NeuroEngineering and Rehabilitation*, vol. 10, no. 1, p. 102, 2013.

[3] C. Liu, K. Conn, N. Sarkar and W. Stone, "Online Affect Detection and Robot Behavior Adaptation for Intervention of Children With Autism", *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 883-896, 2008.

[4] D. Kang and H. Choi, "Robot Task Planning for Mixed-Initiative Human Robot Interaction in Home Service Robot", *IEEE*, 2018.

[5] M. Khoramshahi and A. Billard, "A dynamical system approach to task-adaptation in physical human–robot interaction," *Autonomous Robots*, vol. 43, no. 4, pp. 927–946, 2018.

[6] Y. Li, K. Tee, R. Yan, W. Chan, Y. Wu and D. Limbu, "Adaptive Optimal Control for Coordination in Physical Human-Robot Interaction", pp. 20-25, 2015.

[7] B. Lacevic and P. Rocco, "Kinetostatic Danger Field - a Novel Safety Assessment for Human-Robot Interaction", pp. 2169-2174, 2010.

[8] C. Liu and M. Tomizuka, "Robot Safe Interaction System for Intelligent Industrial Co-Robot", pp. 1-12, 2018.

[9] Y. Li, K. Tee, W. Chan, R. Yan, Y. Chua and D. Limbu, "Continuous Role Adaptation for Human–Robot Shared Control", *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 672-681, 2015.

[10] Y. Li, K. Tee, R. Yan, W. Chan and Y. Wu, "A Framework of Human–Robot Coordination Based on Game Theory and Policy Iteration", *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1408-1418, 2016.

[11] A. Peca, R. Simut, S. Pintea, C. Costescu, and B. Vanderborght, "How do typically developing children and children with autism perceive different social robots?," *Computers in Human Behavior*, vol. 41, pp. 268–277, 2014.

[12] D. Losey, C. McDonald, E. Battaglia and M. O'Malley, "A Review of Intent Detection, Arbitration, and Communication Aspects of Shared Control for Physical

Human–Robot Interaction", *Applied Mechanics Reviews*, vol. 70, no. 1, p. 010804, 2018.

[13] S. Albrecht, K. Ramirez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich and M. Beetz, "Imitating human reaching motions using physically inspired optimization principles", pp. 602-607, 2011.

[14] A. L. Thomaz and C. Chao, "Turn-Taking Based on Information Flow for Fluent Human-Robot Interaction," *AI Magazine*, vol. 32, no. 4, p. 53, 2011.

[15] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schaeffer, and E. Burdet, "Human-Like Adaptation of Force and Impedance in Stable and Unstable Interactions," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 918–930, 2011.

[16] Y. Wang, Y. Sheng, J. Wang, and W. Zhang, "Optimal Collision-Free Robot Trajectory Generation Based on Time Series Prediction of Human Motion," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 226–233, 2018.

[17] A. Sawers and L. Ting, "Perspectives on human-human sensorimotor interactions for the design of rehabilitation robots", *Journal of NeuroEngineering and Rehabilitation*, vol. 11, no. 1, p. 142, 2014.

[18] R. Ros, I. Baroni, and Y. Demiris, "Adaptive human–robot interaction in sensorimotor task instruction: From human to robot dance tutors," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 707–720, 2014.

[19] M. Edwards, "Robots in industry: An overview," *Applied Ergonomics*, vol. 15, no. 1, pp. 45–53, 1984.

[20] W. R. Tanner, "Industrial Robots and Flexible Manufacturing," *Robotics and Factories of the Future*, pp. 435–441, 1984.

[21] N. Jarrassé, V. Sanguineti, and E. Burdet, "Slaves no longer: review on role assignment for human–robot joint motor action," *Adaptive Behavior*, vol. 22, no. 1, pp. 70–82, 2013.

[22] M. Sharifi, S. Behzadipour, and G. Vossoughi, "Nonlinear model reference adaptive impedance control for human–robot interactions," *Control Engineering Practice*, vol. 32, pp. 9–27, 2014.

[23] W. Huo, J. Huang, W. Xu, S. Mohammed, and Y. Amirat, "Control of upper-limb power-assist exoskeleton based on motion intention recognition," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 04, 2015.

[24] D. Copaci, E. Cano, L. Moreno, and D. Blanco, "New Design of a Soft Robotics Wearable Elbow Exoskeleton Based on Shape Memory Alloy Wire Actuators," *Applied Bionics and Biomechanics*, vol. 2017, pp. 1–11, 2017.

[25] C. Jarrett and A. J. Mcdaid, “Robust Control of a Cable-Driven Soft Exoskeleton Joint for Intrinsic Human-Robot Interaction,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 9, no. 1, pp. 976–986, 2001.

[26] J. E. Young, JaYoung Sung, A. Voida, E. Sharlin, T. Igarashi, H. I. Christensen, and R. E. Grinter, “Evaluating Human-Robot Interaction - Focusing on the Holistic Interaction Experience,” *International Journal of Social Robotics*, pp. 53–67, Jan. 2011.

[27] D. V. Lu and W. D. Smart, “Human-robot interactions as theatre,” *2011 Ro-Man*, 2011.

[28] C. Breazeal, “Social Interactions in HRI: The Robot View,” *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 181–186, 2004.

[29] E. Al-Gallaf, K. A. Mutib, and H. Hamdan, “Artificial neural network dexterous robotics hand optimal control methodology: grasping and manipulation forces optimization,” *Artificial Life and Robotics*, vol. 15, no. 4, pp. 408–412, 2010.

[30] T. Belpaeme, P. E. Baxter, R. Read, R. Wood, H. Cuayáhuitl, B. Kiefer, S. Racioppa, I. Kruijff-Korbayová, G. Athanasopoulos, V. Enescu, R. Looije, M. Neerincx, Y. Demiris, R. Ros-Espinoza, A. Beck, L. Cañamero, A. Hiolle, M. Lewis, I. Baroni, M. Nalin, P. Cosi, G. Paci, F. Tesser, G. Sommavilla, and R. Humbert, “Multimodal Child-Robot Interaction: Building Social Bonds,” *Journal of Human-Robot Interaction*, vol. 1, no. 2, 2013.

[31] F. Jimenez, T. Yoshikawa, T. Furuhashi, M. Kanoh, and T. Nakamura, “Feasibility of Collaborative Learning and Work Between Robots and Children with Autism Spectrum Disorders,” *New Frontiers in Artificial Intelligence Lecture Notes in Computer Science*, pp. 454–461, 2017.

[32] S. Costa, F. Soares, C. Santos, A. Pereira, and M. Moreira, “Robot Lego & Trastorno del Espectro Autista: Una asociación posible? || Lego Robots & Autism Spectrum Disorder: a potential partnership?,” *Revista de Estudios e Investigación en Psicología y Educación*, vol. 3, no. 1, p. 50, 2016.

[33] T. Basara and G. J. Olsder. (1998). *Dynamic Noncooperative Game Theory*. 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. [Online]. Available: http://epubs.sian.org/doi/abs/10.1137/1.9781611971132

[34] S. Ge, Y. Li and C. Wang, "Impedance adaptation for optimal robot–environment interaction", *International Journal of Control*, vol. 87, no. 2, pp. 249-263, 2013.

[35] A. Mörtl, M. Lawitzky, A. Kucukyilmaz, M. Sezgin, C. Basdogan, and S. Hirche, “The role of roles: Physical cooperation between humans and robots,” *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1656–1674, 2012.

[36] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*, Cambridge University Press, 2007.

[37] N. Jarrasse, T. Charalambous, and E. Burdet, "A Framework to Describe, Analze and Generate Interactive Motor Behaviors", *PLoS ONE,* vol. 7, no. 11, p. e49945, 2013.

[38] A. Wagner, "Using Games to Learn Games: Game-Theory Representations as a Source for Guided Social Learning," *Social Robotics Lecture Notes in Computer Science*, pp. 42–51, 2016.

[39] K. Tsiakas, M. Dagioglou, V. Karkaletsis, and F. Makedon, "Adaptive Robot Assisted Therapy Using Interactive Reinforcement Learning," *Social Robotics Lecture Notes in Computer Science*, pp. 11–21, 2016.

[40] F. Nagata and K. Watanabe, "Controller design for industrial robots and machine tools," 2013.

APPENDIX A

SYSTEM'S VALUES

The values for each equation are given below as they are referenced in the equations referenced. These values are unedited from [9]. However, if the variable is ever changing as time goes one, those values are not recorded here.

(1) $$\dot{\bar{z}}(t) = \bar{A}\bar{z}(t) + \overline{B_1}u(t) + \overline{B_2}f(t)$$

$$\bar{A} = \begin{bmatrix} A & 0_{2mxl} \\ 0_{lx2m} & U \end{bmatrix}$$

$$\overline{B_1} = \overline{B_2} = \begin{bmatrix} B_1 \\ 0_{lxm} \end{bmatrix}$$

$$A = \begin{bmatrix} 0_{mxm} & I_{mxm} \\ 0_{mxm} & -{M_d}^{-1}C_d \end{bmatrix}$$

$$B_1 = B_2 = \begin{bmatrix} 0_{mxm} \\ {M_d}^{-1} \end{bmatrix}$$

$$M_d = I_{2x2}$$

$$C_d = I_{2x2}$$

APPENDIX B

MATLAB CODE

```
clear
close all

Md=[eye(2)]; %2x2 pg677
Cd=[eye(2)]; %2x2 pg677
a=5000; %pg 677
U=[0 pi/5;-pi/5 0]; %2x2 pg677
V=(1/pi)*[eye(2)]; %2x2 pg677
A=[zeros(2) eye(2);zeros(2) -Md^-1*Cd]; %4x4 pg673
B1=[zeros(2);Md^-1]; %4x2 pg673
B2=B1;
Abar=[A zeros(4,2);zeros(2,4) U]; %6x6 pg674
Bbar1=[B1;zeros(2)]; %6x2 pg674
Bbar2=Bbar1;
Q1=50*[eye(2)]; %2x2 pg677
Q2=[eye(2)]; %2x2 pg677
R1=0.5*[eye(2)]; %2x2 pg677
R2=20*[eye(2)]; %2x2 pg677
VT=transpose(V); %or could do V'
 Q=[Q1 zeros(2) -Q1*V;zeros(2) Q2 zeros(2);-VT*Q1 zeros(2) VT*Q1*V]; %6x6 pg674
 dt=0.01;
t=0:dt:10; %pg 677 %time vector length of sim - start:timestep:end

f=zeros(max(size(t)),1); %pg 677 %linear function-     force=f(t)
f(401:500)=0.5; %pg677 %force vector from 4 to 5 sec
f(501:700)=0.1; %pg677 %force vector from 5 to 7 sec
f(701:1001)=0; %pg677 %force vector from 7 to 10 sec
fy=0;

xd=[-0.2*cos((pi/5)*0),0.2*sin((pi/5)*0)]';
x=xd;
xdot=[(sin((pi*0)/5))/25,(pi*cos((pi*0)/5))/25]'; %differential of xd
%pg 673 finding w=auxiliary state
%xd=V*w; %pg673 desired trajectory
w=linsolve(V,xd);
wdot=U*w; %pg673
 %Finding P eq 12 pg674
P=care(Abar,(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')./2,Q);
 %equation 26 pg 675
K1=0.5*(R1^-1)*Bbar1'*P;
K2=0.5*(R2^-1)*Bbar2'*P;
 z=[x;xdot]; %equation given on pg 673
zbar=[z;w]; %zbar initial pg 673
zbarstar=zbar; %6x1
 r2=20; %initial scalar value
 for k=1:length(t) %algorithm 1 pg 675 step of milliseconds
            %caluate f from pg677
               f=zeros(max(size(t(k))),1); %pg 677 %linear function-     force=f(t)
               f(1:400)=0;
               f(401:500)=0.5; %pg677 %force vector from 4 to 5 sec
               f(501:700)=0.1; %pg677 %force vector from 5 to 7 sec
               f(701:1001)=0; %pg677 %force vector from 7 to 10 sec
               fy=0;
               ft=[f(k);fy];
```

```
    xd=[-0.2*cos((pi/5)*t(k));0.2*sin((pi/5)*t(k))]; %page 677
%pg 673 finding w=auxiliary state
%xd=V*w; %pg673 desired trajectory equation 6
    w=linsolve(V,xd);
    wdot=U*w; %pg673
    %Finding P eq 12 pg674
    P=care(Abar,(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')./2,Q); %solved using
equation A-9 from page 282 of reference 31 in article
    %solving for u equation 24 pg 675
    u=-0.5*(R1^-1)*Bbar1'*P*zbar;
    %equation 26 pg 675
    K1=0.5*(R1^-1)*Bbar1'*P;
    K2=0.5*(R2^-1)*Bbar2'*P;
 %Pg 674
    ustar=-0.5*(R1^-1)*(Bbar1')*P*zbarstar; %equation 10 pg 674
    fstar=-0.5*(R2^-1)*(Bbar2')*P*zbarstar; %equation 11 pg 674
 %Update R2 Pg 675 equations 17-23
     X=Abar-(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')*P;
    Y=-(1/r2^2)*P*Bbar2*Bbar2'*P;
%Equation 22
    dP=0.5*(X'^-1)*Y;
%equation 13
    zbarstardot=Abar*zbarstar+Bbar1*ustar-Bbar2*fstar;
%pg 674
    ef=ft-fstar;
    %equations 23 with 18 and 17 substituted in
        %zbarstardot=r2dot*dzbarstar
            %r2dot=-a*ef'*def %17
            %def=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar)-
((1/(2*r2))*Bbar2'*P*dzbarstar) %18
        %zbarstardot=-a*ef'*((1/(2*r2^2))*Bbar2'*P*zbarstar)-
((1/(2*r2))*Bbar2'*dP*zbarstar-((1/(2*r2))*Bbar2'*P*dzbarstar)*dzbarstar;
    E=-a*ef';
    g=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar);
    c=E*g;
    v=E*(-(1/(2*r2))*Bbar2'*P);
    %zbarstardot==(E*(G-v*dzbarstar))*dzbarstar;
%     D=2*eye(6)*[v;v;v;v;v;v]+c*eye(6);
%     dzbarstar=linsolve(D,zbarstardot);
%equation 18
    def=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar)-
((1/(2*r2))*Bbar2'*P*dzbarstar);
%pg 674
    ef=ft-fstar;
%equation 17
    r2dot=-a*ef'*def;
    r2=r2+r2dot*dt;
    if r2<0.4
            r2=0.4;
    end
     R2=r2*eye(2);
%solving for zbarstar equation 12 pg 674
    zbarstardot=Abar*zbarstar+Bbar1*ustar-Bbar2*fstar;
    zbarstar=zbarstar+zbarstardot*dt;
    %solving for zbar equation 7 -- results in desired path since x=xd
```

```
                zbardot=Abar*zbar+Bbar1*u-Bbar2*ft; %makes zbar a 6x1
                zbar=zbar+zbardot*dt; %new zbar, makes a 6x1
                %Storing values
                zbarstarresult(:,k)=zbarstar;
                xdresult(:,k)=xd;
                fstarresult(:,k)=fstar;
                zbarresult(:,k)=zbar;
                efresult(:,k)=ef;
                fstarresult(:,k)=fstar;
                K1result(:,k)=norm(K1);
                zbarstardotresult(:,k)=zbarstardot;
                zbardotresult(:,k)=zbardot;
                K2result(:,k)=norm(K2);
                r2results(:,k)=r2;
                ustarresult(:,k)=ustar;
        end

figure
hold on
plot(xdresult(1,:),xdresult(2,:))
plot(zbarresult(1,:),zbarresult(2,:))
plot(zbarstarresult(1,:),zbarstarresult(2,:))
hold off

figure
hold on
plot(f)
plot(fstarresult(1,:))
plot(efresult(1,:))
%plot(r2results)
hold off

figure
hold on
plot(K1result)
hold off

figure
hold on
plot(K2result)
hold off

figure
hold on
plot(K1result)
plot(K2result)
hold off
```

APPENDIX C

CODE USING THE *fsolve* FUNCTION

```
clear
close all
%% ssetup intial values and constants
Md=[eye(2)]; %2x2 pg677
Cd=[eye(2)]; %2x2 pg677
a=5000; %pg 677
U=[0 pi/5;-pi/5 0]; %2x2 pg677
V=(1/pi)*[eye(2)]; %2x2 pg677
A=[zeros(2) eye(2);zeros(2) -Md^-1*Cd]; %4x4 pg673
B1=[zeros(2);Md^-1]; %4x2 pg673
B2=B1;
Abar=[A zeros(4,2);zeros(2,4) U]; %6x6 pg674
Bbar1=[B1;zeros(2)]; %6x2 pg674
Bbar2=Bbar1;
Q1=50*[eye(2)]; %2x2 pg677
Q2=[eye(2)]; %2x2 pg677
R1=0.5*[eye(2)]; %2x2 pg677
%% Orginal Start Value of R2
R2=20*[eye(2)]; %2x2 pg677
%% Change Start value of r2
r2=19;
%%
Q=[Q1 zeros(2) -Q1*V;zeros(2) Q2 zeros(2);-V'*Q1 zeros(2) V'*Q1*V]; %6x6 pg674

dt=0.01;
t=0:dt:10; %pg 677 %time vector length of sim - start:timestep:end

f=zeros(max(size(t)),1); %pg 677 %linear function-     force=f(t)
f(401:500)=0.5; %pg677 %force vector from 4 to 5 sec
f(501:700)=0.1; %pg677 %force vector from 5 to 7 sec
f(701:1001)=0; %pg677 %force vector from 7 to 10 sec
fy=0;

xd=[-0.2*cos((pi/5)*0),0.2*sin((pi/5)*0)]';
x=xd;
xdot=[(sin((pi*0)/5))/25,(pi*cos((pi*0)/5))/25]'; %differential of xd
%pg 673 finding w=auxiliary state
%xd=V*w; %pg673 desired trajectory
w=linsolve(V,xd);
wdot=U*w; %pg673
%Finding P eq 12 pg674
P=care(Abar,(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')./2,Q);
%equation 26 pg 675
K1=0.5*(R1^-1)*Bbar1'*P;
K2=0.5*(R2^-1)*Bbar2'*P;
 z=[x;xdot]; %equation given on pg 673
zbar=[z;w]; %zbar intial pg 673
zbarstar=zbar; %6x1
 %% for loop
for k=1:length(t) %algorithm 1 pg 675 step of milliseconds
%caluate f from pg677
   f=zeros(max(size(t(k))),1); %pg 677 %linear function-     force=f(t)
   f(1:400)=0;
   f(401:500)=0.5; %pg677 %force vector from 4 to 5 sec
   f(501:700)=0.1; %pg677 %force vector from 5 to 7 sec
```

```
    f(701:1001)=0; %pg677 %force vector from 7 to 10 sec
    fy=0;
    ft=[f(k);fy];
    xd=[-0.2*cos((pi/5)*t(k));0.2*sin((pi/5)*t(k))]; %page 677
%pg 673 finding w=auxiliary state
%xd=V*w; %pg673 desired trajectory equation 6
    w=linsolve(V,xd);
    wdot=U*w; %pg673
%Finding P eq 12 pg674
    P=care(Abar,(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')./2,Q); %solved using equation A-9 from
page 282 of reference 31 in article
%solving for u equation 24 pg 675
    u=-0.5*(R1^-1)*Bbar1'*P*zbar;
%Pg 674
    ustar=-0.5*(R1^-1)*(Bbar1')*P*zbarstar; %equation 10 pg 674
    fstar=-0.5*(R2^-1)*(Bbar2')*P*zbarstar; %equation 11 pg 674
%% Update R2 Pg 675 equations 17-23
    %equation 19 solved using care function
    q=(1/(2*r2^2))*P*Bbar2*Bbar2'*P;
    b=((Bbar1*(R1^-1)*Bbar1'*P+P*Bbar1*(R1^-1)*Bbar1')+(Bbar2*(R2^-1)*Bbar2'*P+P*Bbar2*(R2^-
1)*Bbar2'));
    dP=care(Abar,b,q);
%Equation 22
    X=Abar-(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')*P;
    Y=-(1/r2^2)*P*Bbar2*Bbar2'*P;
    dp=0.5*(X'^-1)*Y;
%equation 13
    zbarstardot=Abar*zbarstar+Bbar1*ustar-Bbar2*fstar;
%pg 674
    ef=ft-fstar;
    %equations 23 with 18 and 17 substituted in
        %zbarstardot=r2dot*dzbarstar
            %r2dot=-a*ef'*def
            %def=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar)-
((1/(2*r2))*Bbar2'*P*dzbarstar)
        %zbarstardot=-a*ef'*((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar-
((1/(2*r2))*Bbar2'*P*dzbarstar)*dzbarstar;
    E=-a*ef';
    g=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar);
    c=E*g;
    v=E*((1/(2*r2))*Bbar2'*P);
%% solving for dzbarstar
    %fsolve
    %Equation 30 from paper
  %% Nonlinear fsolve - Same results as everything else
   x0=[-0.2;0;0;0.1257;-0.6283;0];
    options = optimoptions('fsolve','Display','iter');
     [dzbarstar,fval] = fsolve(@(dzbarstar)root2d(dzbarstar,c,v,zbarstardot),x0,options);
%% finish updating R2
%equation 18
    def=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar)-((1/(2*r2))*Bbar2'*P*dzbarstar);
%pg 674
    ef=ft-fstar;
%equation 17
    r2dot=-a*ef'*def;
```

```
    r2=r2+r2dot*dt;
    if r2<0.4
        r2=0.4;
    end
    R2=r2*eye(2);
    %% update states
%solving for zbarstar equation 12 pg 674
    zbarstardot=Abar*zbarstar+Bbar1*ustar-Bbar2*fstar;
    zbarstar=zbarstar+zbarstardot*dt;
%solving for zbar equation 7
    zbardot=Abar*zbar+Bbar1*u-Bbar2*ft; %makes zbar a 6x1
    zbar=zbar+zbardot*dt; %new zbar, makes a 6x1
%% Storing values
    zbarstarresult(:,k)=zbarstar;
    dzbarstarresult(:,k)=dzbarstar;
    xdresult(:,k)=xd;
    fstarresult(:,k)=fstar;
    zbarresult(:,k)=zbar;
    efresult(:,k)=ef;
    fstarresult(:,k)=fstar;
    K1result(:,k)=norm(K1);
    zbarstardotresult(:,k)=zbarstardot;
    zbardotresult(:,k)=zbardot;
    K2result(:,k)=norm(K2);
    r2results(:,k)=r2;
    ustarresult(:,k)=ustar;
end
%% plots
figure
hold on
plot(xdresult(1,:),xdresult(2,:))
plot(zbarresult(1,:),zbarresult(2,:))
plot(zbarstarresult(1,:),zbarstarresult(2,:))
hold off

figure
hold on
plot(f)
plot(fstarresult(1,:))
plot(efresult(1,:))
hold off

figure
hold on
plot(K1result)
plot(K2result)
hold off
```

APPENDIX D

FUNCTION CODE

```
function F=root2d(dzbarstar,c,v,zbarstardot)
   %% for Nonlinear fsolve

F=[c*dzbarstar(1)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzbars
tar(1)-zbarstardot(1);

c*dzbarstar(2)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzbarstar(
2)-zbarstardot(2);

c*dzbarstar(3)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzbarstar(
3)-zbarstardot(3);

c*dzbarstar(4)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzbarstar(
4)-zbarstardot(4);

c*dzbarstar(5)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzbarstar(
5)-zbarstardot(5);

c*dzbarstar(6)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzbarstar(
6)-zbarstardot(6);];

   %% for basic fsolve
%
F(1)=c*dzbarstar(1)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzba
rstar(1)-zbarstardot(1);
%
F(2)=c*dzbarstar(2)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzba
rstar(2)-zbarstardot(2);
%
F(3)=c*dzbarstar(3)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzba
rstar(3)-zbarstardot(3);
%
F(4)=c*dzbarstar(4)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzba
rstar(4)-zbarstardot(4);
%
F(5)=c*dzbarstar(5)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzba
rstar(5)-zbarstardot(5);
%
F(6)=c*dzbarstar(6)+v*[dzbarstar(1);dzbarstar(2);dzbarstar(3);dzbarstar(4);dzbarstar(5);dzbarstar(6)]*dzba
rstar(6)-zbarstardot(6);


end
```

APPENDIX E

CODE USING BASIC *fsolve* FUNCTION

```
clear
close all
%% ssetup intial values and constants
Md=[eye(2)]; %2x2 pg677
Cd=[eye(2)]; %2x2 pg677

a=5000; %pg 677
U=[0 pi/5;-pi/5 0]; %2x2 pg677
V=(1/pi)*[eye(2)]; %2x2 pg677

A=[zeros(2) eye(2);zeros(2) -Md^-1*Cd]; %4x4 pg673
B1=[zeros(2);Md^-1]; %4x2 pg673
B2=B1;
 Abar=[A zeros(4,2);zeros(2,4) U]; %6x6 pg674
Bbar1=[B1;zeros(2)]; %6x2 pg674
Bbar2=Bbar1;
 Q1=50*[eye(2)]; %2x2 pg677
Q2=[eye(2)]; %2x2 pg677
R1=0.5*[eye(2)]; %2x2 pg677
%% Orginal Start Value of R2
R2=20*[eye(2)]; %2x2 pg677
%r2=20
%% Change Start value of r2
r2=19;
%% R2dot initial
%r2dot=25;
%%
Q=[Q1 zeros(2) -Q1*V;zeros(2) Q2 zeros(2);-V'*Q1 zeros(2) V'*Q1*V]; %6x6 pg674
 dt=0.01;
t=0:dt:10; %pg 677 %time vector length of sim - start:timestep:end
f=zeros(max(size(t)),1); %pg 677 %linear function-     force=f(t)
f(401:500)=0.5; %pg677 %force vector from 4 to 5 sec
f(501:700)=0.1; %pg677 %force vector from 5 to 7 sec
f(701:1001)=0; %pg677 %force vector from 7 to 10 sec
fy=0;
xd=[-0.2*cos((pi/5)*0),0.2*sin((pi/5)*0)]';
x=xd;
xdot=[(sin((pi*0)/5))/25,(pi*cos((pi*0)/5))/25]'; %differential of xd
%pg 673 finding w=auxiliary state
%xd=V*w; %pg673 desired trajectory
w=linsolve(V,xd);
wdot=U*w; %pg673
%Finding P eq 12 pg674
P=care(Abar,(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')./2,Q);
z=[x;xdot]; %equation given on pg 673
zbar=[z;w]; %zbar intial pg 673
zbarstar=zbar; %6x1
%% for loop
for k=1:length(t) %algorithm 1 pg 675 step of milliseconds
%caluate f from pg677
   f=zeros(max(size(t(k))),1); %pg 677 %linear function-     force=f(t)
   f(1:400)=0;
   f(401:500)=0.5; %pg677 %force vector from 4 to 5 sec
   f(501:700)=0.1; %pg677 %force vector from 5 to 7 sec
   f(701:1001)=0; %pg677 %force vector from 7 to 10 sec
```

```
    fy=0;
    ft=[f(k);fy];
    xd=[-0.2*cos((pi/5)*t(k));0.2*sin((pi/5)*t(k))]; %page 677
%pg 673 finding w=auxiliary state
%xd=V*w; %pg673 desired trajectory equation 6
    w=linsolve(V,xd);
    wdot=U*w; %pg673

%Finding P eq 12 pg674
    P=care(Abar,(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')./2,Q); %solved using equation A-9 from
page 282 of reference 31 in article
%solving for u equation 24 pg 675
    u=-0.5*(R1^-1)*Bbar1'*P*zbar;
%Pg 674
    ustar=-0.5*(R1^-1)*(Bbar1')*P*zbarstar; %equation 10 pg 674
    fstar=-0.5*(R2^-1)*(Bbar2')*P*zbarstar; %equation 11 pg 674
%% Update R2 Pg 675 equations 17-23
%Equation 22
    X=Abar-(Bbar1*(R1^-1)*Bbar1'+Bbar2*(R2^-1)*Bbar2')*P;
    Y=-(1/r2^2)*P*Bbar2*Bbar2'*P;
    dP=0.5*(X'^-1)*Y;
%equation 13
    zbarstardot=Abar*zbarstar+Bbar1*ustar-Bbar2*fstar;
%pg 674
    ef=ft-fstar;
    %equations 23 with 18 and 17 substituted in
        %zbarstardot=r2dot*dzbarstar
            %r2dot=-a*ef'*def %17
            %def=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar)-
((1/(2*r2))*Bbar2'*P*dzbarstar) %18
        %zbarstardot=-a*ef'*((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar-
((1/(2*r2))*Bbar2'*P*dzbarstar)*dzbarstar;
    E=-a*ef';
    g=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar);
    c=E*g;
    v=E*((1/(2*r2))*Bbar2'*P);
%% solving for dzbarstar
    %fsolve
   %% fsolve basic - Same results as Nonlinear
    fun=@(dzbarstar)root2d(dzbarstar,c,v,zbarstardot);
    x0=[-0.2;0;0;0;0;0];
%% Equation 30 from paper
%     dzbarstar=zbarstardot/r2dot;
%% finish updating R2
%equation 18
    def=((1/(2*r2^2))*Bbar2'*P*zbarstar)-((1/(2*r2))*Bbar2'*dP*zbarstar)-((1/(2*r2))*Bbar2'*P*dzbarstar);
%pg 674
    ef=ft-fstar;
%equation 17
    r2dot=-a*ef'*def;
    r2=r2+r2dot*dt;
    if r2<0.4
        r2=0.4;
    end
    R2=r2*eye(2);
```

```
    %% update states
%solving for zbarstar equation 12 pg 674
    zbarstardot=Abar*zbarstar+Bbar1*ustar-Bbar2*fstar;
    zbarstar=zbarstar+zbarstardot*dt;
%solving for zbar equation 7
    zbardot=Abar*zbar+Bbar1*u-Bbar2*ft; %makes zbar a 6x1
    zbar=zbar+zbardot*dt; %new zbar, makes a 6x1
%% Storing values
    zbarstarresult(:,k)=zbarstar;
    dzbarstarresult(:,k)=dzbarstar;
    xdresult(:,k)=xd;
    fstarresult(:,k)=fstar;
    zbarresult(:,k)=zbar;
    efresult(:,k)=ef;
    fstarresult(:,k)=fstar;
    K1result(:,k)=norm(K1);
    zbarstardotresult(:,k)=zbarstardot;
    zbardotresult(:,k)=zbardot;
    K2result(:,k)=norm(K2);
    r2results(:,k)=r2;
    ustarresult(:,k)=ustar;
end

%% plots
figure
hold on
plot(xdresult(1,:),xdresult(2,:))
plot(zbarresult(1,:),zbarresult(2,:))
plot(zbarstarresult(1,:),zbarstarresult(2,:))
hold off

figure
hold on
plot(f)
plot(fstarresult(1,:))
plot(efresult(1,:))
hold off

figure
hold on
plot(K1result)
plot(K2result)
hold off
```