TRAJECTORY GENERATION TECHNIQUES FOR CABLE DRIVEN PARALLEL ROBOTS

KABLOYLA ÇALIŞAN PARALEL ROBOTLAR İÇİN ROTA OLUŞTURMA YÖNTEMLERİ

ABDUL MOIZ AWAN

PROF. DR. S. ÇAĞLAR BAŞLAMIŞLI

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Mechanical Engineering.

2024

This research is lovingly dedicated to my parents Asif Hayat and Sadaf Noran. They have always believed in me and empowered me to pursue my dreams of becoming an engineer. Without their love and support I would not be here today. I love you Ammi and Abbu.

ABSTRACT

TRAJECTORY GENERATION TECHNIQUES FOR CABLE DRIVEN PARALLEL ROBOTS

Abdul Moiz AWAN

Master's of Philosophy, Department of Mechanical Engineering

Supervisor: Prof. Dr. S. Çağlar BAŞLAMIŞLI

June 2024, 141 pages

The aim of this thesis is to implement CNC-based trajectory generation techniques with Cable Driven Parallel Robots (CDPRs). CDPRs are a special type of parallel robots that use motors and cables to manipulate an end-effector in space. Cable robots are becoming popular due to several advantages they have over traditional manipulators such as their large workspace and lightweight actuators. For the cable robots to perform properly there are a number of parameters that need to be studied. These include maintaining positive cable tension, cable elasticity, trajectory generation, controller design, and structural optimization.

The scope of this thesis is limited to planar cable robots, where a four-cable robot has been chosen for the analysis. The reason for choosing a four-cable robot is that the fourth actuator offers a redundancy which allows control of the end effector with 3 DOFs while also being able to maintain tension. Maintain positive tension is very important as unlike traditional manipulators, CDPRs are not able to push against the end-effector, only pull. A positive tension algorithm ensures that none of the cables ever go slack. Cable elasticity

and its incorporation into the mathematical model of the cable robot is another important part of the developed model.

Trajectory generation is one of the most important topics in this research as it uses CNC-based trajectory generation algorithms to generate trajectories for a cable robot. These algorithms range from simple linear or circular interpolation to complicated 5^{th} order splines which ensure continuity up to at least the second derivative. A combination of splines, and simple segments are used to generate standard and custom shapes and non-uniform data supplied by the user is also simulated by connecting splines with the data. These trajectories are tested in different scenarios and for different conditions such as changing speed or some non-zero angle reference for the end-effector.

The controller design is another important aspect. It is a cascade controller which is very common for controlling motors. The controller gains are determined based on the settling time and the maximum overshoot. Lastly the structural optimization of the cable robot for dexterity, stiffness, and workspace is studied where the optimized robot was found to provide better results. A GUI is developed which incorporates all the different codes to make an easy-to-use tool for designing and simulating a cable robot.

Keywords: Cable Driven Parallel Robot, Cascade Control, Trajectory Generation, Optimization, Cable Elasticity, GUI

ÖZET

KABLOYLA ÇALIŞAN PARALEL ROBOTLAR İÇİN ROTA OLUŞTURMA YÖNTEMLERİ

Abdul Moiz AWAN

Yüksek Lisans, Makina Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. S. Çağlar BAŞLAMIŞLI

Haziran 2024, 141 sayfa

Bu tezin amacı, Kabloyla Çalışan Paralel Robotlar (KÇPR'ler) ile CNC tabanlı rota oluşturma tekniklerini uygulamaktır. KÇPR'ler, motorlar ve kablolar kullanarak bir uç efektörünü uzayda manipüle eden özel bir paralel robot türüdür. Kablo robotları, geleneksel manipülatörlere göre geniş çalışma alanı ve hafif aktüatörler gibi çeşitli avantajları nedeniyle popüler hale gelmektedir. Kablo robotlarının düzgün çalışması için incelenmesi gereken birçok parametre bulunmaktadır. Bunlar arasında pozitif kablo geriliminin korunması, kablo esnekliği, yörünge oluşturma, kontrolör tasarımı ve yapısal optimizasyon yer almaktadır.

Bu tezin kapsamı, dört kablolu bir robotun analiz için seçildiği düzlemsel kablo robotları ile sınırlıdır. Dört kablolu bir robotun seçilmesinin nedeni, dördüncü aktüatörün bir yedeklilik sunması ve bu sayede 3 serbestlik derecesi (DOF) ile uç efektörünün kontrolünü sağlarken aynı zamanda gerilimi koruyabilmesidir. Pozitif gerilimin korunması çok önemlidir çünkü geleneksel manipülatörlerin aksine, KPR'ler uç efektörüne karşı itme yapamazlar, yalnızca çekebilirler. Pozitif gerilim algoritması, kablolardan hiçbirinin gevşememesini sağlar. Kablo esnekliği ve bu esnekliğin kablo

robotunun matematiksel modeline dahil edilmesi, geliştirilen modelin önemli bir parçasıdır.

Bu araştırmadaki en önemli konulardan biri yörünge oluşturmadır çünkü CNC tabanlı yörünge oluşturma algoritmalarını kullanarak kablo robotu için yörüngeler oluşturur. Bu algoritmalar, basit doğrusal veya dairesel enterpolasyondan, en az ikinci türevine kadar sürekliliği sağlayan karmaşık 5. dereceden splinelere kadar çeşitlilik gösterir. Standart ve özel şekiller oluşturmak için splineler ve basit segmentlerin bir kombinasyonu kullanılır ve kullanıcı tarafından sağlanan uniform olmayan veriler de splinelerle birleştirilerek simüle edilir. Bu yörüngeler, farklı hız değiştirme senaryoları veya uç efektör için sıfır olmayan bir açı referansı gibi farklı koşullar altında test edilir.

Kontrolör tasarımı da bir diğer önemli konudur. Bu tasarım, motorları kontrol etmek için çok yaygın olan bir kaskad kontrolördür. Kontrolör kazançları, yerleşme süresi ve maksimum aşım temel alınarak belirlenir. Son olarak, kablo robotunun hareket kabiliyeti, rijitlik ve çalışma alanı için yapısal optimizasyonu incelenir ve optimize edilmiş robotun daha iyi sonuçlar sağladığı görülür. Tüm farklı kodları bir araya getiren ve kablo robotu tasarlamak ve simüle etmek için kullanımı kolay bir araç sunan bir grafik kullanıcı arayüzü (GUI) geliştirilmiştir.

Anahtar Kelimeler: Kabloyla Çalişan Paralel Robot, Kademeli Kontrol, Rota Oluşturma, Optimizasyon, Kablo Esnekliği, GKA

ACKNOWLEDGEMENT

I would like to thank Prof. Dr. S. Çağlar Başlamişli who has always gone above and beyond to support me in this research. I have not only benefited from his academic experience, but also the advice and insight he has provided for different problems. He has always taken a personal interest in my academic progress and well-being. I would like to thank him for always motivating me and providing me with the opportunity to work under his supervision.

I would also like to thank my family, who have enabled and encouraged me to pursue higher education in Turkey. While it is difficult being away from them, they have always supported me and ensured that all my needs are met.

I would also thank my BS Thesis supervisor Dr. Sana Waheed who helped me apply for this MS program and was my role model as an academic and a researcher.

Lastly, I would like to thank my friends who have always been there for me through thick and thin. While we are far from each other they have always been my support system and helped me deal with any challenge I have faced. I would like to thank Ammar Ijaz particularly who was a huge support during the final days of this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

Symbols

α_i	Cable Angles
θ	End-Effector orientation angle
L_i	Cable Lengths
L_{0i}	Initial Cable Lengths
A_i	Motor Positions
B_i	End Effector Cable Connection Points
R_A/R_B	Distance of A_i/B_i from center of global frame
J	Motor Inertia
C	Motor damping
τ	Motor torque
T	Cable Tensions
k_i	Cable stiffness

Abbreviations

CDPR/M	Cable Driven Parallel Robot/Manipulator
DOF	Degree of Freedom
COG/CG	Center of Gravity
CT	Cornering Tolerance
SN	Stiffness Number
GCI	Global Conditioning Index
WFW	Wrench Feasible Workspace
SFW	Stiffness Feasible Workspace
WVI	Workspace Volume Index

1. INTRODUCTION

1.1. Overview

Parallel manipulators are devices which use a set of linkages in parallel to an end-effector which can have translational or rotational motion, defined as the Degrees-of-Freedom or DOFs of the system. Cable-Driven-Parallel-Robots or CDPRs are specialized type of parallel robotic manipulators which use cables that are constantly in tension to manipulate an object or end-effector. The Cable Robot has a number of cables and motors which are used to manipulate the end-effector and follow and maintain a desired trajectory and pose. Advantages of using Cable Robots include lower mass and cost compared to traditional robots. Cable Robots can also have very large workspaces, but a large area is needed to accommodate such setups. Disadvantages include the actuator redundancy needed to operate cable robots as the cables need to be kept constantly in tension.

A critical aspect of the cable robot design is trajectory generation. The method of trajectory generation can affect the accuracy of the CDPR. Another important factor is the optimization of the cable robot. The location of the attachment points for the cables on the end-effector, and the locations of the motors can significantly impact the performance of a cable robot. This study focuses on accurate trajectory generation, controller design, and cable robot optimization to create a more efficient design for a cable robot.

1.2. Thesis Objectives and Outcomes

The primary objective of this thesis was to model and simulate a 3-DOF Planar Cable Robot and use different techniques to develop different trajectories for the robot to follow. Particularly, trajectory generation techniques traditionally used for CNC applications were implemented in the context of cable robots to achieve highly accurate results and improving the cable robot performance.

Additional objectives include the development of a Cascade Controller for simulating the Cable Robot and the design optimization using different cost functions. These include dexterity, stiffness, and workspace optimization.

A comprehensive analysis is carried out testing the cable robot performance before and after optimization for different trajectories. All of this procedure is packaged into an easy-to-use GUI application which can be used to completely design and simulate a cable robot device.

1.3. Research Contribution

In light of the reviewed literature, there are no other studies implementing CNC trajectory generation techniques for Cable Robots. The GUI developed as a part of this thesis is also a unique tool which offers functionality not comparable with other existing tools.

The research contributions of this thesis can be summed up as given below:

- Implementation of different trajectory generation techniques, particularly from CNC trajectory generation, for a CDPR.
- Comparison of different trajectory generation techniques and results before and after optimization.

1.4. Thesis Organization

The structure of the thesis is organized as follows: Chapter 2 provides a background and literature review about different types of Cable Robots. Moreover, literature related to cable robot modelling, trajectory generation, and optimization is also discussed. Chapter 3 provides the methodology and detailed information about the cable robot modelling including cable modelling. The different methods and algorithms used for trajectory generation are discussed in Chapter 4. Chapter 5 presents the controller design used to run the cable robot to follow the designed trajectories and Chapter 6 provides information about the Cable Robot optimization. Chapter 7 provides the results and discussion for

different simulations and finally, Chapter 9 provides the conclusion and outlines the future work to be done. Figures of the GUI are provided in the Appendix.

2. BACKGROUND AND LITERATURE REVIEW

In this Chapter, a detailed literature review and background of cable driven robots is presented. Firstly, cable driven parallel robots are introduced along with their potential applications. Then, the concepts behind cable robot modelling, trajectory generation, controller design and cable robot optimization are presented from different studies.

2.1. Cable Driven Parallel Robots Background

Cable driven parallel robots (CDPRs) are a special type of parallel robots that use several cables to manipulate the end-effector or the payload. These flexible cables replace the traditional rigid links used in serial manipulators. By using cables, CDPRs have several advantages over conventional manipulators which include much larger workspaces, lower costs of manufacturing, as well as significantly increased payload-to-weight ratios[1][2][3,4]. CDPRs have gained the attention of researchers due to these properties and several theoretical and practical applications have been discovered for these robots including robotic cranes, modular solar collectors, sports cameras, and many more.

The NIST Robocrane is widely accepted as one of the first implementations of a Cable Driven Parallel Robot. It was developed in 1992 at the National Institute of Standards and Technology in Maryland, USA. It consists of a triangular platform suspended by 6 cables. Each vertex of the platform is connected to two cables, and the cables are actuated by 6 winches. This allows the platform to be positioned at a specific pose and orientation. The robot is kinematically constrained when all the cables are in tension and in this case a fixed relation is present between the length of the cables and the position and orientation of the system. The Robocrane was capable of having both manual and closed loop control.

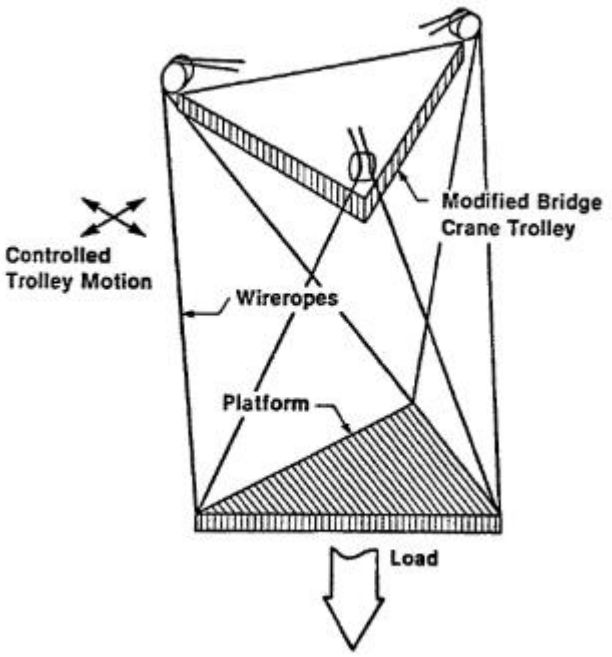

Figure 2.1. NIST RoboCrane [5]

During manual control, a Stewart platform with potentiometers was used for a master-slave rate control, while the closed loop control a computer is used to calculate the desired cable lengths and implement position, velocity, and force control.[5]

Tang performed a detailed study of different implementations of CDPRs. Cable robots are divided into two categories, suspended robots and redundantly constrained robots. For suspended robots, gravity is necessary in maintaining positive cable tension. While for redundantly constrained robots, the actuator forces are enough to maintain cable tension. The Robocrane is an example of a suspended CDPR. Other challenges associated with the design of cable robots include cable models, static and dynamic workspace, and tension distribution. Based on the literature, the two most popular types of control are cable length based (joint space) and end-effector position based (task space) control. A simple PD controller-based feedback control using cable lengths is a common control scheme, while other researchers have also used Computed Torque controllers in the task space, but these rely on the availability of accurate sensors. Cable robots are most commonly used in large telescopes, medical rehabilitation robotics, sports cameras, and large simulators.[3] These CDPRs are used in a variety of applications ranging from

mobile cranes, rehabilitative equipment, video cameras etc.[6] CDPRs have also been extensively used in rehabilitative applications [7–9]. The controller for a CDPR is generally designed in the task space of the robot as the dynamic equations of the model are also determined in the task space. Therefore, the design of the controller becomes less challenging. However, control in the task space can require many complex sensors, so the joint space might be preferable for controller design.[10]

2.2. Cable Robot Modelling

The use of cables to manipulate the end effector introduces a unique challenge as the cables can only be in tension and cannot exert a compressive force. The slackness of the cables is also not desirable. This introduces a new and unique control problem. CDPRs are divided into under-constrained, fully constrained, and redundantly constrained categories. In under-constrained systems the tension of the cables is usually provided by gravity. In fully constrained systems, an added extra actuator is used to provide tension in the cables using internal forces, and in redundantly constrained systems, even more actuators are added as compared to a fully constrained system, and the tension can be provided either by suspension or with internal forces.[11]

Qian et al. conducted a review of CDPRs covering the history of CDPR development. Due to the development of control theory and design improvements, CDPRs have greatly improved in terms of their dynamic and kinematic performances and have seen increased use in practical applications. However, they are still rarely used as compared to traditional serial manipulators. The review paper analyzes several Cable Driven Parallel robots which are controlled using different algorithms. It highlights the need for the integrated design of different configurations of CDPRs, the development of higher performance control algorithms, and – lastly – to improve the stiffness and load bearing capacity using composite materials [6]. Electronic motors are the most common methods of actuation. The cables and pulleys used for the system can be of different materials. A variety of controllers have also been used to control CDPRs such as hybrid, PD, model predictive, and adaptive control highlighted in [11–14], however the most common controller is a PID controller. The CDPR can be seen as a combination of the actuator, the pulleys, the

cables, and the end effectors. The position of the pulleys and the actuators limits the final achievable workspace.

The open-source WireX [15] has the capability to generate geometry and perform kinematic and static analysis of CDPRs. It is also able to analyze the workspace of the designed CDPRs as well as other useful features. However, it has limited design and analysis capabilities, and it is not able to design a controller for the analyzed system.

A CDPR has the additional challenge compared to a regular parallel manipulator with rigid links, that it is only able to pull on the end-effector and not push. This means, the cables need to be in a constant state of tension. Oh et al. provide a detailed approach to determining the statics and dynamics of a generalized cable robot with 'n' number of cables as well as determining the feasible regions for the cable arrangements. To maintain positive cable tension, a cable robot needs one more cable than the desired DOFs. So, for a planar robot if planar movement and rotation is desired, a four-cable robot would be suitable. This paper particularly presented an approach to control cable robots with redundant cables while also maintaining positive cable tension [1].

Based on the resources studied above, the design problem of the cable robot can be divided into a few different sections. CDPRs can have a number of different configurations and orientation, any number of cables, controllers, and these robots each have a different performance objective.

Zarebidoki [16] performed a thorough review of cable robots considering these exact parameters. The CDPRs are classified into Incompletely Restrained, Completely Restrained, and Redundantly Restrained mechanisms. Based on the literature, redundantly constrained devices are not very common, while completely restrained devices are the most common. Incompletely restrained devices are also used but they are usually feasible for suspended devices. Planar and Spatial devices are also equally common but planar devices are almost always completely restrained. While modelling the cable robots, another important aspect to consider are the cables. The simplest way of

modelling the cables are non-elastic massless cables. Other papers model cables with elasticity, and cables with mass, while a small subset of the literature deals with cables with both mass and elasticity. Cable elasticity is important as the extension of the cables can have important implications regarding trajectory tracking and control. Cable mass can cause cable sag, especially in larger devices. Cable sag is very complicated to model and can often be neglected for small to intermediate sized cable robots, while cable elasticity is more relevant to most design problems. The paper also shows the workspace of a CDPR as an important factor and compares literature optimizing the workspace based on the positioning of the cable configurations. The wrench-closure workspace considers the area where positive tension can be maintained while the wrench-feasible workspace considers both upper and lower bounds for the cable tension. Lastly, the trajectory planning and control of the cable robots are an important factor. For robots with inelastic or elastic cables, different control approaches can be developed but PID and fuzzy control were common techniques.

Figure 2.2 below shows the profile for a cable under sag.

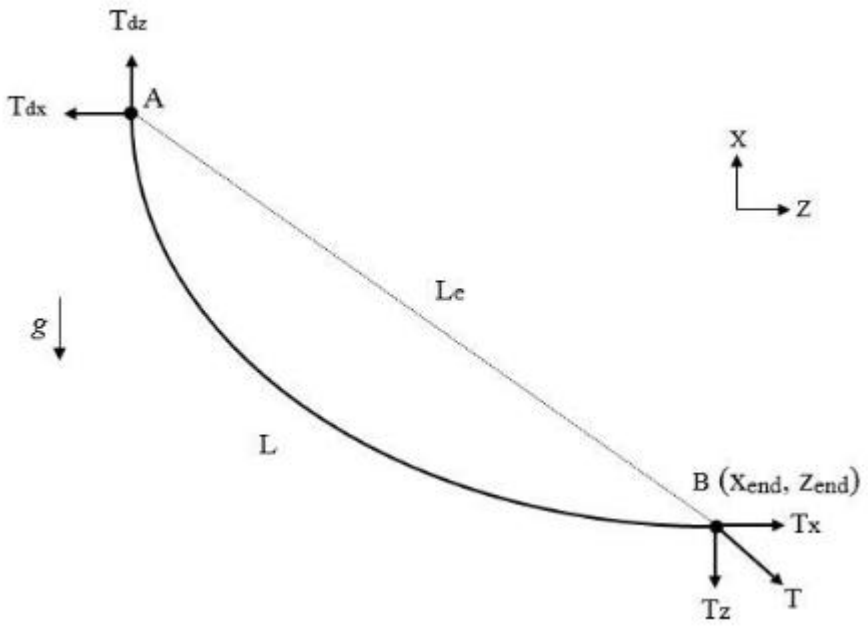

Figure 2.2. Sagging Cable Profile between two points [17]

The cable catenary equations are well known, and for the case where the cable is inextensible the equations are given below [17],

$$x_{\text{iend}} = \frac{|T_{xi}|}{\rho_L g}\, g \left[\sinh^{-1}\left(\frac{T_{zi}}{T_{xi}}\right) - \sinh^{-1}\left(\frac{T_{zi} - \rho_L g L_L}{T_{xi}}\right)\right]$$

$$z_{\text{iend}} = \frac{1}{\rho_L g}\left[\sqrt{T_{xi}^2 + T_{zi}^2} - \sqrt{T_{xi}^2 + (T_{zi} - \rho_L g L_i)^2}\right] \tag{2.1}$$

$$T_i = \sqrt{T_{xi}^2 + T_{zi}^2}$$

Where x_{iend} and z_{iend} are the coordinates in the global frame of the end point of the cable, T_{xi} and T_{zi} are the components in the x and z direction of the cable tension, ρ_L is the density of the cable, g is gravitational acceleration in the negative z direction, L_i is the cable length, and T_i is the total magnitude of cable tension. To obtain the cable lengths, and ultimately the desired cable angles for a specified position, these equations need to be solved iteratively using a function like fsolve() at each time step as well as some kind of optimization algorithm which finds the solution with the cable tensions kept to a minimum. With the addition of cable elasticity, cable sag becomes a very challenging and computationally expensive aspect of the cable robot which more most cases except really large cable robots can be essentially ignored.

2.2. Trajectory Generation

Trajectory generation is another important part of the cable robot design process. Hwang [18] used a trajectory generation algorithm to suppress oscillations in under-constrained cable robots such as those suspended due to gravity. The unwanted oscillations are prevented according to both experimental and simulation data.

Kevac [19] also developed an algorithm to generate the trajectory of a cable suspended parallel robot which is cable of tracking different objects. Jiang [20] developed a trajectory generation method where considering the mathematical model for the kinematic and dynamic formulations of the robot. It is assumed that there are positive constant ratios between the cable tensions and cable lengths. Assuming positive constant ratios between the cable tensions and lengths, the dynamic equations can be converted into linear differential equations with constant coefficients for positioning. Concurrently, the orientation equation becomes a pendulum-like differential equation. These equations can then be solved to determine the trajectory.

In the literature, there is a lack of publications where CNC-based trajectory generation techniques have been implemented with cable robots. These trajectory generation techniques use 5^{th} order splines which guarantee continuity of the trajectory at least up to the second derivative which can result in smoother trajectories for different custom and non-custom shapes.

2.2. Control System

The controller chosen for implementation with the cable robot model is a Cascade Controller. Cascade controllers are popular for controlling electronic motors and consist of two cascading control loops. The inner-loop is a speed control loop, whereas the outer-loop is a position control loop. Cascade control can be particularly beneficial in cable robot applications. When controlling cable robots, it is often difficult to obtain the exact position of the end-effector so joint space-based control strategies are preferred over task space control strategies.

Khosravi [21] developed a cascade controller for a 3-DOF planar-cable-robot which is similar to our application. Experimental results showed that the cascade controller was able to track the desired trajectory effectively with very minimal error. Similarly, [22] developed an adaptive cascade controller for the KNTU CDPRM which is a 6-DOF cable robot. Experimental results again verified that the tracking errors are quite small. The adaptive algorithm is able to change the gains of the controller as needed. Khalilpour [23] developed a cable robot and used sliding mode cascaded controller for trajectory tracking. During experimentation, the results for both cases with and without the inner loop are given and the addition of the inner loop, i.e., using the cascade control structure significantly increased the tracking performance.

2.2. Cable Robot Optimization

The scope of this thesis is limited to Planar CDPRs which are completely restrained, that is the number of cables used in the robot is the number of DOFs plus one. Table 2.1 below

shows different sources and the associated cost functions being used for cable robot optimization.

Table 2.1. Cost Functions for Cable Robot Optimization

No.	Paper Name	Objective Function
1	"On the Design of Cable-Suspended Planar Parallel Robots" [24]	Maximize workspace, Maximize Global Condition Index
2	"Workspace optimization for a planar cable-suspended direct-driven robot" [25]	Maximize workspace efficiency (based on shape of workspace rather than size)
3	"DESIGN AND OPTIMIZATION OF A PLANAR CABLE ROBOT" [26]	Minimize sum of maximum tensions during performance of task
4	"Optimization based Trajectory Planning of Mobile Cable-Driven Parallel Robots" [27]	Minimize position and velocity vectors to find most efficient path
5	"Design and optimization of three-degree-of-freedom planar adaptive cable-driven parallel robots using the cable wrapping phenomenon" [28]	Maximize workspace and Dexterity
6	"Optimizing Stiffness and Dexterity of Planar Adaptive Cable-Driven Parallel Robots" [29]	Maximum dexterity with target stiffness, OR, Maximum stiffness with target dexterity
7	"Orientation Workspace and Stiffness Optimization of Cable-Driven Parallel Manipulators with Base Mobility" [30]	Maximize stiffness, Tension Factor, Minimize error between desired and actual joint positions

8	"Multi-Objective Optimal Design of a Cable-Driven Parallel Robot Based on an Adaptive Adjustment Inertia Weight Particle Swarm Optimization Algorithm" [31]	Maximize Workspace Index, and Dexterity Index
9	"Simulation and optimization of automated masonry construction using cable robots" [32]	Minimize spline time, Maximize stiffness, Minimize energy consumption
10	"Cable Attachment Optimization for Reconfigurable Cable-Driven Parallel Robots Based on Various Workspace Conditions" [33]	Maximize Tension Factor, or Minimize Cable Force Sum
11	"Kinematic Analysis and Design Optimization of a Cable-Driven Universal Joint Module" [34]	Maximize tension-closure workspace volume

Analyzing the different sources, Dexterity, Stiffness, and Workspace optimization stand out as a clear trend, therefore these cost functions have been chosen for the cable robot optimization procedure.

3. MODELLING METHODOLOGY

This chapter will describe the modelling problem of the cable driven parallel robot. This will include the kinematics, dynamic equations of the cable robot, positive tension algorithm, and addition of cable elasticity to the model.

3.1. Problem Formulation

The scope of this thesis is limited to a planar 4-Cable robot. The robot has 3-DOFs namely, translatory motion in the X-axis, translatory motion in the Y-axis, and rotation about the Z-axis (rotation in the plane of motion). As the robot has 4 cables and 3-DOFs, it is redundantly actuated and has the ability to maintain positive tension in the cables.

If a cable robot is not redundantly actuated it is not able to maintain positive cable tension without an external force (such as gravity). 4 cables allow the planar robot to have freedom of motion and rotation while maintaining positive cable tension. If more than 4 cables are used, it does not affect the DOFs of the cable robot however the computational complexity of the problem increases.

To model the cable robot, the first step is to determine the forward and inverse kinematics of the system as well as the dynamic formulation. Using the obtained equations, the model of the Cable Robot can be simulated using MATLAB/Simulink. The basic model of the cable robot needs to be a general formulation where the user can specify different cable connection points, motor positions, mass, inertia, workspace etc. according to the design specification. After the basic model is developed, cable elasticity has to be incorporated followed by the positive cable tension algorithm which ensures that none of the cables end up sagging which is very important. The mass and subsequent sagging of cables has not been considered within the scope of this thesis as it is generally negligible apart from some very large cable robots.

Figure 3.1 below shows the general configuration for a planar CDPR with four cables. Here A_i are the motor positions, B_i are the cable connection points measured from the

center of the end-effector, L_i are the lengths of the cables, and α_i are the angles made by the cables in the global frame. The global coordinate system used for the modelling process is located at the central point of the area defined by the motor positions. The motor positions and end-effector cable connection points are usually symmetric.

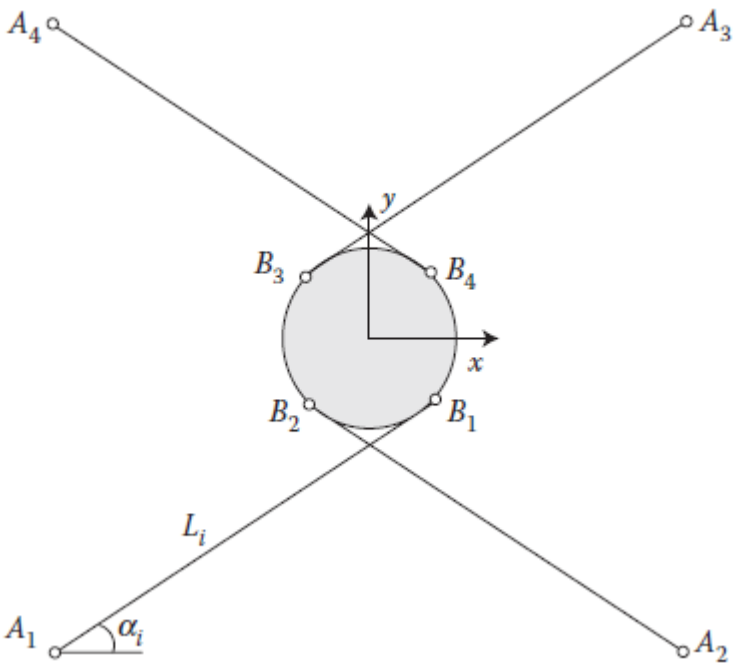

Figure 3.1. A generic 4-cable planar robot [10]

3.2. Cable Robot Modelling – Inverse Kinematics

For cable robots, the inverse kinematics problem is usually more important and is easier to solve as compared to the forward kinematics problem. For the inverse kinematics, the position and pose of the end-effector is given while the required motor positions or cable lengths must be calculated. Figure 3.2 below shows the kinematic configuration for the manipulator.

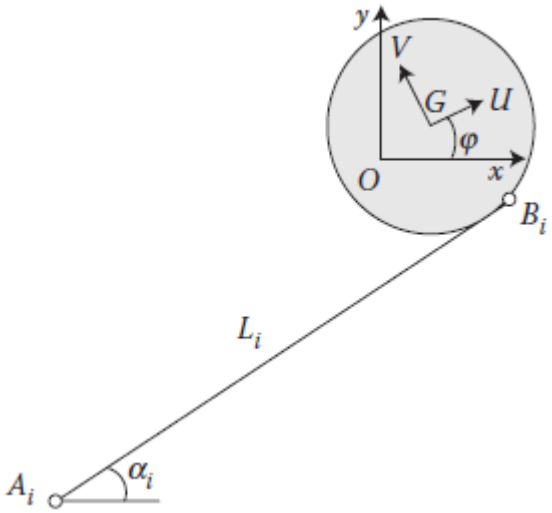

Figure 3.2. CDPR showing the cable and end-effector angle [10]

As the motor positions A_i and the end-effect cable connection points B_i are usually symmetric, it can be said that these points are on circles with radii R_A and R_B respectively. We can also define θ_{Ai} and θ_{Bi} which are the angles of the lines connecting the points A_i and B_i to the center of the global coordinate system assuming the end-effector is placed at that point. Then, if the end-effector rotates by an angle ϕ, the new angles ϕ_i of points B_i are given as,

$$\phi_i = \phi + \theta_{B_i} \tag{3.1}$$

Based on these, the points A_i and B_i can be given as,

$$A_i = \left[R_A \cos\left(\theta_{A_i}\right), R_A \sin\left(\theta_{A_i}\right)\right]^T \tag{3.2}$$

The end-effector position and orientation is given as,

$$\mathcal{X} = [x_G, y_G, \phi]^T \tag{3.3}$$

And the joint variables or the cable lengths are given as,

$$L = [L_1, L_2, L_3, L_4]^T \tag{3.4}$$

Then for each limb where i = 1, 2, 3, 4, based on the geometry, the loop-closure equation is given as,

$$\overrightarrow{A_\iota G} = \overrightarrow{A_\iota B_\iota} - \overrightarrow{G B_\iota} \tag{3.5}$$

Rewriting the loop closure equation in terms of its components,

$$x_G - x_{Ai} = L_i \cos(\alpha_i) - R_B \cos(\phi_i)$$
$$y_G - y_{Ai} = L_i \sin(\alpha_i) - R_B \sin(\phi_i) \quad (3.6)$$

Here, x_{Ai} and y_{Ai} are the x and y components of the position vectors of points A_i and B_i respectively. To solve the above equation, it is necessary to eliminate α_i and solve for L_i. Rearranging we obtain,

$$L_i \cos(\alpha_i) = x_G - x_{Ai} + R_B \cos(\phi_i)$$
$$L_i \sin(\alpha_i) = y_G - y_{Ai} + R_B \sin(\phi_i) \quad (3.7)$$

By adding the squares of both equations, the expression for calculating the cable lengths is given as,

$$L_i = \left[\left(x_G - x_{Ai} + R_B \cos(\phi_i)\right)^2 + \left(y_G - y_{Ai} + R_B \sin(\phi_i)\right)^2\right]^{1/2} \quad (3.8)$$

And the limb angles α_i are obtained as,

$$\alpha_i = \text{Atan } 2\left[\left(y_G - y_{Ai} + R_B \sin(\phi_i)\right), \left(x_G - x_{Ai} + R_B \cos(\phi_i)\right)\right] \quad (3.9)$$

Therefore, the inverse kinematics problem has a unique solution for each manipulator location. This approach is used to calculate the required cable length and the equivalent motor angles as a function of the generated trajectories so motor reference trajectories can be generated in the task space for the cascade controller described in section 5.

3.3. Cable Robot Modelling – Forward Kinematics

In the forward kinematics problem, the cable lengths (L_i) are known and the manipulator location needs to be determined. First, we define two intermediate variables given below,

$$\begin{cases} x_i = -x_{Ai} + R_B \cos(\phi_i) \\ y_i = -y_{Ai} + R_B \sin(\phi_i) \end{cases} \quad (3.10)$$

Then taking the square of Equation 3.11, it becomes,

$$L_i^2 = (x_G + x_i)^2 + (y_G + y_i)^2 \quad (3.11)$$

Solving for x_G and y_G,

$$x_G^2 + y_G^2 + r_i x_G + s_i y_G + u_i = 0 \quad (3.12)$$

Where,

$$r_i = 2x_i,\ s_i = 2y_i,\ u_i = x_i^2 + y_i^2 - L_i^2 \tag{3.13}$$

Equation 3.13 provides four quadratic relationships for i = 1, 2, 3, 4. Subtracting each equation from the other yields linear equations in terms of x_G and y_G.

$$A \cdot \begin{bmatrix} x_G \\ y_G \end{bmatrix} = b \tag{3.14}$$

Where,

$$A = \begin{bmatrix} r_1 - r_2 & s_1 - s_2 \\ r_2 - r_3 & s_2 - s_3 \\ r_3 - r_4 & s_3 - s_4 \\ r_4 - r_1 & s_4 - s_1 \end{bmatrix},\ b = \begin{bmatrix} u_2 - u_1 \\ u_3 - u_2 \\ u_4 - u_3 \\ u_1 - u_4 \end{bmatrix} \tag{3.15}$$

All the elements of the A and b matrices are functions of ϕ. Although only two equations from above are sufficient to evaluate x_G and y_G in terms of ϕ, however using all four equations helps to achieve tractable solutions even at singular configurations. This equation can be solved using the pseudoinverse of A,

$$\begin{bmatrix} x_G \\ y_G \end{bmatrix} = A^{\dagger} \cdot b \tag{3.16}$$

Where,

$$A^{\dagger} = (A^T A)^{-1} A^T \tag{3.17}$$

Equation 3.16 gives the solution with least-squares error for x_G and y_G. To obtain ϕ, Equations 3.12 and 3.16 are combined, resulting in a single function where the unknown variable is ϕ:

$$f_i(\phi) = x_G^2 + y_G^2 + r_i x_G + s_i y_G + u_i \tag{3.18}$$

Then consider,

$$f(\phi) = \sum_{i=1}^{4} f_i(\phi) \tag{3.19}$$

and use numerical methods that use iterative search routine to obtain the actual solution for the function f(ϕ) = 0. It appears that any function from Equation 3.18 could be used to obtain the solution, however, all the four equations are summed up and used together, it allows the equation to have a tractable solution even at singular configurations. Figure 3.3 below shows the flowchart explaining the procedure for the forward kinematics problem.

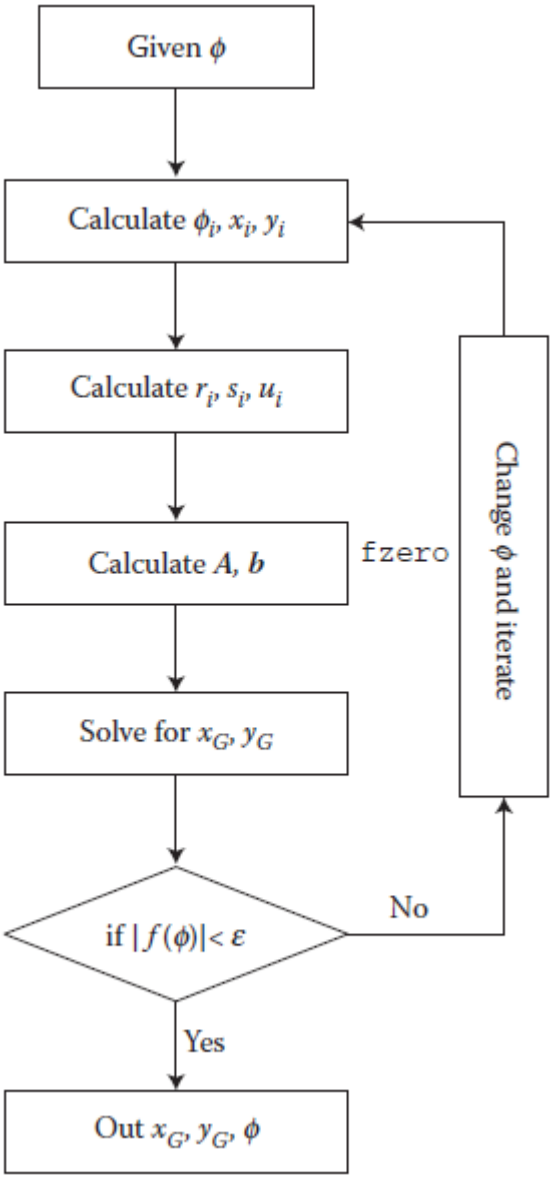

Figure 3.3. Iterative Forward Kinematics Procedure for a CDPR [10]

3.4. Cable Robot Modelling – Dynamics

Figure 3.4 below shows a Free-Body-Diagram for the cable robot end-effector.

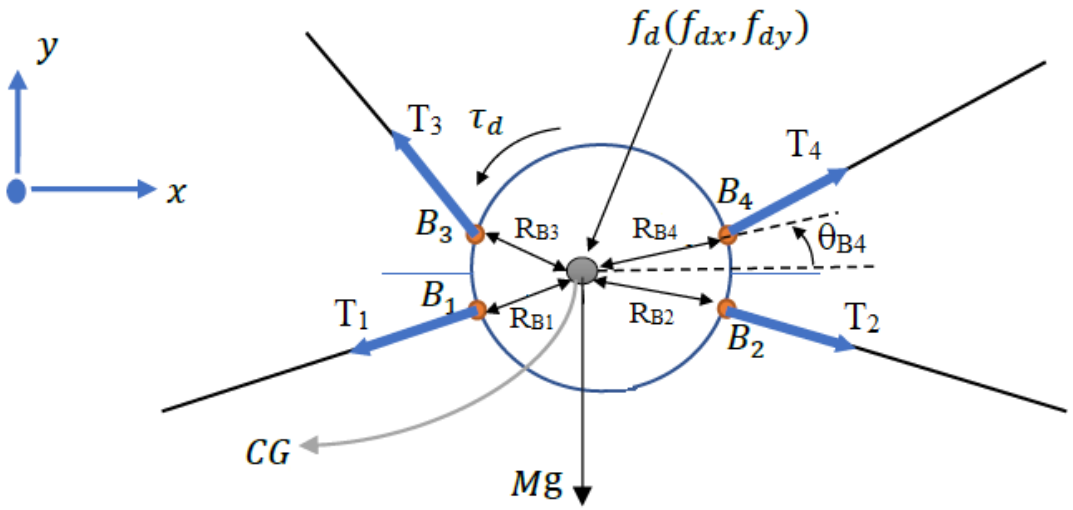

Figure 3.4. Free-Body-Diagram for an end-effector attached with four cables

Here, f_d is the vector of external forces, M is end-effector mass, g is the vertical acceleration is gravitational effects are considered, T_i are the cable tensions, B_i are the cable attachment points, R_{Bi} are the distances of B_i from the COG of the end-effector, τ_d is external torque, and θ_{Bi} are the angles in the global coordinate system of the points B_i. CG is the position of the end-effector in the global coordinate system given as,

$$CG = [x_G, y_G\ ] \tag{3.20}$$

Then the positions of B_i are given as,

$$B_i = [x_G + R_{Bi} * \cos(\theta_{Bi} + \phi) \quad y_G + R_{Bi} * \sin(\theta_{Bi} + \phi)] \tag{3.21}$$

Then the unit vector from the position of the center of the end-effector CG, to the connection points B_i is given as,

$$n_{R_{Bi}} = \begin{bmatrix} (x_{Bi} - x_G)/\sqrt{(x_{Bi} - x_G)^2 + (y_{Bi} - y_G)^2} \\ (y_{Bi} - y_G)/\sqrt{(x_{Bi} - x_G)^2 + (y_{Bi} - y_G)^2} \end{bmatrix} \tag{3.22}$$

Similarly, the unit vector between A_i and B_i is given as,

$$S_i = \begin{bmatrix} (x_{Ai} - x_{Bi})/\sqrt{(x_{Ai} - x_{Bi})^2 + (y_{Ai} - y_{Bi})^2} \\ (y_{Ai} - y_{Bi})/\sqrt{(x_{Ai} - x_{Bi})^2 + (y_{Ai} - y_{Bi})^2} \end{bmatrix} \tag{3.23}$$

Then the dynamics equations for the 3-DOF planar robot end-effector are given as,

$$M.\ddot{x}_G = f_{dx} + T_1S_{1,x} + T_2S_{2,x} + T_3S_{3,x} + T_4S_{4,x} \tag{3.24}$$

$$M.\ddot{y}_G = f_{dy} - Mg + T_1S_{1,y} + T_2S_{2,y} + T_3S_{3,y} + T_4S_{4,y} \tag{3.25}$$

$$I.\ddot{\phi} = \tau_d + (R_{B1}n_{R_{B1}} \times T_1S_1) + (R_{B1}n_{R_{B1}} \times T_1S_1) + (R_{B1}n_{R_{B1}} \times T_1S_1) + (R_{B1}n_{R_{B1}} \times T_1S_1) \tag{3.26}$$

3.5. Cable Robot Modelling – Motor Dynamics

Figure 3.5 below shows the free-body-diagram of a motor or actuator.

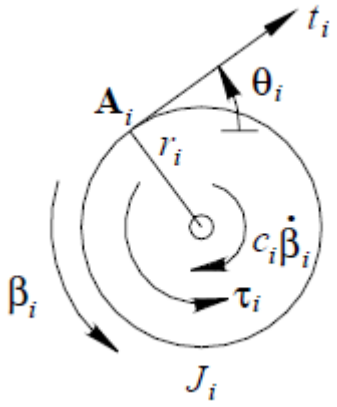

Figure 3.5. End-effector dynamics [35]

The equation for the dynamics of the actuator can then be written as,

$$J\ddot{\beta} + C\dot{\beta} + f = \tau - rT \tag{3.27}$$

Where, the inertia of the motor is represented by J, β is the motor angle, C is the damping, f is coulomb friction, τ is the torque supplied by the motor, r is the radius of the motor or cable spool, and T is the tension in the cable attached to the motor. Assuming the tensions acting in the cable are larger than zero, the tensions can be expressed in terms of the motor torque, speed and acceleration given as below,

$$T = \frac{1}{r}(\tau - J\ddot{\beta} - C\dot{\beta} - f) \tag{3.28}$$

The torque signal from the controller is used to obtain the required tensions.

3.6. Maintaining Positive Cable Tension [35]

It is necessary to always maintain positive cable tension to prevent cable sag, as the torque command obtained from the controller may not always achieve this. Equation 3.29 below shows how the cable forces (T) can be used to obtain the resultant end-effector wrench vector (W_R) by multiplying with the Jacobian matrix (S).

$$\mathbf{ST} = \mathbf{W}_R \tag{3.29}$$

For CDPRs with actuation redundancy, like our case, obtaining the required end-effector wrench W_R can have infinite solutions solving the above equation.

Inverting Equation 3.29 we can write it as,

$$\mathbf{T} = \mathbf{S}^{+}\mathbf{W}_R + (\mathbf{I}_n - \mathbf{S}^{+}\mathbf{S})\mathbf{z} \tag{3.30}$$

The first term of the equation represents the particular solution required to obtain the desired wrench, while the second term is the homogeneous solution that is able to project the vector z into the null space of the Jacobian matrix S. For calculating positive tensions for the planar robot possessing one redundant actuator, the approach is given below,

$$\mathbf{T} = \begin{Bmatrix} t_{P1} \\ t_{P2} \\ t_{P3} \\ t_{P4} \end{Bmatrix} + \alpha \begin{Bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{Bmatrix} \tag{3.31}$$

Here the first vector is the solution of the first term of equation 3.30, while the second vector 'N' is the kernel vector obtained from the matrix S, multiplied by arbitrary scalar α. This approach determines if a specified point is inside the static workspace for a given CDPR. To ensure the point is in the static workspace, all components of $N = [n_1, n_2, n_3, n_4]^T$ must have the same sign, i.e., all $n_i > 0$ or $n_i < 0$. If any $n_i = 0$ then the point is not in the static workspace. If the above conditions are satisfied it is possible to find a scalar α that guarantees positive tensions. Here N is given as,

$$\mathbf{N} = \begin{Bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{Bmatrix} = \begin{Bmatrix} \begin{bmatrix} \cos(\theta_2 - \theta_3 - \theta_4) - \cos(\theta_2 - \theta_3 + \theta_4) \\ +\sin(\theta_2 - \theta_3 + \theta_4) - \sin(\theta_2 + \theta_3 - \theta_4) \end{bmatrix} \\ \begin{bmatrix} \cos(\theta_1 + \theta_3 - \theta_4) - \cos(\theta_1 - \theta_3 - \theta_4) \\ +\sin(\theta_1 + \theta_3 - \theta_4) - \sin(\theta_1 - \theta_3 + \theta_4) \end{bmatrix} \\ \begin{bmatrix} \cos(\theta_1 - \theta_2 - \theta_4) - \cos(\theta_1 + \theta_2 - \theta_4) \\ +\sin(\theta_1 - \theta_2 - \theta_4) + \sin(\theta_1 - \theta_2 + \theta_4) \end{bmatrix} \\ \begin{bmatrix} \cos(\theta_1 + \theta_2 - \theta_3) - \cos(\theta_1 - \theta_2 + \theta_3) \\ -\sin(\theta_1 - \theta_2 - \theta_3) - \sin(\theta_1 - \theta_2 + \theta_3) \end{bmatrix} \end{Bmatrix} \tag{3.32}$$

Here θ_i are the cable angles and the allowable angles are $0^o < \theta_1 < 90^o$, $90^o < \theta_2 < 180^o$, $180^o < \theta_3 < 270^o$, and $270^o < \theta_4 < 360^o$. For these given ranges, the signs of all the components of N are always the same for the square formed by the four motors, decreased by half the length of the end-effector on the sides and increased at the top and bottom for the general configuration shown in Figure 3.1.

α from Equation 3.31 can be calculated as given below,

$$\alpha_i = \frac{(t_{min} - t_{pi})}{n_i} \tag{3.33}$$

Then, the largest α_i is selected at each control cycle. The tensions obtained from Equation 3.31 are input to the positive tension algorithm and the new tensions are used to obtain the actual torque signal given as,

$$\tau_{com} = T_{pos} * r_i \tag{3.34}$$

Here τ_{com} is the torque signal, T_{pos} is the positive tension, and r_i is the motor radius. The torque command is supplied to the motor which generates a torque depending on the torque-speed curve of the motor. Figure 3.6 shows a typical torque-speed curve for a PMDC motor.

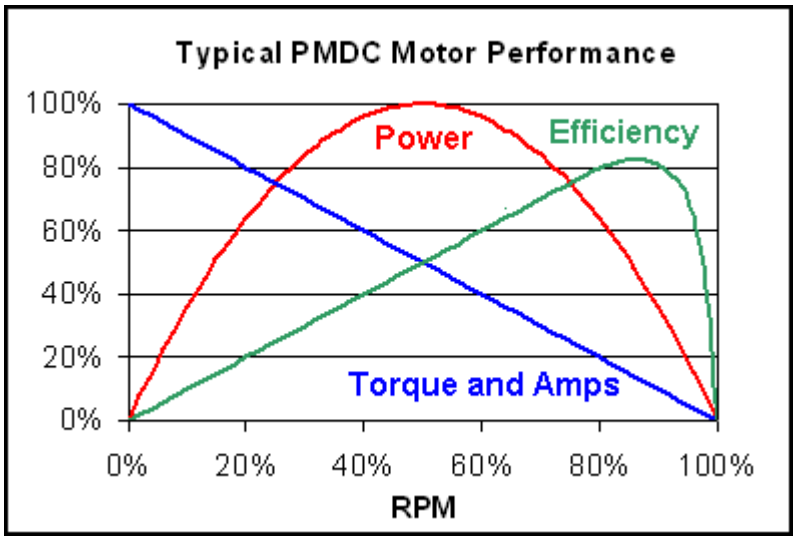

Figure 3.6. An example of a motor's Torque-Speed Curve

The torque generated by the motor is used in Equation 3.28 to obtain the acceleration, velocity, and position of the motors.

3.7. Addition of Cable Elasticity

For adding cable elasticity, the extension of the cable needs to be determined. For a given cable, the actual length of any cable would be given by the actual difference between the location of the points A_i and B_i. This distance is written as A_iB_i. The unstretched length of cable that is released from the motor spool, is given as $\beta.r_i$, and the initial cable lengths when the end-effector is at its initial location are represented as L_0 and can be calculated using equation 3.8.

Each cable has a certain stiffness, $k_{s,i}$ which is given as,

$$k_{s,i} = \frac{EA}{L_i} \tag{3.35}$$

Then the cable forces for elastic cables are given as,

$$T_i = k_{s,i}(A_iB_i + \beta r_i - L_{0,i}) \tag{3.36}$$

Here A_iB_i is the actual cable length, βr_i is the length of the cable unwound from the motor, and $L_{0,i}$ the initial cable length at the initial position. This force is then used in the dynamic equations 3.24, 3.25, and 3.26. Figure 3.7 below shows the diagram of the elastic cable.

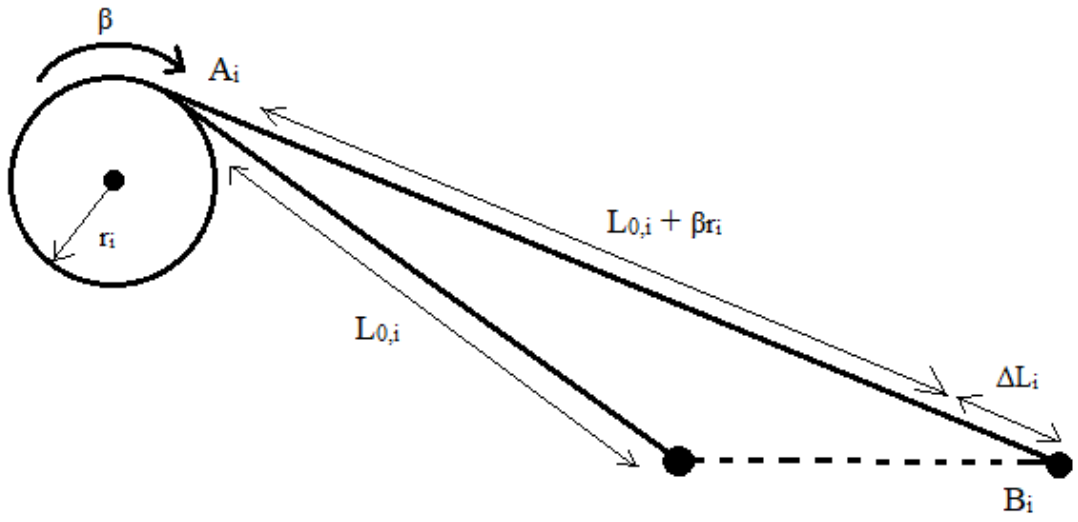

Figure 3.7. Diagram of Length of Elastic Cable

4. TRAJECTORY GENERATION

This chapter will discuss three trajectory generation methods. These are CNC trajectory generation methods which have been implemented for generating trajectories for cable robots. These methods use the trapezoidal acceleration profile to generate different trajectories. Methods 1 and 2 use a combination of linear, circular, and micro-spline interpolation to generate standard and custom trajectories. Method 3 uses splines to generate trajectories based on provided coordinates for a custom trajectory.

4.1. Trapezoidal Acceleration [36]

Figure 4.1 shows the trapezoidal acceleration profile. The profile is divided into 7 distinct sections. Initially the acceleration is zero, but it increases steadily during section 1 until it reaches a value A. In section 2, the acceleration remains constant at this value A. In section 3, the acceleration reduces until it becomes zero. Section 4 consists of a constant velocity section, so the acceleration is equal to zero. Section 5 consists of a deceleration phase, where the deceleration increases until it reaches a value -D, followed by a phase of constant deceleration in section 6, and in section 7 the deceleration slowly decreases to zero. The profile has a starting velocity of f_s, and ending velocity of f_e, and a constant velocity reached during section 5 given as F. The time duration for each section is given as T_1, T_2, T_3, T_4, T_5, T_6, and T_7.

The Jerk during these phases is given as,

$$J(\tau) = \begin{pmatrix} J_1 & 0 \le t < t_1 \\ 0 & t_1 \le t < t_2 \\ -J_3 & t_2 \le t < t_3 \\ 0 & t_3 \le t < t_4 \\ -J_5 & t_4 \le t < t_5 \\ 0 & t_5 \le t < t_6 \\ J_7 & t_6 \le t < t_7 \end{pmatrix} \tag{4.1}$$

The acceleration, velocity, and position are then given as,

$$a(\tau) = a(t_i) + \int_{t_i}^{t} J(\tau) d\tau \tag{4.2}$$

$$f(\tau) = f(t_i) + \int_{t_i}^{t} a(\tau) d\tau \tag{4.3}$$

$$l(\tau) = l(t_i) + \int_{t_i}^{t} f(\tau)d\tau \tag{4.4}$$

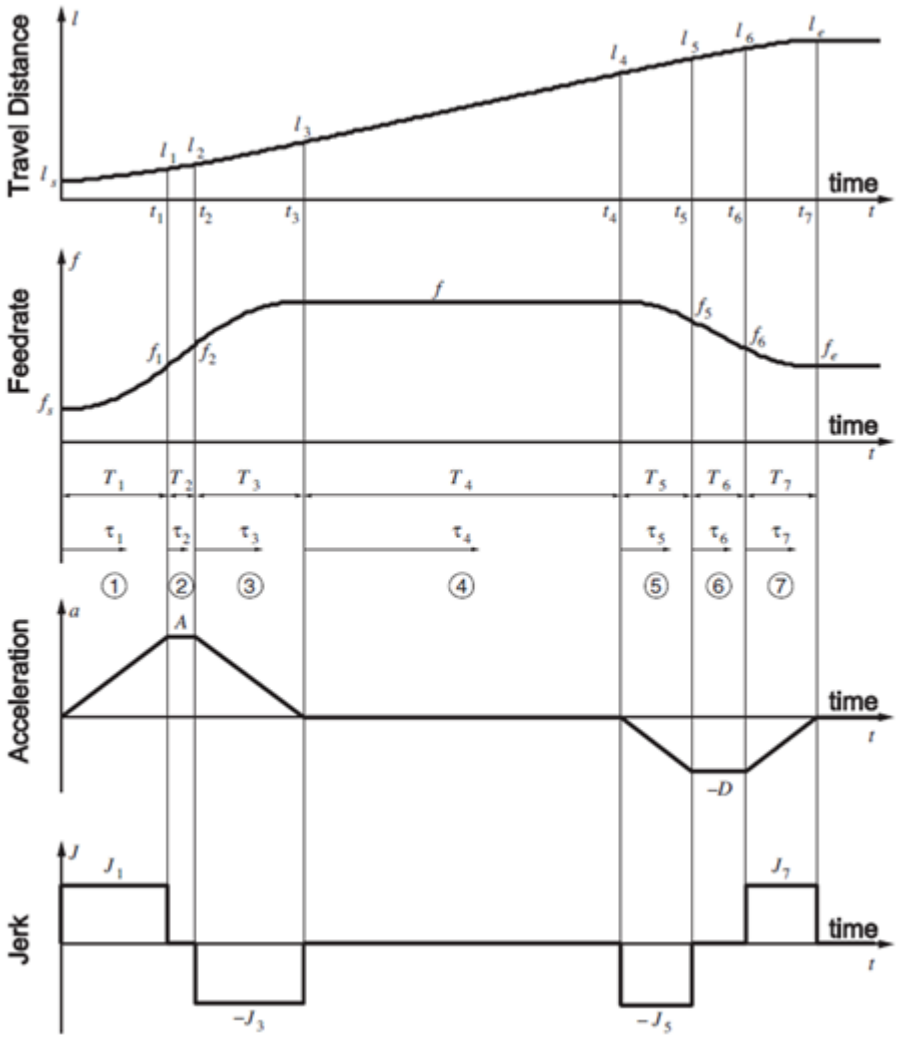

Figure 4.1. Trapezoidal Acceleration Profile

Acceleration is given as,

$$a(\tau) = \begin{cases} J_1\tau_1, & 0 \le t < t_1 \\ A, & t_1 \le t < t_2 \\ A - J_3\tau_3, & t_2 \le t < t_3 \\ 0, & t_3 \le t < t_4 \\ -J_5\tau_5, & t_4 \le t < t_5 \\ -D, & t_5 \le t < t_6 \\ -D + J_7\tau_7, & t_6 \le t \le t_7 \end{cases} \tag{4.5}$$

Here τ_k is a relative time parameter. It begins from zero at the start of each phase k. Velocity is given as,

$$f(\tau) = \begin{cases} f_5 + \frac{1}{2}J_1\tau_1^2, & 0 \le t < t_1, f_5\text{: initial feedrate} \\ f_1 + A\tau_2, & t_1 \le t < t_2, f_1 = f_5 + \frac{1}{2}J_1T_1^2 \\ f_2 + A\tau_3 - \frac{1}{2}J_3\tau_3^2, & t_2 \le t < t_3, f_2 = f_1 + AT_2 \\ f_3, & t_3 \le t < t_4, f_3 = f_2 + AT_3 - \frac{1}{2}J_3T_3^2 = F \\ f_4 - \frac{1}{2}J_5\tau_5^2, & t_4 \le t < t_5, f_4 = f_3 \\ f_5 - D\tau_6, & t_5 \le t < t_6, f_5 = f_4\frac{1}{2}J_5T_5^2 \\ f_6 - D\tau_7 + \frac{1}{2}J_7\tau_7^2, & t_6 \le t \le t_7, f_6 = f_5 - DT_6 \end{cases} \quad (4.6)$$

And distance is given as,

$$f(\tau) = \begin{cases} f_s\tau_1 + \frac{1}{6}J_1\tau_1^3, & 0 \le t < t_1 \\ s_1 + f_1\tau_2\frac{1}{2}A\tau_2^2, & t_1 \le t < t_2,\ s_1 = s_s + f_sT_1 + \frac{1}{6}J_1T_1^3 \\ s_2 + f_2\tau_3 + \frac{1}{2}A\tau_3^2 - \frac{1}{6}J_3\tau_3^3, & t_2 \le t < t_3,\ s_2 = s_1 + f_1T_2 + \frac{1}{2}AT_2^2 \\ s_3 + f_3\tau_4, & t_3 \le t < t_4,\ s_3 = s_2 + f_2T_3 + \frac{1}{2}AT_3^2 - \frac{1}{6}J_3T_3^3 \\ s_4 + f_4\tau_5 - \frac{1}{6}J_5\tau_5^3, & t_4 \le t < t_5, s_4 = s_3 + f_3T_4 \\ s_5 + f_5\tau_6 - \frac{1}{2}D\tau_6^2, & t_5 \le t < t_6, s_5 = s_4 + f_4T_5 - \frac{1}{6}J_5T_5^3 \\ s_6 + f_6\tau_7 - \frac{1}{2}D\tau_7^2 + \frac{1}{6}J_7\tau_7^3, & t_6 \le t \le t_7,\ s_6 = s_5 + f_5T_6 - \frac{1}{2}DT_6^2 \end{cases} \quad (4.7)$$

Where s_k represents the total distance covered at the conclusion of each phase 'k'. The individual distances that are covered during separate phases are given as,

$$l_k = \begin{cases} l_1 = s_1 + f_s T_1 + \frac{1}{6} J_1 T_1^3 \\ l_2 = s_2 - s_1 = f_1 T_2 + \frac{1}{2} A T_2^2 \\ l_3 = s_3 - s_2 = f_2 T_3 + \frac{1}{2} A T_3^2 - \frac{1}{6} J_3 T_3^3 \\ l_4 = s_4 - s_3 = f_3 T_4 \\ l_5 = s_5 - s_4 = f_4 T_5 - \frac{1}{6} J_5 T_5^3 \\ l_6 = s_6 - s_5 = f_5 T_6 - \frac{1}{2} D T_6^2 \\ l_7 = s_7 - s_6 = f_6 T_7 - \frac{1}{2} D T_7^2 + \frac{1}{6} J_7 T_7^3 \end{cases} \quad (4.8)$$

Furthermore it holds that,

$$\begin{aligned} A &= J_1 T_1 = J_3 T_3 \\ D &= J_5 T_5 = J_7 T_7 \end{aligned} \quad (4.9)$$

Keeping in mind the fact that the require velocity ‘F’ must be reached at the conclusion of the 3rd phase,

$$f_3 = F \rightarrow T_2 = \frac{1}{A}\left[F - f_s - \frac{1}{2} J_1 T_1^2 - A T_3 + \frac{1}{2} J_3 T_3^2\right] \quad (4.10)$$

Then the velocity at the end of the 7th phase, reached at the conclusion of the trajectory is given as,

$$f_7 = f_6 - D T_7 + \frac{1}{2} J_7 T_7^2 = f_e \rightarrow T_6 = \frac{1}{D}\left[F - f_e - \frac{1}{2} J_5 T_5^2 - D T_7 + \frac{1}{2} J_7 T_7^2\right] \quad (4.11)$$

Additionally, according to the definition of the acceleration profile, at the conclusion of the 7th phase, the total covered distance must be equal to the total travel length ‘L’

$$s_7 = s_6 + f_6 T_7 - \frac{1}{2} D T_7^2 + \frac{1}{6} J_7 T_7^3 = L \quad (4.12)$$

4.1.1. Linear Interpolation

Figure 4.2 shows the representation for linear interpolation from point P_s to P_e with a velocity f.

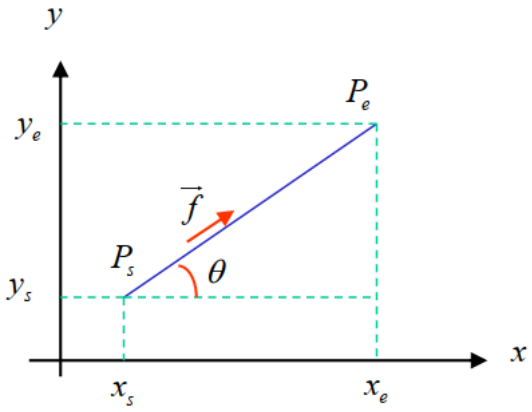

Figure 4.2. Linear Interpolation between two points

The path and angle of travel is given as,

$$L = \sqrt{\Delta x^2 + \Delta y^2}, \Delta x = x_e - x_s, \Delta y = y_e - y_s \tag{4.13}$$

$$\theta = tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) \tag{4.14}$$

Then the trapezoidal acceleration approach described in section 4.1 is used to calculate the variation of displacement, velocity, acceleration and jerk. The displacement variation is added to the initial position to generate the tool trajectory.

4.1.2. Circular Interpolation

Figure 4.3 shows the representation for circular interpolation from point P_s to P_e with a velocity f.

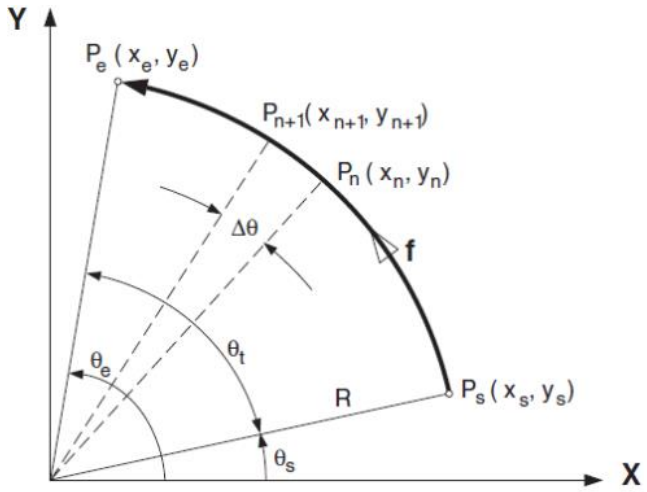

Figure 4.3. Circular Interpolation between two points

The path and angle are given as,

$$L = R(\theta_e - \theta_s) = R\theta_t \tag{4.15}$$

$$tan\theta_e = \frac{y_e}{x_e}, tan\theta_s = \frac{y_s}{x_s} \tag{4.16}$$

The change in position is given as,

$$\delta(k) = l(k) - l(k-1), \delta(k) = R\Delta\theta(k) \tag{4.17}$$

The path as a function of time is given as,

$$l(t) = R\Theta(t) \tag{4.18}$$

$$\Theta(t) = \frac{l(t)}{R} \tag{4.19}$$

$$l(k) = l(k-1) + \delta(k) = l(k-1) + R\Delta\theta(k) \tag{4.20}$$

And the rotational velocity is given as,

$$\omega(t) = \frac{f}{R} \tag{4.21}$$

And the rotational position is,

$$\Theta(t) = \frac{f}{R} * t \tag{4.22}$$

Then the x and y position [x(t), y(t)] and velocities [f_x(t), f_y(t)] are given as,

$$x(t) = R\cos\Theta(t) = R\cos\left(\frac{f}{R}t\right) \tag{4.23}$$

$$y(t) = R\sin\Theta(t) = R\sin\left(\frac{f}{R}t\right) \tag{4.24}$$

$$f_x(t) = \frac{dx}{dt} = -\frac{f}{R}R\sin\Theta(t) = -\frac{f}{R}y(t) \tag{4.25}$$

$$f_y(t) = \frac{dy}{dt} = \frac{f}{R}R\cos\Theta(t) = \frac{f}{R}x(t) \tag{4.26}$$

Similar to the linear interpolation, the trapezoidal acceleration function is used to determine the displacement, velocity, acceleration and jerk variation which is used to obtain the trajectory using the above equations.

4.1.3. Microsplines

To construct different shapes, they are divided into segments consisting of linear and circular sections. These segments are joined using 5th order splines, particularly sharp corners which can be difficult for simulation and control. Figure 4.4 below shows two linear segments joined by a spline.

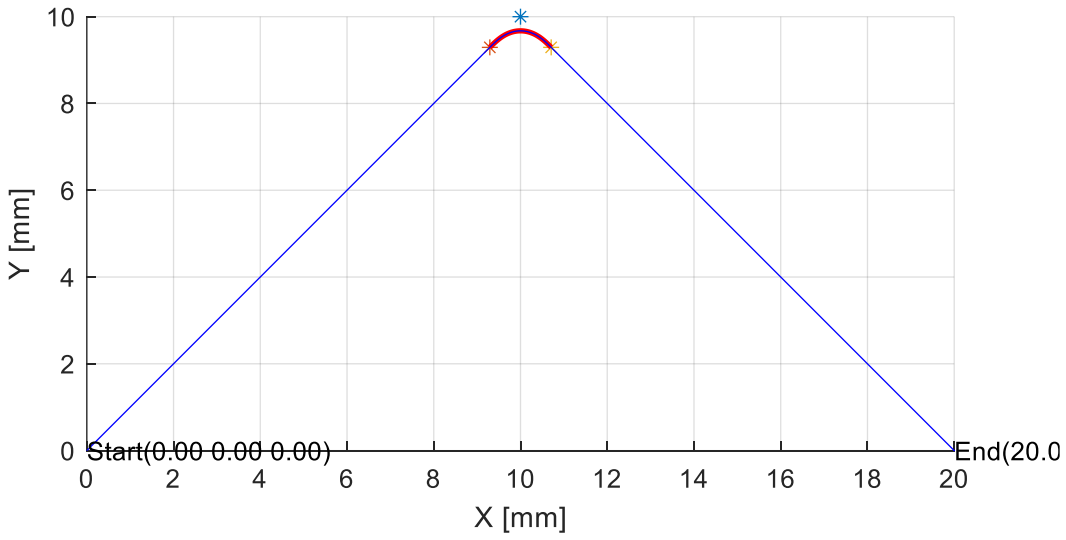

Figure 4.4. Spline at the corner between two line-segments

The x and y equations for the spline is written in the plane formed by the start, end, and corner points for the spline. The equation for the 5th degree polynomial is given as,

$$x = a_{1x}\tau_x^5 + b_{1x}\tau_x^4 + c_{1x}\tau_x^3 + d_{1x}\tau_x^2 + e_{1x}\tau_x^1 + f_{1x} \tag{4.27}$$

Here, a_{1x}, b_{1x}, c_{1x}, d_{1x}, e_{1x}, and f_{1x} are the coefficients of the 5th order polynomial while τ_x is the normalized time. T_x is the total time of spline travel and t is the time so,

$$\tau_x = \frac{t}{T_x} \tag{4.28}$$

Since the equation is of the 5th degree, it has 6 unknowns (coefficients). If the first and second derivatives are taken for $x(\tau_x)$, the initial conditions can be used to obtain the final expressions for the splines. The initial conditions include q_{0x} the spline starting position, q_{1x} the spline ending position, v_{0x} the spline starting velocity, v_{1x} the spline ending velocity, a_{0x} the spline starting acceleration, and a_{1x} the spline ending acceleration. The solution for the equation is presented below,

$$\begin{aligned} x &= q_{0x} + tv_{0x} + (a_{0x}t^2)/2 \\ &- \left(t^5(6q_{0x} - 6q_{1x} + 3T_xv_{0x} + 3T_xv_{1x} + (T_x^2a_{0x})/2 - (T_x^2a_{1x})/2)\right)/T_x^5 \\ &- \left(t^3(10q_{0x} - 10q_{1x} + 6T_xv_{0x} + 4T_xv_{1x} + (3T_x^2a_{0x})/2 - (T_x^2a_{1x})/2)\right)/T_x^3 \\ &+ \left(t^4(15q_{0x} - 15q_{1x} + 8T_xv_{0x} + 7T_xv_{1x} + (3T_x^2a_{0x})/2 - (T_x^2a_{1x})/2)\right)/T_x^4 \end{aligned} \tag{4.29}$$

The expression for velocity is given as,

$$\begin{aligned} \dot{x} &= v_{0x} + ta_{0x} \\ &- \left(5t^4(6q_{0x} - 6q_{1x} + 3T_xv_{0x} + 3T_xv_{1x} + (T_x^2a_{0x})/2 - (T_x^2a_{1x})/2)\right)/T_x^5 \\ &- \left(3t^2(10q_{0x} - 10q_{1x} + 6T_xv_{0x} + 4T_xv_{1x} + (3T_x^2a_{0x})/2 - (T_x^2a_{1x})/2)\right)/T_x^3 \\ &+ \left(4t^3(15q_{0x} - 15q_{1x} + 8T_xv_{0x} + 7T_xv_{1x} + (3T_x^2a_{0x})/2 - (T_x^2a_{1x})/2)\right)/T_x^4 \end{aligned} \tag{4.30}$$

The expression for acceleration is given as,

$$
\begin{aligned}
&\ddot{x} \\
&= a_{0x} \\
&- \left(20t^3(6q_{0x} - 6q_{1x} + 3T_x v_{0x} + 3T_x v_{1x} + (T_x^2 a_{0x})/2 - (T_x^2 a_{1x})/2)\right)/T_x^5 \\
&- \left(6t(10q_{0x} - 10q_{1x} + 6T_x v_{0x} + 4T_x v_{1x} + (3T_x^2 a_{0x})/2 - (T_x^2 a_{1x})/2)\right)/T_x^3 \\
&+ \left(12t^2(15q_{0x} - 15q_{1x} + 8T_x v_{0x} + 7T_x v_{1x} + (3T_x^2 a_{0x})/2 - (T_x^2 a_{1x})/2)\right)/T_x^4
\end{aligned}
\tag{4.31}
$$

And the expression for jerk is given as,

$$
\begin{aligned}
&\dddot{x} \\
&= -\left(60t^2(6q_{0x} - 6q_{1x} + 3T_x v_{0x} + 3T_x v_{1x} + (T_x^2 a_{0x})/2 - (T_x^2 a_{1x})/2)\right)/T_x^5 \\
&- \left(6(10q_{0x} - 10q_{1x} + 6T_x v_{0x} + 4T_x v_{1x} + (3T_x^2 a_{0x})/2 - (T_x^2 a_{1x})/2)\right)/T_x^3 \\
&+ \left(24t(15q_{0x} - 15q_{1x} + 8T_x v_{0x} + 7T_x v_{1x} + (3T_x^2 a_{0x})/2 - (T_x^2 a_{1x})/2)\right)/T_x^4
\end{aligned}
\tag{4.32}
$$

These same equations can be written for the y-axis in the same manner and these provide the position, speed, acceleration, and jerk profile for the spline.

4.1.4. Determination of Spline Start and End Points

The coordinates of the Start-End points are calculated according to the type of segment (Arc or Line) and the definition of the cornering tolerance. A larger cornering tolerance results in a larger spline.

- As for the line segment, the start/end point of the spline is found by moving away from the corner point by the cornering tolerance on the line. The point is determined to be the start or end point based on its position relative to the edge of the segment.
- In the Arc segment, a circle with cornering tolerance radius is drawn at the corner point on the Arc plane. The intersection point of this drawn circle and the Arc is the start/end point of the spline. When finding the intersection point, the intersection formula of two circles with known center and radius is used. Since two circles can intersect at two different points, the correct intersection point is determined by taking the point close to the corner point on the Arc. The intersection point with the smaller distance is the spline start/end point. Whether it is a starting or ending point is determined by whether the segment is before or after the corner point.

Figure 4.5 below shows an example of the beginning and ending points of the spline when two line segments are being connected. The distance between the start/end points and the corner point is equal to the cornering tolerance.

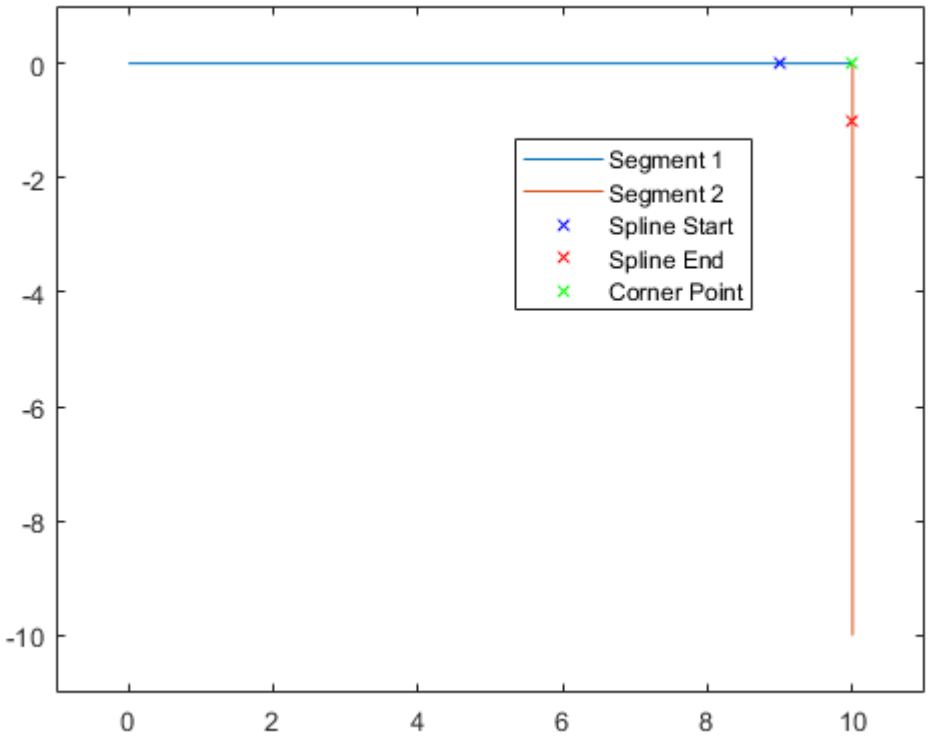

Figure 4.5. Spline Start/Corner/End points between two line segments

Figure 4.6 below shows the start and end points of the spline when a line segment is being connected with an arc segment.

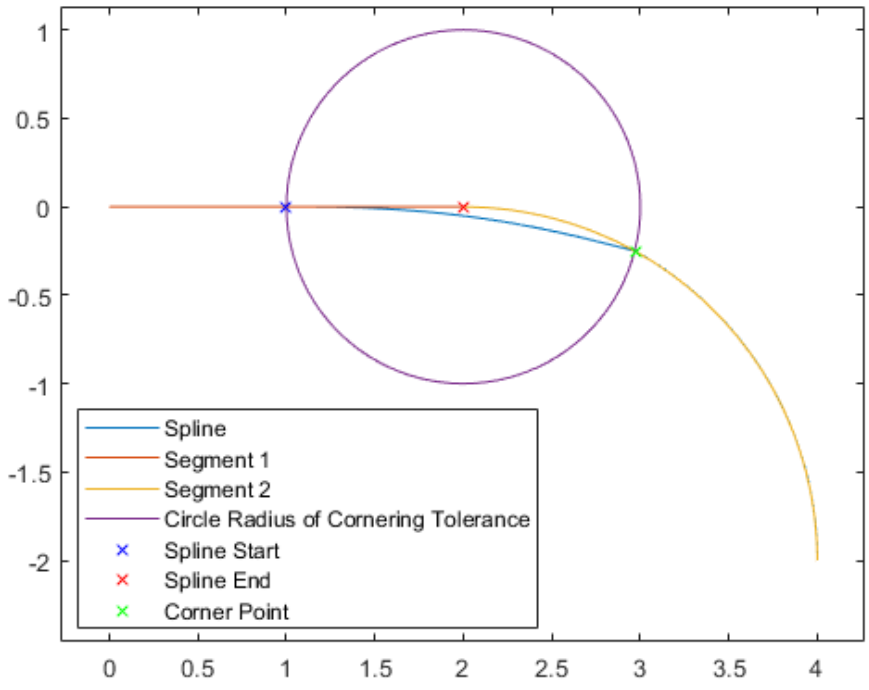

Figure 4.6. Spline Start/Corner/End points between a line and arc segment

Figure 4.7 below shows the start and end points of the spline when two arc segments are being connected.

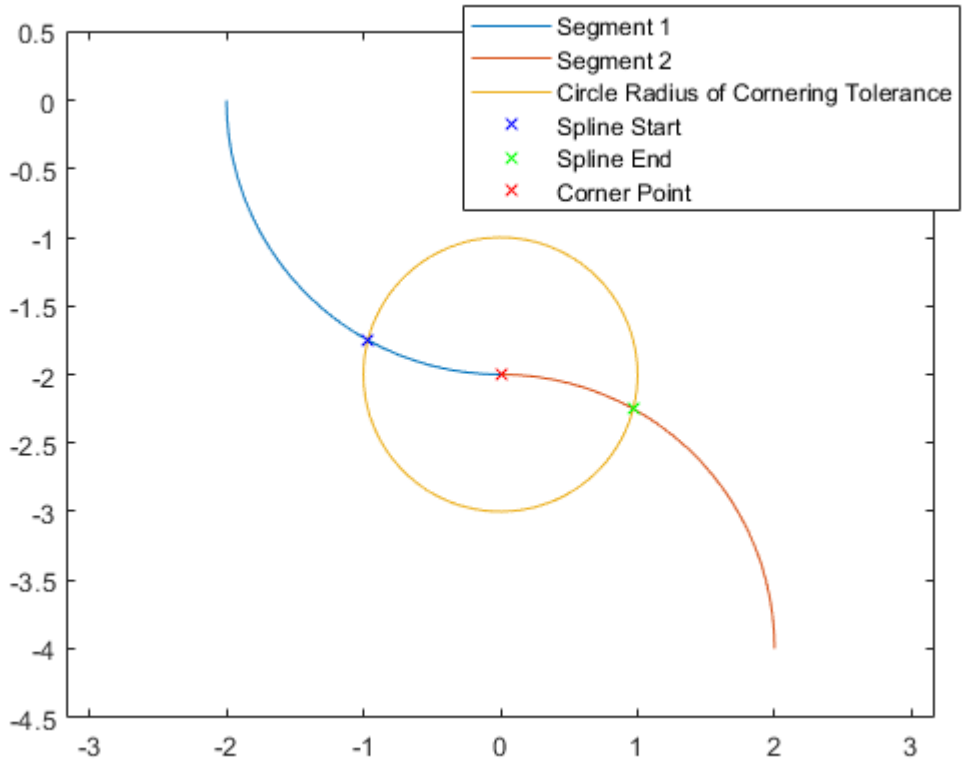

Figure 4.7. Spline Start/Corner/End points between two arc segments

4.1.5. Spline Start/End Acceleration and Velocity

The starting and ending acceleration for the spline depends on the type of segment before and after the corner point of the spline. Table 4.1 below summarizes the different scenarios.

Table 4.1. Starting and Ending accelerations for spline for different segments

Segment before corner point	**Segment after corner point**	**Spline Start Acceleration (a_0)**	**Spline End Acceleration (a_1)**
Line	Arc	0	${v_1}^2/R$
Line	Line	0	0
Arc	Line	${v_0}^2/R$	0
Arc	Arc	${v_0}^2/R$	${v_1}^2/R$

Here, v_0 and v_1 are the start and end velocities of the spline and R is the radius of the arc segment. The spline Start-End speeds are taken from the speed of the segments. The speed of the segment before the corner point is the starting speed of the spline. The speed of the segment after the corner point is the spline end speed.

4.1.6. Optimization of Speed and Acceleration Initial Conditions

The above formulations are used to develop the spline-based trajectory, however if an appropriate spline cannot be found, the bisection method is used to determine the proper maximum speed and acceleration.

In the Bisection Method, the spline start (v_0) and end (v_1) speeds taken from the feed rate are multiplied by a reduction ratio of k. The k value is initially taken as 1. The value of k is equal to half the sum of the maximum drop rate (k_f) and the minimum drop rate (k_i). With the new k value, velocities and accelerations are reduced and a new spline is searched. If a suitable spline is found, the minimum reduction rate k_i is set to the current value of k. If a suitable spline cannot be found, the maximum reduction rate k_f is set to the current value of k. Here, the k values in the range in which the appropriate spline is found are accumulated and the difference between them is examined. The iteration is completed when the difference is less than 0.005. The beginning and ending accelerations within the spline occur only when there is an Arc, due to centripetal acceleration. That is, in other cases, the initial and final accelerations are 0. Since only centripetal acceleration acts, the acceleration reduction ratio is the square of the velocity reduction ratio. In addition, since the speed limits along the spline are desired to be lower than the spline starting and spline ending speeds, the max and minimum speed limits are adjusted according to the reduced spline starting and ending speeds.

4.1.7. Determination of Correct Spline Timing

For proper spline generation, the correct travel time of the spline must be calculated so that the total simulation time is not affected by the addition of the spline. The total distance covered by the original trajectory without the addition of splines is two times the cornering tolerance.

Then using the velocity of the trajectory where the spline is added, the travel time for the spline is given as,

$$SplineTravelTime = \frac{2 * CorneringTolerance}{Velocity} \tag{4.33}$$

4.2. Trajectory Generation 1 – Standard Shapes

Trajectory Generation 1 uses the above defined formulations to develop many different trajectories of standard shapes such as circle, square, rectangle etc. Figure 4.8 below shows a standard rectangular shape which has been constructed using a series of linear and arc segments.

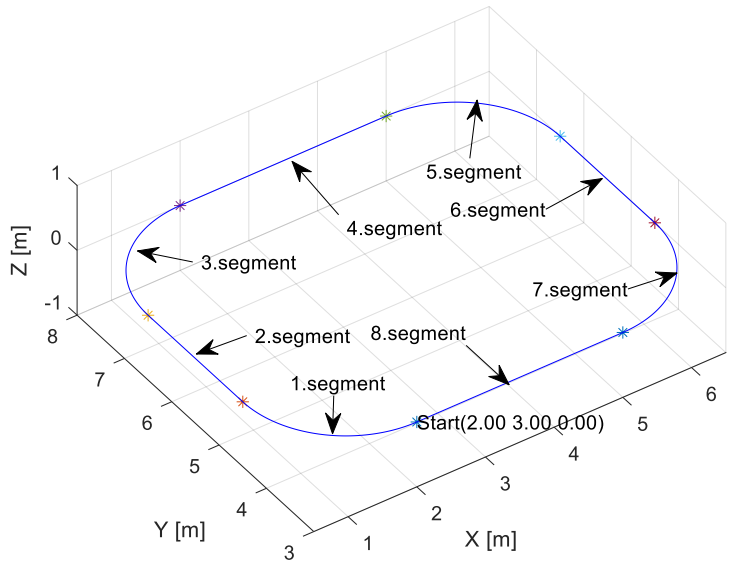

Figure 4.8. Standard trajectory composed of linear and arc segments

This approach has 3 types of trajectory generation:

1. Point-to-Point: In this approach each segment starts and ends at zero velocity for smooth transition between each segment.
2. Continuous: In this approach the trajectory has a continuous velocity between segments

3. Microsplines: In this approach the individual segments are joined using splines as defined in section 4.1.3.

4.3. Trajectory Generation 2 – Custom Data

Trajectory Generation 2 accepts custom trajectory data as an input. This custom trajectory data is resampled at the desired sampling rate, but this trajectory can have a certain starting velocity or acceleration so splines are used to accelerate the end-effector to the starting conditions of the custom trajectory and decelerate at the end of the custom trajectory. Figure 4.9 below shows a custom trajectory in red and splines generated at the beginning and end of the trajectory. Using this approach, trajectories with non-uniform acceleration and velocities can be simulated.

For the first spline, the angle at the beginning of the custom trajectory data and a cornering tolerance value is used to determine the starting and corner point for the spline, whereas the end point is the first point of the custom trajectory data. Meanwhile, for the second spline, the starting point is the last point of the custom trajectory data, and similar to the first spline, the angle at the end of the custom trajectory data and the cornering tolerance are used to determine the corner and end point of the spline.

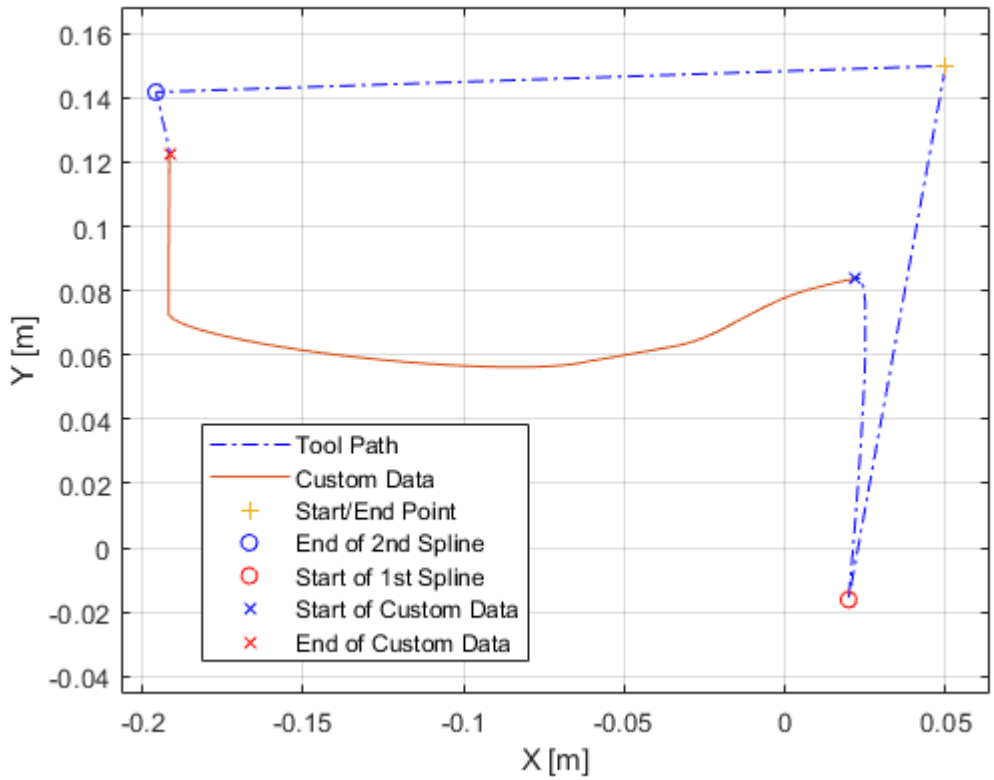

Figure 4.9. Trajectory generated for custom data

Table 4.2 below shows the initial conditions for both splines.

Table 4.2. Initial and Final Conditions for first and second spline

	First Spline	Second Spline
Initial Position	Calculated according to the angle at the beginning of custom trajectory and cornering tolerance	The last point in the custom trajectory data
Initial Speed	The initial speed is zero, the end-effector starts from rest	The initial velocity is the velocity at the end of the custom trajectory
Initial Acceleration	The initial acceleration is zero, the end-effector starts from rest	The initial acceleration is the acceleration at the end of the custom trajectory

Final Position	The first point in the custom trajectory data	Calculated according to the angle at the end of custom trajectory and cornering tolerance
Final Velocity	The final velocity is the velocity at the beginning of the custom trajectory	The final speed is zero, the end-effector comes to rest at the end of the spline
Final Acceleration	The final acceleration is the acceleration at the beginning of the custom trajectory	The final acceleration is zero, the end-effector comes to rest at the end of the spline

The start and corner point for spline 1 are calculated as,

$$P_{start\_spline\_1} = \begin{bmatrix} x(1)_{customdata} - \cos(\theta_{customdata}) * CT * 2 \\ y(1)_{customdata} - \sin(\theta_{customdata}) * CT * 2 \end{bmatrix}^T \tag{4.34}$$

$$P_{corner\_spline\_1} = \begin{bmatrix} x(1)_{customdata} - \cos(\theta_{customdata}) * CT \\ y(1)_{customdata} - \sin(\theta_{customdata}) * CT \end{bmatrix}^T \tag{4.35}$$

Here, CT is the cornering tolerance, $\theta_{customdata}$ is the angle at the beginning of the custom data which can be calculated using the first two data points as,

$$\begin{aligned} \Delta x &= x(2)_{customdata} - x(1)_{customdata} \\ \Delta y &= y(2)_{customdata} - y(1)_{customdata} \\ \theta_{customdata} &= \text{atan}\,(\Delta y\,, \Delta x) \end{aligned} \tag{4.36}$$

Additionally, it may also be desired that the end-effector should start and end from a determined point instead of arbitrary starting and ending points. This can be easily achieved by using linear interpolation to move the end-effector to the desired starting point of the first spline, and then from the ending point of the second spline to the desired end point.

4.4. Trajectory Generation 3 - Jerk Limited Trajectory Generation Using 5th Order Spline Interpolation [36]

This trajectory generation technique also uses the trapezoidal acceleration profile described in section 4.1. This approach is different from the previous approaches as it is capable of generating non-standard shapes such as a spiral trajectory where the radius of the arcs is constantly changing. 5th Order splines guarantee continuity of the trajectory up to the second derivative at each point. The trajectory generation procedure consists of three steps:

1. Path Generation Using Quintic (5th Order) Splines
2. Feedrate Generation for the Path
3. Trajectory Resampling at the Control Loop Frequency

Figure 4.10 below illustrates the process.

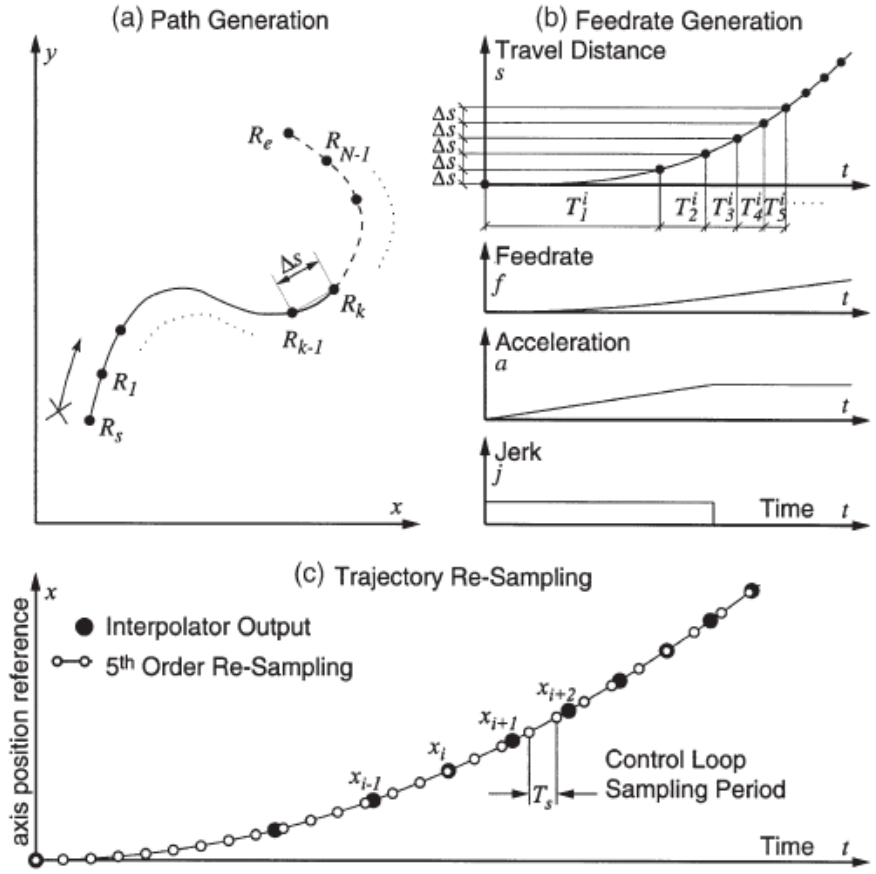

Figure 4.10. Trajectory generation procedure for Algorithm 3

4.4.1. Quintic Spline Path Generation

In CNC machining, a series of reference points are input to the machine to realize tool motion and the process of generating these points is known as interpolation. Simpler techniques like linear and circular interpolation were shown in sections 4.1.1 and 4.1.2, but quintic splines aim to connect N number of reference knots with N-1 number of fifth order splines as shown in Figure 4.11 below.

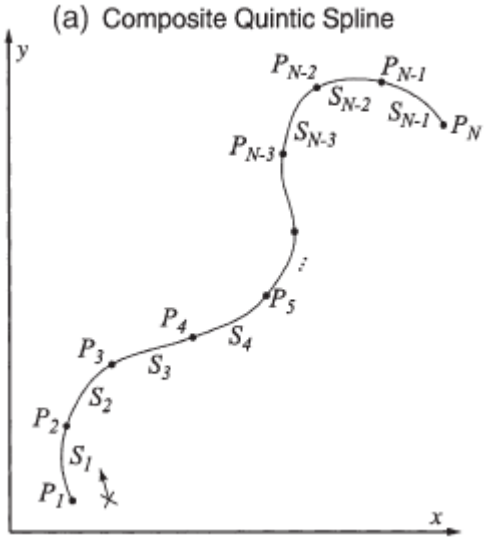

Figure 4.11. Path generation with quintic splines

These splines make up an entire composite curve where continuity is guaranteed up to the second derivative for the entire path. The point where two splines meet is called a knot represented as P_i. To estimate the first and second derivatives at these points, a 3rd order polynomial can be fit between the points P_{i-1}, P_{i+1}, P_i, and P_{i+2}, given as

$$Q_i(u) = a_i u^3 + b_i u^2 + C_i u + d_i \tag{4.37}$$

Here u is the cord length of each segment between knots. For a two-dimensional case,

$$Q_i = \begin{bmatrix} Q_{xi} \\ Q_{yi} \end{bmatrix}, a_i = \begin{bmatrix} a_{xi} \\ a_{yi} \end{bmatrix}, \boldsymbol{b}_i = \begin{bmatrix} b_{xi} \\ b_{yi} \end{bmatrix}, c_i = \begin{bmatrix} c_{xi} \\ c_{yi} \end{bmatrix}, d_i = \begin{bmatrix} d_{xi} \\ d_{yi} \end{bmatrix} \tag{4.38}$$

Now the cord length between two consecutive knots P_{i-1} and P_i is give as,

$$l_{i-1} = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{4.39}$$

Also, we can define additional terms as,

$$l_{i-1,i} \triangle= l_{i-1} + l_i, l_{i-1,i+1} \triangle= l_{i-1} + l_i + l_{i+1} \tag{4.40}$$

Then, the coefficients of the polynomial are given as,

$$a_i = \frac{1}{\Delta}\begin{bmatrix}\Delta_{ax}\\ \Delta_{ay}\end{bmatrix}, b_i = \frac{1}{\Delta}\begin{bmatrix}\Delta_{bx}\\ \Delta_{by}\end{bmatrix}, c_i = \frac{1}{\Delta}\begin{bmatrix}\Delta_{cx}\\ \Delta_{cy}\end{bmatrix}, d_i = \begin{bmatrix}x_{i-1}\\ y_{i-1}\end{bmatrix} \tag{4.41}$$

Where,

$$\Delta = \begin{vmatrix} l_{i-1}^3 & l_{i-1}^2 & l_{i-1} \\ l_{i-1,i}^3 & l_{i-1,i}^2 & l_{i-1,i} \\ l_{i-1,i+1}^3 & l_{i-1,i+1}^2 & l_{i-1,i+1} \end{vmatrix}, \Delta ax = \begin{vmatrix} x_i - x_{i-1} & l_{i-1}^2 & l_{i-1} \\ x_{i+1} - x_{i-1} & l_{i-1,i}^2 & l_{i-1,i} \\ x_{i+2} - x_{i-1} & l_{i-1,i+1}^2 & l_{i-1,i+1} \end{vmatrix} \tag{4.42}$$

$$\begin{aligned} \Delta_{bx} &= \begin{vmatrix} l_{i-1}^3 & x_i - x_{i-1} & l_{i-1} \\ l_{i-1,i}^3 & x_{i+1} - x_{i-1} & l_{i-1,i} \\ l_{i-1,i+1}^3 & x_{i+2} - x_{i-1} & l_{i-1,i+1} \end{vmatrix}, \Delta_{cx} \\ &= \begin{vmatrix} l_{i-1}^3 & l_{i-1}^2 & x_i - x_{i-1} \\ l_{i-1,i}^3 & l_{i-1,i}^2 & x_{i+1} - x_{i-1} \\ l_{i-1,i+1}^3 & l_{i-1,i+1}^2 & x_{i+2} - x_{i-1} \end{vmatrix} \end{aligned} \tag{4.43}$$

Δ_{ay}, Δ_{by}, and Δ_{cy} can be calculated similarly by substituting the x terms with y. Then the first derivative (t_i) and the second derivative (n_i) at each know can be calculated as:

$$t_i \triangle= \begin{bmatrix}t_{xi}\\ t_{yi}\end{bmatrix} = \frac{\mathrm{d}Q_i}{\mathrm{d}u}|u = l_{i-1} = (3a_i u^2 + 2b_i u + c_i)|_{u=l} \tag{4.44}$$

$$n_i \triangle= \begin{bmatrix}n_{xi}\\ n_{vi}\end{bmatrix} = \frac{\mathrm{d}^2 Q_i}{\mathrm{d}u^2}|u = l_i - 1 = (6a_i u + 2b_i)|_{u=l_{i-1}} \tag{4.45}$$

However, the knots P_1, P_{N-1}, and P_N don't have enough points before or after them so t_1 and n_1 is estimated using,

$$Q_2(u)|_{u=0} \tag{4.46}$$

t_{N-1} and n_{N-1} are obtained using,

$$Q_{N-2}(u)|_{u=I_{N-3}+l_{N-2}} \tag{4.47}$$

t_N and n_N are obtained using,

$$Q_{N-2}(u)|_{u=l_{N-3}+l_{N-2}+l_{N-1}} \tag{4.48}$$

This gives the following equation,

$$t_1 = 3a_2 u^2 + 2b_2 u + c_2, n_1 = 6a_2 u + 2b_2 \text{ for } u = 0 \tag{4.49}$$

$$t_{N-1} = 3a_{N-2}u^2 + 2b_{N-2}u + c_{N-2}, n_{N-1} = 6a_{N-2}u + 2b_{N-2} \quad \text{for} \quad u = l_{N-3} + l_{N-2} \tag{4.50}$$

$$t_N = 3a_{N-2}u^2 + 2b_{N-2}u + c_{N-2}, n_N = 6a_{N-2}u + 2b_{N-2} \text{ for } u = l_{N-3} + l_{N-2} + l_{N-1} \tag{4.51}$$

The expression for the quintic spline is given as,

$$S_i(u) = A_i u^5 + B_i u^4 + C_i u^3 + D_i u^2 + E_i u + F_i \tag{4.52}$$

Which is fit between consecutive knots. Then for two-dimensional case,

$$S_i = \begin{bmatrix} S_{xi} \\ S_{yi} \end{bmatrix}, A_i = \begin{bmatrix} A_{xi} \\ A_{yi} \end{bmatrix}, B_i = \begin{bmatrix} B_{xi} \\ B_{yi} \end{bmatrix}, \dots, F_i = \begin{bmatrix} F_{xi} \\ F_{yi} \end{bmatrix} \tag{4.53}$$

Now the following boundary conditions must be considered, to fit a spline between two consecutive points P_i and P_{i+1}

$$\left.\begin{aligned} & S_i(u)|_{u=0} = p_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, S_i(u)|_{u=l_i} = p_{i+1} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} \\ & \left.\frac{\mathrm{d}S_i(u)}{\mathrm{d}u}\right|_{u=0=} = t_i = \begin{bmatrix} t_{xi} \\ t_{yi} \end{bmatrix}, \left.\frac{\mathrm{d}S_i(u)}{\mathrm{d}u}\right|_{u=l_i} = t_{i+1} = \begin{bmatrix} t_{x,i+1} \\ t_{y,i+1} \end{bmatrix} \\ & \left.\frac{\mathrm{d}^2S_i(u)}{\mathrm{d}u^2}\right|_{u=0=} = n_i = \begin{bmatrix} n_{xi} \\ n_{yi} \end{bmatrix}, \left.\frac{\mathrm{d}^2S_i(u)}{\mathrm{d}u^2}\right|_{u=l_i} = n_{i+1} = \begin{bmatrix} n_{x,i+1} \\ n_{y,i+1} \end{bmatrix} \end{aligned}\right\} \tag{4.54}$$

The x-axis solution for equation 4.52 is given below. The solution in the y-direction can also be easily obtained by substituting each 'x' term with 'y'.

$$A_{xi} = \frac{1}{l_i^5}\left[6(x_{i+1} - x_i) - 3\left(t_{x,i+1} + t_{xi}\right)l_i + 0.5\left(n_{x,i+1} - n_{xi}\right)l_i^2\right]$$

$$B_{xi} = \frac{1}{l_i^4}\left[15(x_i - x_{i+1}) + \left(7t_{x,i+1} + 8t_{xi}\right)l_i + \left(1.5n_{xi} - n_{x,i+1}\right)l_i^2\right]$$

$$C_{xi} = \frac{1}{l_i^3}\left[10(x_{i+1} - x_i) - \left(4t_{x,i+1} + 6t_{xi}\right)l_i - \left(1.5n_{xi} - 0.5n_{x,i+1}\right)l_i^2\right] \tag{4.55}$$

$$D_{xi} = 0.5n_{xi}$$

$$E_{xi} = t_{xi}$$

$$F_{xi} = x_i$$

Once the coefficients are determined, the total travel length can be calculated as,

$$L = \sum_{i=1}^{N-1} s_i = \sum_{i=1}^{N-1} \int_0^{s_i} \mathrm{d}s \tag{4.56}$$

Here L is length of travel or total length of the trajectory while s_i represents the length of each individual spline segment.

The length for each arc s_i of the spine is equal to each chord length 'l_i' from each spline split into M_i segments. Then, for each increment of the chord, the corresponding spline points are calculated. Then the change in the x and y positions can be used to obtain the displacement s_i. Therefore, M_i can be written as,

$$M_i = \text{round}\left(\frac{l_i}{F \cdot T_s}\right) \tag{4.57}$$

Then cord increment becomes,

$$dl_i = l_i / M_i \tag{4.58}$$

The M_i points on the i^{th} spline are given as,

$$\begin{bmatrix} x_{ij} \\ y_{ij} \end{bmatrix} = A_i(j \cdot dl_i)^5 + B_i(j \cdot dl_i)^4 + C_i(j \cdot dl_i)^3 + D_i(j \cdot dl_i)^2 + E_i(j \cdot dl_i) + F_i \tag{4.59}$$

The arc length between 2 successive points is given as,

$$\mathrm{d}s_{ij} \cong \sqrt{\left(\mathrm{d}x_{ij}\right)^2 + \left(\mathrm{d}y_{ij}\right)^2} = \sqrt{\left(x_{ij} - x_{i,j-1}\right)^2 + \left(y_{ij} - y_{i,j-1}\right)^2} \tag{4.60}$$

Then the total travel length becomes,

$$\begin{aligned} L = \sum_{i=1}^{N-1} \int_0^{s_i} \mathrm{d}s &\cong \sum_{i=1}^{N-1} \sum_{j=1}^{M_i} \mathrm{d}s_{ij} \\ &= \sum_{i=1}^{N-1} \sum_{j=1}^{M_i} \sqrt{\left(x_{ij} - x_{i,j-1}\right)^2 + \left(y_{ij} - y_{i,j-1}\right)^2} \end{aligned} \tag{4.61}$$

Sometimes velocity fluctuations occur due to the actual arc length and the chord length of segments being different different. For interpolation without this fluctuation in

velocity, the magnitude of displacement between each interpolation step is kept constant. This increment Δs is defined such that the speed at the time step being equal to the sampling rate T_i is not larger than the maximum speed requirement.

So if the total travel length (L) was covered with the maximum velocity (f_{max}), then the step size is given as,

$$N_i = \text{round}\left(\frac{L}{f_{\max} T_s}\right) \tag{4.62}$$

Which gives,

$$\Delta s = L/N_i \tag{4.63}$$

Path increment can also be given as,

$$\Delta s = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{4.64}$$

Here the equations are given as

$$\left.\begin{aligned} \Delta x = x_{i,j+1} - x_{ij} = A_{xi}u^5 + B_{xi}u^4 + C_{xi}u^3 + D_{xi}u^2 + E_{xi}u + F_{xi} - x_{ij} \\ \Delta y = y_{i,j+1} - y_{ij} = A_{yi}u^5 + B_{yi}u^4 + C_{yi}u^3 + D_{yi}u^2 + E_{yi}u + F_{yi} - y_{ij} \end{aligned}\right\} \tag{4.65}$$

It is necessary to find the variable 'u'. It is given by the tenth order polynomial,

$$g(u) = \alpha_0 u^{10} + \alpha_1 u^9 + \cdots + \alpha_{10} \tag{4.66}$$

$$\begin{aligned} \alpha_0 &= A_{xi}^2 + A_{yi}^2 \\ \alpha_1 &= 2\left(A_{xi}B_{xi} + A_{yi}B_{yi}\right) \\ \alpha_2 &= B_{xi}^2 + B_{yi}^2 + 2\left(A_{xi}C_{xi} + A_{yi}C_{yi}\right) \\ \alpha_3 &= 2\left(B_{xi}C_{xi} + B_{yi}C_{yi} + A_{xi}D_{xi} + A_{yi}D_{yi}\right) \\ \alpha_4 &= C_{xi}^2 + C_{yi}^2 + 2\left(A_{xi}E_{xi} + A_{yi}E_{yi} + B_{xi}D_{xi} + B_{yi}D_{yi}\right) \\ \alpha_5 &= 2\left(A_{xi}F'_{xi} + A_{yi}F'_{\ yi} + B_{xi}E_{xi} + B_{yi}E_{yi} + C_{xi}D_{xi} + C_{yi}D_{yi}\right) \\ \alpha_6 &= D_{xi}^2 + D_{yi}^2 + 2\left(B_{xi}F'_{\ xi} + B_{yi}F'_{\ yi} + C_{xi}E_{xi} + C_{yi}E_{yi}\right) \\ \alpha_7 &= 2\left(D_{xi}E_{xi} + D_{yi}E_{yi} + C_{xi}F'_{\ xi} + C_{yi}F'_{\ yi}\right) \\ \alpha_8 &= E_{xi}^2 + E_{yi}^2 + 2\left(D_{xi}F'_{\ xi} + D_{yi}F'_{\ yi}\right) \\ \alpha_{99} &= 2\left(E_{xi}F'_{xi} + E_{yi}F''\right) \\ \alpha_{10} &= F'^2 + F'^2_{\ yi} - (\Delta s)^2 \end{aligned} \tag{4.67}$$

The equation can be solved using iterative solving methods. It was solved using the Newton-Rhapson method as shown below. A good initial guess for this to work well is $\Delta s * j$, where j is the number of steps in each spline segment.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.68}$$

4.4.2. Trajectory Generation Algorithm

This algorithm requires some inputs which define the acceleration profile. These are the following:

- Sampling Time (T_s)
- Total Distance Covered (L)
- Total Interpolation Steps (N_i)
- Initial, desired, and final velocities (f_s, F, and f_e, respectively)
- Magnitude of desired Acceleration (A) and Deceleration (D)
- Maximum Jerk Magnitude (J)

First it must be checked that the Acceleration, Deceleration, and Jerk variables have the proper sign.

$$\left.\begin{array}{l} A = \text{sgn}\,(F - f_s) \cdot |A|,\ J_1 = J_3 = \text{sgn}\,(A) \cdot |J| \\ D = \text{sgn}\,(F - f_e) \cdot |D|, J_5 = J_7 = \text{sgn}\,(D) \cdot |J| \end{array}\right\} \tag{4.69}$$

In the trajectory's initial phase, if the value of A is less than zero, this signifies deceleration, however if the value of D is less than zero, this denotes acceleration. If both A and D are zero in value, then the trajectory only has a constant speed phase.

The input for the total interpolation steps must also be realistic. If the magnitudes of both acceleration and deceleration variables is not zero, then $N_i \geq 4$ so that the acceleration and deceleration phases (1, 3, 5, 7) are all able to be active. However, if either the magnitude of the acceleration of deceleration is zero, then it is enough to have $N_i \geq 2$ as only two phases are required minimum in this case. Similarly, if no acceleration or

deceleration phases are present then the condition becomes $N_i \geq 1$ which ensures that the length of travel doesn't become negative and only a constant speed phase would be present. Based on the supplied values, the following conditions must be checked and satisfied.

Jerk Check

The magnitude of the required jerk should not violate the demanded acceleration and deceleration conditions. Equation 4.70 is used to verify this condition.

$$J \leq \min\left(\frac{|A|}{T_s}, \frac{|D|}{T_s}\right) \tag{4.70}$$

If the condition is not satisfied (when acceleration and deceleration are present) then jerk is set to the minimum value as,

$$J = \min\left(\frac{|A|}{T}, \frac{|D|}{T}\right) \tag{4.71}$$

If either the acceleration or deceleration is zero, then only the corresponding non-zero term is used to check this condition. If both are zero however, then it is not necessary to perform this check.

Acceleration Check

If the acceleration is not zero, then the velocity at the conclusion of the third stage must be equal to the required maximum velocity 'F' or 'f_{max}'. This process may or may not include a constant acceleration second phase. The time for the second phase 'T_2' is then obtained from Equation 4.10 which is either zero or greater than zero. The acceleration check is then given as,

$$T_2 = \frac{F - f_s}{A} - \frac{A}{J_1} \geq 0 \tag{4.72}$$

This check must only my performed if acceleration is not zero. If equation 4.72 is not satisfied then T_2 is set to zero, and the actual acceleration can be obtained by Equation 4.73 which gives the maximum possible magnitude. The jerks and time for the first and third phase are also adjusted according to $J_1=J_3$ and $T_3=A/J_3$.

$$A = \text{sgn}\,(A) \cdot \sqrt{J_1(F - f_s)} \tag{4.73}$$

Deceleration Check

Similarly, if the deceleration is not zero, then the final velocity at the conclusion of the last stage must be equal to the required end velocity 'f_e'. This process may or may not include a constant deceleration sixth phase. The time for the sixth phase 'T_6' is then obtained from Equation 4.11 which is either zero or greater than zero. The deceleration check is then given as,

$$T_6 = \frac{F - f_e}{D} - \frac{D}{J_5} \geq 0 \tag{4.74}$$

This check must only my performed if deceleration is not zero. If equation 4.74 is not satisfied then T_6 is set to zero, and the actual deceleration can be obtained by Equation 4.75 which gives the maximum possible magnitude. The jerks and time for the fifth and seventh phase are also adjusted according to J_5=J_7 and T_7=D/J_7.

$$D = \text{sgn}\,(D) \cdot \sqrt{J_5(F - f_e)} \tag{4.75}$$

Travel Total Length Check

The total length of the travel 'L' which is specified at the beginning of the procedure must be covered at the conclusion of the seven different phases. Depending on the requirements, the constant velocity phase may or may not be present. So T_4 is greater than or equal to zero. By considering the jerks J_1=J_3 for phase 1 and 3, the jerks J_5=J_7 for phase 5 and 7, the times T_1=T_3=A/J_1 for phase 1 and 3, and the time T_5=T_7=D/J_5 for phases 5 and 7, the total length equation is given as,

$$L = \left(\frac{1}{2A} + \frac{1}{2D}\right)F^2 + \left(\frac{A}{2J_1} + \frac{D}{2J_5} + T_4\right)F + \left(\frac{Af_s}{2J_1} + \frac{Df_e}{2J_5} - \frac{f_s^2}{2A} - \frac{f_e^2}{2D}\right) \tag{4.76}$$

Acceleration and Deceleration terms should not be considered if those phases are not present. For the T_4 condition to hold the following equation must be satisfied.

$$T_4 = \frac{1}{F}\left[L - \left\{\left(\frac{1}{2A} + \frac{1}{2D}\right)F^2 + \left(\frac{A}{2J_1} + \frac{D}{2J_5}\right)F + \left(\frac{Af_s}{2J_1} + \frac{Df_e}{2J_5} - \frac{f_s^2}{2A} - \frac{f_e^2}{2D}\right)\right\}\right] \geq 0 \quad (4.77)$$

If equation 4.77 is not satisfied then the time of the fourth phase 'T_4' is set to zero and the maximum velocity that is achieved during the interpolation becomes equal to the maximum possible magnitude given as,

$$F = \frac{-\beta + \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha} \quad (4.78)$$

Where,

$$\alpha = \frac{1}{2A} + \frac{1}{2D}, \beta = \frac{A}{2J_1} + \frac{D}{2J_5}, \gamma = \frac{Af_s}{2J_1} + \frac{Df_e}{2J_5} - \frac{f_s^2}{2A} - \frac{f_e^2}{2D} - L \quad (4.79)$$

If equation 4.78 has complex roots, the procedure has to be restarted and the values for A, D, J, and F need to be adjusted until the condition can be satisfied.

Calculating Travel Length of Segments

The final velocity for each phase given as $f_1, f_2, \ldots, f_6$, and the travel length of each phase given as $l_1, l_2, \ldots, l_6$ can be calculated using Equation 4.7 and 4.8 and then using Equation 4.80 below the required interpolation steps during phases 1, 3, 5, and 7 can be calculated.

$$N_1 = \text{round}\left(\frac{l_1}{\Delta_s}\right), N_3 = \text{round}\left(\frac{l_3}{\Delta_s}\right)$$
$$N_5 = \text{round}\left(\frac{l_{l_l}}{\Delta_s}\right), N_7 = \text{round}\left(\frac{l_l}{\Delta_s}\right) \quad (4.80)$$

If the length of any phase given as l_i is non-zero but the corresponding N_i is zero, it must be set equal to one. Consequently, the total steps of interpolation for the combined acceleration and combined deceleration phases can be obtained using Equation 4.81.

$$N_{\text{acc}} = \text{round}\left((l_1 + l_2 + l_3)/\Delta_s\right), N_{\text{dec}} = \text{round}\left((l_5 + l_6 + l_7)/\Delta_s\right) \quad (4.81)$$

Then for phase 4, the total steps with constant velocity are obtained as given below,

$$N_4 = N - (N_{\text{acc}} + N_{\text{dec}}) \quad (4.82)$$

Now as Δs which is the displacement increment parameter is known, and the total interpolation steps during each phase are also known, the distance travelled for each phase is quantized below in Equation 4.83.

$$l_k{}' = N_k \cdot \Delta_s \quad (4.83)$$

Final Check of Acceleration and Jerk

Using the quantized travel lengths from Equation 4.83, the jerk and acceleration magnitudes must be adjusted so the desired start, constant, and end velocities can be satisfied. If a constant velocity phase is present, i.e., $T_2 > 0$, l_1, l_2, and l_3 are substituted by $l_1{}'$, $l_2{}'$, and $l_3{}'$ in Equation 4.8 and the system of equations in Equation 4.84 can be solved to obtain the new values for Acceleration (A), time of first phase (T_1) and time of third phase (T_3).

$$\begin{gathered} f_5 T_1 + \frac{1}{6} A T_1^2 - l_1' = 0 \\ -\frac{1}{8} A T_1^2 + \frac{1}{8} A T_3^2 - \frac{1}{2} f_5 T_1 - \frac{1}{2} F T_3 + \frac{F^2 - f_s^2}{2A} - l_2' = 0 \\ F T_3 - \frac{1}{6} A T_3^2 - l'_3 = 0 \end{gathered} \quad (4.84)$$

This also results in the readjustment of J_1 and J_3. If $T_2 = 0$, then the expression for l_2 from equation 4.8 is replaced the expression for f_3 from equation 4.7. In this case the system of equations then becomes,

$$\begin{gathered} f_s T_1 + \frac{1}{6} A T_1^2 - l_1' = 0 \\ \frac{1}{2} A T_1 + \frac{1}{2} A T_3 + f_s - F = 0 \end{gathered} \quad (4.85)$$

$$\frac{1}{3}AT_3^2 - \frac{1}{2}AT_1T_3 + f_sT_3 - l_3{}' = 0$$

Similarly, for the case when $T_6 > 0$, Equation 4.86 are used and l_5, l_6, and l_7 are substituted by $l_5{}'$, $l_6{}'$, and $l_7{}'$. Then, the following equation can be solved for D, T_5, and T_7.

$$\left.\begin{aligned} FT_5 - \frac{1}{6}DT_5^2 - l_5' &= 0 \\ \frac{1}{8}DT_5^2 - \frac{1}{8}DT_2^2 - \frac{1}{2}FT_5 - \frac{1}{2}f_eT_7 + \frac{F^2 - f_e^2}{2D} - l_6' &= 0 \\ f_eT_7 + \frac{1}{6}DT_7^2 - l_7' &= 0 \end{aligned}\right\} \quad (4.86)$$

This also results in the readjustment of J_5 and J_7. If $T_6 = 0$, then the expression for l_6 from equation 4.8 is replaced the expression for f_7 from equation 4.7. In this case the system of equations then becomes,

$$\left.\begin{aligned} FT_5 - \frac{1}{6}DT_5^2 - l_5' &= 0 \\ \frac{1}{2}DT_5 + \frac{1}{2}DT_7 + f_e - F &= 0 \\ -\frac{1}{3}DT_7^2 - \frac{1}{2}DT_5T_7 + FT_7 - l_7' &= 0 \end{aligned}\right\} \quad (4.87)$$

These equations can be solved iteratively using numerical methods. Once the values for the different variables converge, the values for the jerks are updated, and the final velocities for each phase are also recalculated using Equation 4.7.

Calculation of Time Steps

The final step in the trajectory generation is to calculate the time steps between each interpolation step for each of the seven phases. The displacement as a function of the time parameter for the k^{th} phase is given below in Equation 4.88.

$$s(\tau_k) = \frac{1}{6}j_{0k}\tau_k^3 + \frac{1}{2}a_{0k}\tau_k^2 + f_{0k}\tau_k + s_{0k}, 0 \le \tau_k \le T_k \quad (4.88)$$

where j_{0k} is the initial jerk of the phase, a_{0k} is the initial acceleration of the phase, f_{0k} is the initial velocity of the phase, and s_{0k} is the initial displacement value of the phase which were calculated during the previous section. Here τ_k is the time which is zero at the start of each phase. As the displacement () is constant for each step then Equation 4.88 becomes

$$s_{kn}(\tau_k) = n.\Delta s = \frac{1}{6} j_{0k}\tau_k^3 + \frac{1}{2} a_{0k}\tau_k^2 + f_{0k}\tau_k + s_{0k} \tag{4.89}$$

Here n is the interpolation step which goes up to N_i. The interpolation period for each point is obtained by solving the above equation 4.89 (using Newton-Raphson iterative algorithm) and then given as,

$$T_{kn}^i = \tau_{kn} - \tau_{k,n-1} \tag{4.90}$$

4.4.3. Reconstructing Trajectory at Desired Control Loop Frequency

A trajectory is obtained in the last section where the displacement is constant but the time steps are varying. However, it is desired to supply a signal to the control loop at a fixed frequency. So, the trajectory is reconstructed at this desired frequency to ensure a smooth acceleration and velocity profile. This is obtained by using a fifth order polynomial with time as the variable instead of the chord lengths. The first derivative and second derivative are derived using the same method described in 4.4.1. Then the equation for the polynomial between any two points is given as,

$$\tilde{x}(\tau) = A\tau^5 + B\tau^4 + C\tau^3 + D\tau^2 + E\tau + F \tag{4.91}$$

This is solved similar to the approach in section 4.4.1 using the initial conditions,

$$\begin{array}{cc} \tilde{x}(0) = x_i & \tilde{x}\left(T_{i+1}^i\right) = x_{i+1} \\ d\tilde{x}(0)/d\tau = \hat{x}_i & d\tilde{x}\left(T_{i+1}^i\right)/d\tau = \widehat{x_{i+1}} \\ d^2\tilde{x}(0)/d\tau^2 = \hat{\ddot{x}}_i & d^2\tilde{x}\left(T_{i+1}^i\right)/d\tau^2 = \widehat{\ddot{x}_{i+1}} \end{array} \tag{4.92}$$

where the time interval T_{i+1}^i corresponds to the interpolation step associated with the reference point x_{i+1} (i.e. $T_{i+1}^i = t_{i+1} - t_i$) and $0 \le \tau \le T_{i+1}^i$.

The polynomial coefficients must be recalculated each time a new reference point is generated by the algorithm. This process is done recursively and ultimately the final trajectory is obtained at the desired control loop time period.

5. CONTROLLER DESIGN

The controller chosen to be used with the cable robot is a cascade controller. Cascade controllers are a popular choice when electric motors are concerned. The general structure of a cascade controller for an electric motor is given below in Figure 5.1.

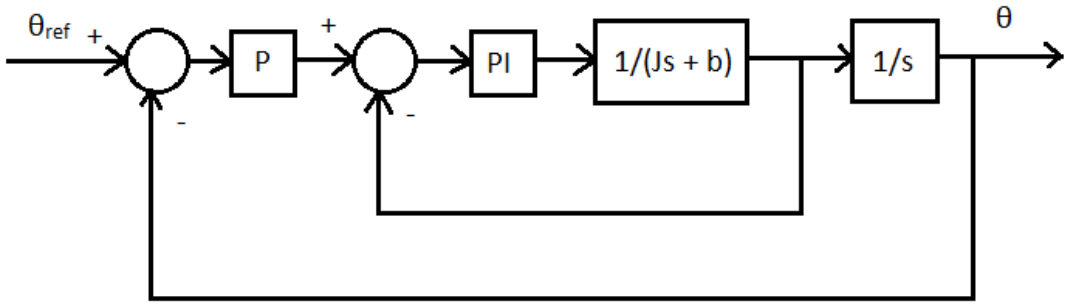

Figure 5.1. Cascade Control of Motor

Here the inner speed sloop is controlled using a PI controller, and the outer position loop is controlled using a P controller. The transfer function $\frac{1}{Js+b}$ represents the motor dynamics, where J is the motor inertia, and b is the damping term. The second transfer function $\frac{1}{s}$ is an integrator to obtain the position command.

The open loop transfer function for the inner loop is given as,

$$OLTF_{inner} = \left(K_P + \frac{K_I}{s}\right)\left(\frac{1}{Js+b}\right) \tag{5.1}$$

$$OLTF_{inner} = \frac{K_P s + K_I}{Js^2 + bs} \tag{5.2}$$

The closed loop transfer function is then given as,

$$CLTF_{inner} = \frac{\frac{K_P s + K_I}{Js^2 + bs}}{1 + \frac{K_P s + K_I}{Js^2 + bs}} \tag{5.3}$$

$$CLTF_{inner} = \frac{(K_P s + K_I)\frac{1}{J}}{s^2 + \left(\frac{K_P + b}{J}\right)s + \frac{K_I}{J}} \tag{5.4}$$

Then the denominator for the closed loop transfer function of the inner loop are given as,

$$s^2 + \left(\frac{K_P + b}{J}\right)s + \frac{K_I}{J} = (s + P_1)(s + P_2) \tag{5.5}$$

Here P_1 and P_2 are the poles of $CLTF_{inner}$. Pole placement can be used to determine the values of K_p and K_I given as,

$$\begin{gathered} K_P = (P_1 - P_2).J - b \\ K_I = P_1 P_2.J \end{gathered} \tag{5.6}$$

If the desired settling time and Maximum percentage overshoot are know, then the the natural frequency and damping are given as,

$$\begin{gathered} \omega_n = \frac{4}{\zeta * T_s} \\ \zeta = \sqrt{\frac{ln^2(MO/100)}{\pi^2 + ln^2(MO/100)}} \end{gathered} \tag{5.7}$$

Then the poles are given as,

$$s_{1,2} = -\zeta\omega_n \pm i\omega_n\sqrt{1 - \zeta^2} \tag{5.8}$$

Then the gains can be obtained from the following equation,

$$s^2 + \left(\frac{K_P + b}{J}\right)s + \frac{K_I}{J} = s^2 + 2\zeta\omega_n s + \omega_n^2 \tag{5.9}$$

Where,

$$\begin{gathered} K_p = 2J\zeta\omega_n - b \\ K_I = \omega_n^2 * J \end{gathered} \tag{5.10}$$

Now the open loop transfer function for the outer loop is given as,

$$OLTF_{outer} = K_{Po}.\frac{(K_P s + K_I)\frac{1}{J}}{s^2 + \left(\frac{K_P + b}{J}\right)s + \frac{K_I}{J}}.\frac{1}{s} \tag{5.11}$$

$$OLTF_{outer} = \frac{K_{Po}(K_P s + K_I)\frac{1}{J}}{s^3 + \left(\frac{K_P + b}{J}\right)s^2 + \frac{K_I}{J}s} \tag{5.12}$$

Then the transfer function of the outer loop is given as,

$$CLTF_{outer} = \frac{K_{Po}.\frac{(K_P s + K_I)\frac{1}{J}}{s^2 + \left(\frac{K_P + b}{J}\right)s + \frac{K_I}{J}}.\frac{1}{s}}{1 + K_{Po}.\frac{(K_P s + K_I)\frac{1}{J}}{s^2 + \left(\frac{K_P + b}{J}\right)s + \frac{K_I}{J}}.\frac{1}{s}} \tag{5.13}$$

It can also be written as,

$$CLTF_{outer} = \frac{K_{Po}.\frac{(K_P s + K_I)\frac{1}{J}}{(s + P_1)(s + P_2)}.\frac{1}{s}}{1 + K_{Po}.\frac{(K_P s + K_I)\frac{1}{J}}{(s + P_1)(s + P_2)}.\frac{1}{s}} \tag{5.14}$$

$$CLTF_{outer} = \frac{K_{Po}.(K_P s + K_I)\frac{1}{J}.\frac{1}{s}}{(s + P_3)(s + P_1{}')(s + P_2{}')} \tag{5.15}$$

Here,

$$\frac{(K_P s + K_I)\frac{1}{J}.\frac{1}{s}}{(s + P_1{}')(s + P_2{}')} \approx 1 \tag{5.16}$$

So,

$$CLTF_{outer} = \frac{K_{Po}}{(s + P_3)} = \frac{K_{Po}}{(s + K_{Po})} = \frac{1}{\frac{1}{K_{Po}}s + 1} \tag{5.17}$$

Then, to determine K_{Po}, a settling time T_s is given, where

$$T_s = 4\tau \text{ and } \tau = \frac{1}{K_{Po}} \tag{5.18}$$

$$T_s = \frac{4}{K_{Po}} \tag{5.19}$$

Then,

$$K_{Po} = \frac{4}{T_s} \tag{5.20}$$

Here K_{Po} is the gain of the outer loop controller, and it must be 10 times smaller than ω_n of the inner loop. The settling-time of the close outer loop must also be 10 times slower than the settling time of the inner loop. The actual diagram of the controller with the plant is shown below in Figure 5.2.

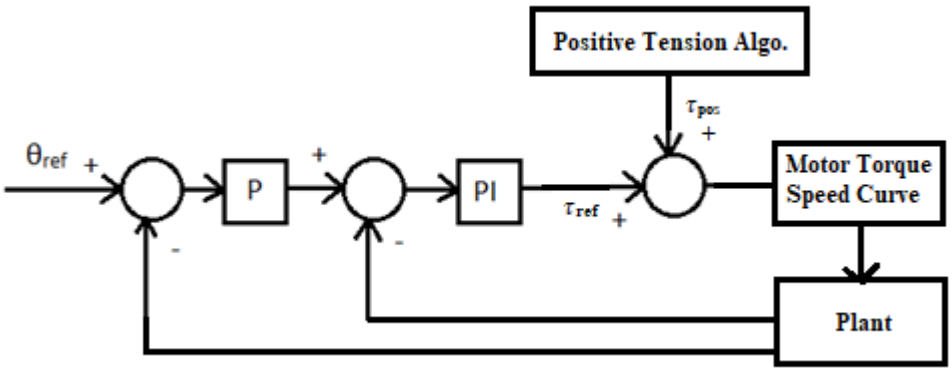

Figure 5.2. Final Controller Diagram

6. CABLE ROBOT OPTIMIZATION

There are a number of parameters that can be chosen for cable robot optimization. These parameters can be then used to develop a cost function. The objective is to usually maximize or minimize that cost function to obtain the desirable characteristics for the cable robot and increasing performance. Table 2.1 in section 2.2 detailed different sources in the literature dealing with cable robot optimization and the associated cost functions. Cable Robot Dexterity, Stiffness, and Workspace were the most common themes for optimization and were chosen for optimization. The variables used during the optimization process are the motor positions and end-effector cable connection points. The different cost functions developed are described below.

6.1. Dexterity Optimization [28]

The local kinematic behavior of the system is described by the dexterity index. The dexterity index is given as,

$$k(J_h) = \frac{\lambda_{max}}{\lambda_{min}} \tag{6.1a}$$

Where J represents the Jacobian matrix in a specific configuration, and λ_{max} is the largest eigenvalue of J, and λ_{min} is the smallest singular value of J. The dexterity index can vary between 1 and infinity, so normally $k(J)^{-1}$ is considered which varies between 0 and 1. Here the calculation of the GCI is done using the homogenous Jacobian matrice given below, where c is the length of the edge of the end-effector.

$$J_h = J.diag(1,1,\frac{2}{c\sqrt{2}}) \tag{6.1b}$$

When the index is near 1, the system is far from a singularity. Then a term called the Global Conditioning Index can be defines as,

$$GCI = \frac{1}{n}\sum_{i=1}^{n} k(J_h)_i{}^{-1} \tag{6.2}$$

Where n is the number of points. For a given configuration of the cable robot, the dexterity index can be evaluated for each point in the workspace and then the GCI is calculated. When the GCI is closer to 1, the system has better dexterity for the given configuration. If the end-effector connection point is fixed, Figure 6.1 below shows the value of GCI as a function of the x and y motor positions.

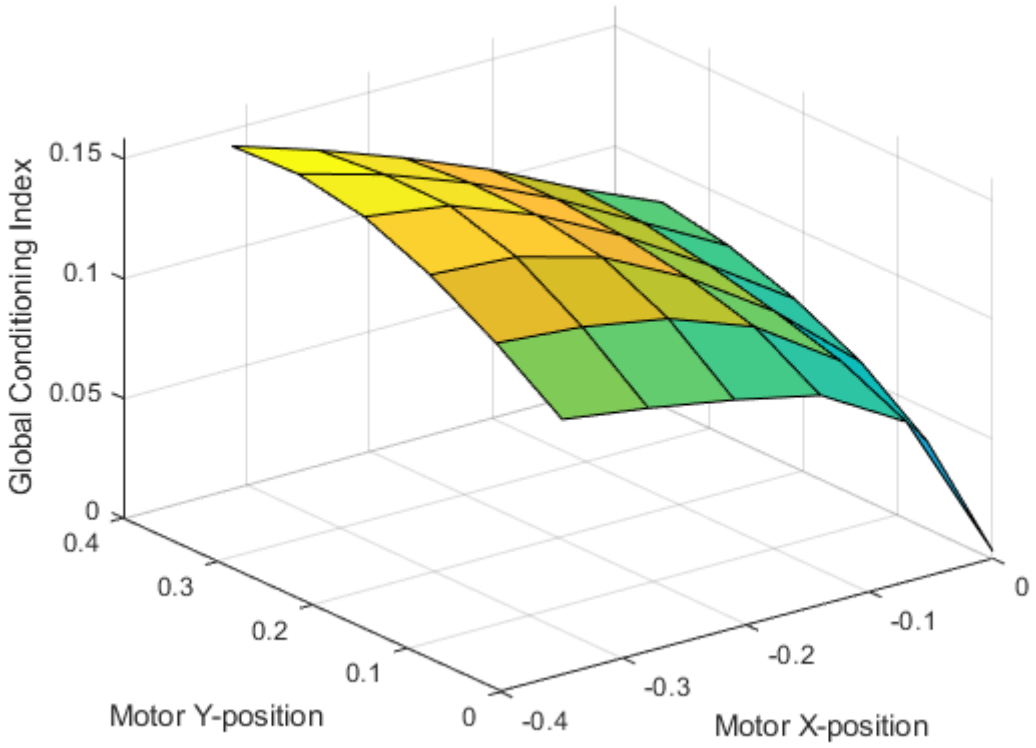

Figure 6.1. GCI as a function of motor position

6.2. Stiffness Optimization [37]

[37] defines the stiffness matrix for a planar CDPR. The stiffness matrix consists of two parts as shown below,

$$\mathbf{K}_e = \sum_{i=1}^{n} k_{\mathbf{i}} \begin{bmatrix} \hat{l}_i^T & \hat{l}_i \hat{l}_i^T [\hat{b}_i \times]^T \\ [\hat{b}_i \times]^T \hat{l}_i \hat{l}_i^T & [\hat{b}_i \times]^T \hat{l}_i \hat{l}_i^T [\hat{b}_i \times]^T \end{bmatrix} \tag{6.3}$$

$$\mathbf{K}_p = \sum_{i=1}^{n} \frac{\tau_\mathbf{i}}{l_i} \begin{bmatrix} 1 - \hat{l}_i \hat{l}_i^T & [\hat{b}_i \times]^T - \hat{l}_i \hat{l}_i^T [\hat{b}_i \times]^T \\ [\hat{b}_i \times] - [\hat{b}_i \times]^T \hat{l}_i \hat{l}_i^T & [\hat{b}_i \times][\hat{b}_i \times]^T - [\hat{b}_i \times]^T \hat{l}_i \hat{l}_i^T [\hat{b}_i \times]^T \end{bmatrix} - \sum_{i=1}^{n} \tau_\mathbf{i} \begin{bmatrix} 0 & 0 \\ 0 & [\hat{l}_i \times][\hat{b}_i \times] \end{bmatrix}$$

Here K_e is the stiffness coming from the elasticity of the cables, and K_p is the stiffness resulting from the orientation of the cable robot.

The directional stiffnesses of the cable-robot in the directions of the eigenvectors is determined by the corresponding eigenvalues of the stiffness matrix. Therefore, the end-effector could have different stiffnesses in different directions. But for most applications the stiffness distribution is desired to be uniform in all directions. The stiffness number represents the uniformity of the stiffness matrix. It is the ratio of the smallest and largest eigenvalues of the stiffness matrix (K) which is given as,

$$SN = \frac{\lambda_{min}(K)}{\lambda_{max}(K)} \tag{6.4}$$

The stiffness number varies between 0 and 1. When it is equal to 1 the system is isotropic, i.e., the stiffness is uniform in each direction. However, the stiffness matrix has non-homogenous units, so the following expression can be written for the stiffness matrix,

$$\mathbf{F}_o = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{\delta} \\ \boldsymbol{\omega} \end{bmatrix} \tag{6.5}$$

Where F_o is the external wrench given as,

$$\boldsymbol{F}_o = [\boldsymbol{S}_i, \boldsymbol{M}_i]^T \tag{6.6}$$

And δ and ω are the infinitesimal translation and rotations of the end-effector.

Then the matrices with homogeneous units are given as,

$$\begin{aligned} \mathbf{U}_\boldsymbol{f} &= [K_{11} O_{K_{11}} \quad K_{12} O_{K_{12}}] \\ \mathbf{U}_\boldsymbol{m} &= [K_{21} O_{K_{21}} \quad K_{22} O_{K_{22}}] \end{aligned} \tag{6.7}$$

Here, U_f is for force and U_m is for moment. $O_{K_{11}}$, $O_{K_{12}}$, $O_{K_{21}}$, and $O_{K_{22}}$ are orthogonal matrices which can be obtained by using the eigenvectors of the matrices $K_{11}^T K_{11}$, $K_{12}^T K_{12}$, $K_{21}^T K_{21}$, and $K_{22}^T K_{22}$ respectively. Then using the matrices $U_f U_f^T$ and $U_m U_m^T$ with homogeneous units, the eigenvalues can be obtained and the stiffness numbers can be calculated for translation and rotational motion.

The stiffness number however has a different magnitude when the end-effector has different positions. To obtain the stiffness-number index over the entire workspace defined by the stiffness feasibility condition, it is given as,

$$ASN = \frac{\sum_{i=1}^{n} SN_i}{n} \tag{6.8}$$

Where n is the total number of feasible points in the workspace. To check the stiffness feasible workspace, the eigenvalues of the stiffness matrix are checked for each pose. If the smallest eigenvalue is positive, that pose is a part of the stiffness feasible workspace which is a subset of the wrench-feasible workspace. If only linear stiffnesses are optimized however, performance can be lost in the rotational direction and the cable robot can become unstable so the average of the Stiffness Number (SN) of the overall stiffness matrix is chosen as the cost function.

If the end-effector connection point is fixed, Figure 6.2 below shows the value of ASN as a function of the x and y motor positions when the end-effector connection point is fixed.

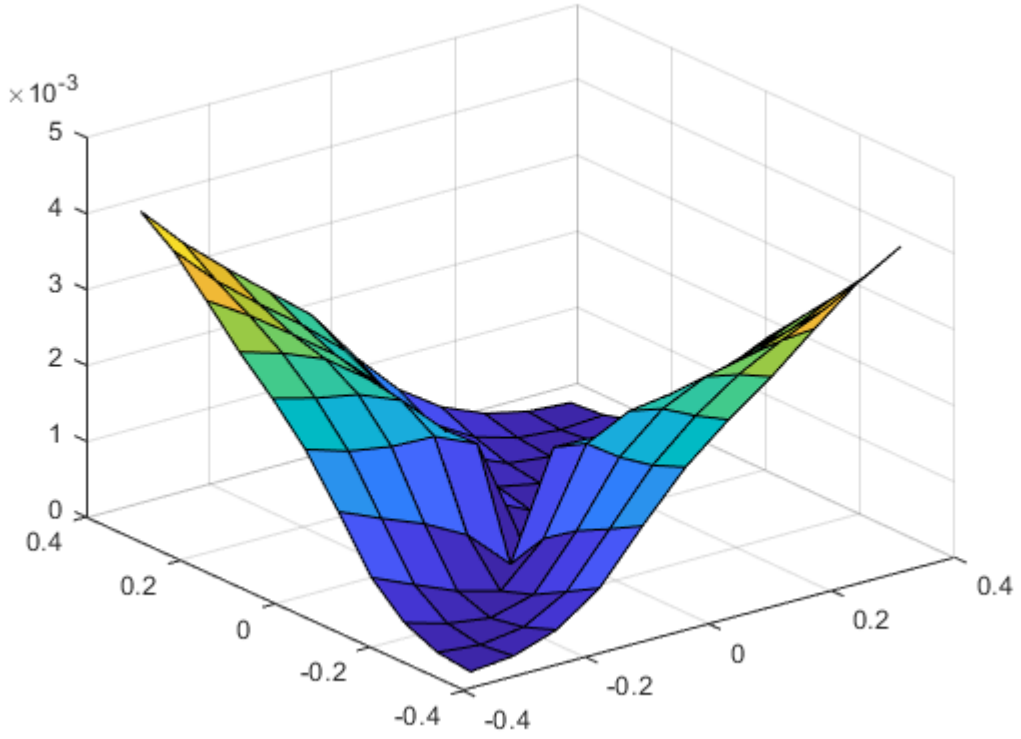

Figure 6.2. SN as a function of motor positions

6.3. Workspace Optimization [38]

Obtaining the maximum feasible workspace volume is another important optimization objective as this allows the cable robot to utilize the maximum amount of area covered by the motors. The workspace volume index is given as,

$$WVI = \frac{n_{feas}}{n_{total}} \tag{6.9}$$

Here n_{feas} are the feasible points in the workspace, and n_{total} are the total points in the workspace. The steps to check the feasibility of the points are:

1. Check the cable length condition given as,

$$l_{i_{min}} \leq l_i \leq l_{i_{max}}, \forall\, i = 1,2 \ldots, m \tag{6.10}$$

2. Check force-closure condition given as,

$$\begin{aligned} &\text{rank } \boldsymbol{J} = n, \text{ if } \boldsymbol{J} \in \mathbb{R}^{m \times n} \text{ where } m > n \\ &\forall \boldsymbol{N} \in \text{null } (\boldsymbol{A}), \exists \boldsymbol{Nh} \in \mathbb{R}_{+}^{m}, \text{ where } m > n \end{aligned} \tag{6.11}$$

Here J represents the Jacobian matrix, and A represents the structure-matrix, while N represents the null-space of the structure matrix A.

3. Calculate the stiffness matrix given in Equation 6.3 and check if it is positive definite.

4. Check the feasible wrench condition given below, and also calculate the cable tensions and check the cable tension condition.

$$\begin{aligned} &\exists\{\tau \mid \tau = -\boldsymbol{A}^{\dagger}\boldsymbol{W} + \boldsymbol{Nh}, \boldsymbol{Nh} \in \mathbb{R}_{+}^{m}, \text{ where } n < m\} \cap \\ &\quad \left\{\tau \mid 0 < \tau_{i,\min} \leq \tau_i \leq \tau_{i,\max} \forall i = 1,2,\dots,m\right\} \end{aligned} \tag{6.12}$$

Where τ is the cable tension, and τ_{min} and τ_{max} are the minimum and maximum tension values. If all the above conditions are satisfied the point is added to the feasible points n_{feas}.

If the end-effector connection point is fixed, Figure 6.3 below shows the value of WVI as a function of the x and y motor positions.

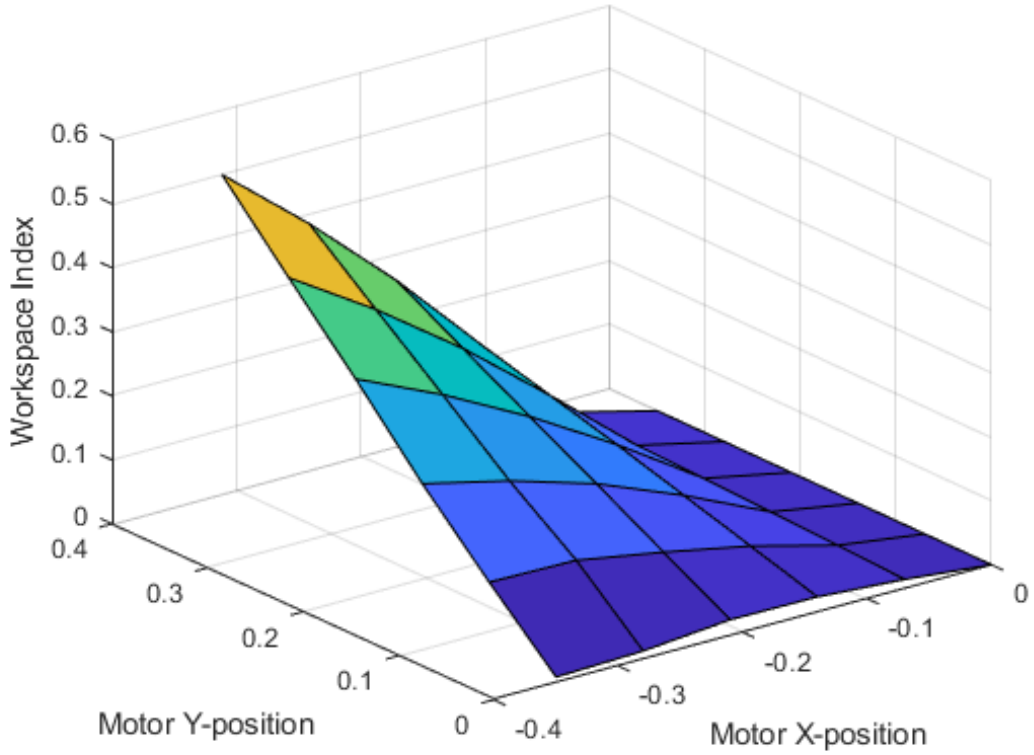

Figure 6.3. WVI as a function of motor positions

The feasible workspace at the initial condition of workspace optimization is given as,

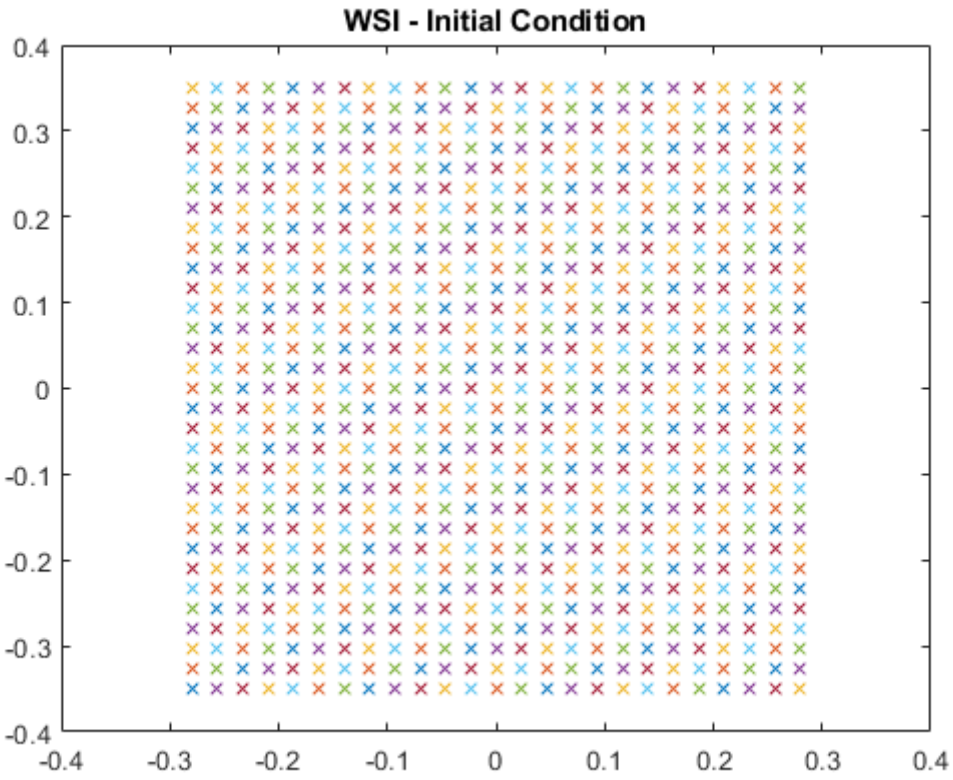

Figure 6.4. Feasible Workspace before optimization

And the workspace after optimization is given as,

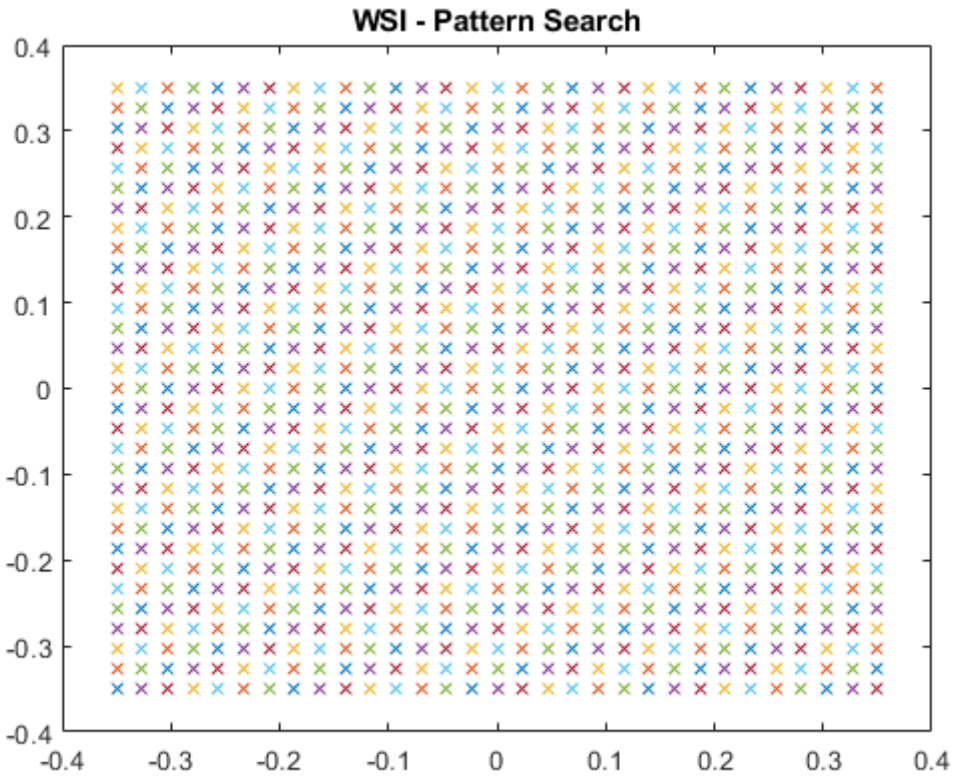

Figure 6.5 Feasible Workspace after optimization

6.4. Optimization Algorithms

There are three main approaches used for optimization these are given as,

6.4.1. MATLAB fmincon algorithm

This function is used to find the minimum of a constrained nonlinear multivariable function. fmincon has the following algorithms it can use,

- 'interior-point'
- 'trust-region-reflective'
- ‘sqp'
- ‘sqp-legacy'
- 'active-set'

Here ‘interior-point’ is the default algorithm. The inputs of the fmincon algorithm include:

1. fun – This is the cost function to be minimized
2. x0 – This is the initial condition
3. A, b – These are the matrices that specify linear inequality constraints
4. A_{eq}, b_{eq} – These are the matrices that specify the linear equality constraints
5. lb, ub – These define the lower and upper bounds

The function is then written as,

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases} \quad (6.13)$$

Or,

$$x = fmincon(fun, x0, A, b, Aeq, beq, lb, ub) \quad (6.14)$$

6.4.2. MATLAB patternseach algorithm

This function is also used to find the minimum of a function using the patternsearch algorithms. patternsearch has the following algorithms it can use,

- "classic"
- "nups" (Nonuniform Pattern Search)
- "nups-gps"
- "nups-mads"

The inputs of the patternsearch algorithm include:

1. fun – This is the cost function to be minimized
2. x0 – This is the initial condition
3. A, b – These are the matrices that specify linear inequality constraints
4. A_{eq}, b_{eq} – These are the matrices that specify the linear equality constraints
5. lb, ub – These define the lower and upper bounds

The function is then written as,

$$x = patternsearch(fun, x0, A, b, Aeq, beq, lb, ub) \quad (6.15)$$

6.4.3. MATLAB fgoalattain algorithm

fgoalattain is used to solve multi-objective goal attainment problems (to either minimize or maximize cost functions). The function is given as,

$$\underset{x,\gamma}{\text{minimize } \gamma \text{ such that }} \begin{cases} F(x) - weight.\gamma & \leq goal \\ c(x) & \leq 0 \\ ceq(x) & = 0 \\ A \cdot x & \leq b \\ Aeq \cdot x & = beq \\ lb & \leq x \leq ub \end{cases} \tag{6.16}$$

Or,

$$x = fgoalattain(fun, x0, goal, weight, A, b, Aeq, beq, lb, ub) \tag{6.17}$$

Here the inputs are,

1. fun – This is the cost function to be minimized
2. x0 – This is the initial condition
3. goal – This is the specified goal value for the cost function
4. weight – This specifies the weights of different functions in the multi-objective optimization problem.
5. A, b – These are the matrices that specify linear inequality constraints
6. A_{eq}, b_{eq} – These are the matrices that specify the linear equality constraints
7. lb, ub – These define the lower and upper bounds

6.5. Combined Optimization

The combined optimization problem considers the cable robot dexterity and stiffness cost functions. The fgoalattain command is used for combined optimization. The workspace cost function is not compatible with the fmincon or fgoalattain algorithms as it is not a mathematical formulation as compared to the other cost functions. Both dexterity and stiffness cost functions are given the same weight, 1, and the goal are set as 1 for each cost function as they are divided by their maximum values obtained in the individual optimizations. A high dexterity and isotropic stiffness are both desirable for the cable robot.

7. SIMULATIONS AND RESULTS

This chapter provides the results for the simulations that are carried out for different scenarios. The simulations are carried out for different trajectories. Initially the cable robot is given some parameters, the controller design is done, the optimization is done, followed by trajectory generation, and ultimately simulation. Separate simulations are also carried out for custom trajectory data.

The cable robot design procedure has the following steps:

1. Setting the Cable Robot Parameters
2. Designing the controller by pole placement
3. Structural Optimization of the Cable Robot
4. Trajectory Generation
5. Running Simulation

7.1. Cable Robot Parameters

Table 7.1 below shows the Cable Robot Parameters that are used for simulation purposes.

Table 7.1. Cable Robot Parameters

No	**Parameter Name**	**Value**
1	Motor 1 Position in Global Coordinate System	[-0.35m, 0.35m]
2	End-Effector Cable connection 1 position	[-0.05m, -0.05m]
3	End-Effector mass	0.91 kg
4	End-Effector Inertia	1.5×10^{-3} kg.m^2
5	Cable Young's Modulus	200 GPa
6	Cable cross sectional area	1×10^{-6} m^2
7	Motor Inertia	0.0026 kg.m^2
8	Motor Damping	0.2 Ns/m
9	Motor Coulomb Friction	0.005 Nm
10	Motor Radius	0.0381 m
11	Minimum Cable Tension	0.5 N

7.2. Controller Design

The cascade controller is designed using the pole placement method described in Section 5. The settling time for the inner loop is set as 0.02s and the maximum overshoot is chosen as 10%. The settling time is specified as 0.2s for the outer loop. This yields the following gains,

Table 7.2. Controller gains

Gain	**Value**
P_o	20
P_I	0.8400
I_I	297.5984

The step response of the inner loop if given as,

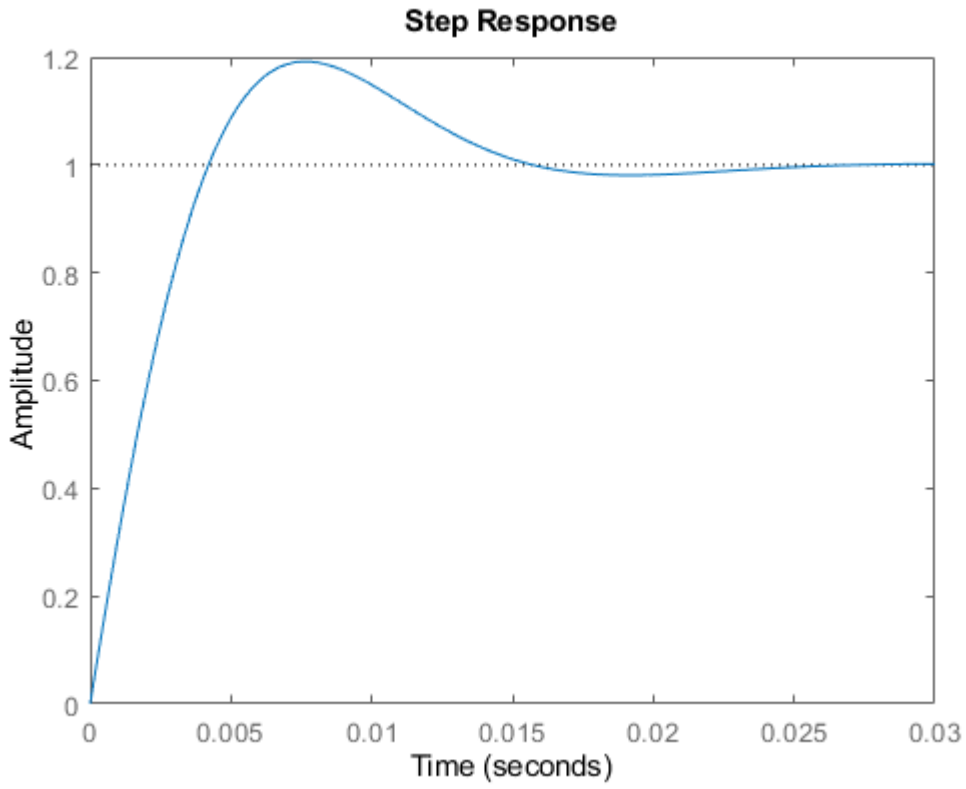

Figure 7.1 Step response of cascade inner loop.

And the step response of the outer loop is given as,

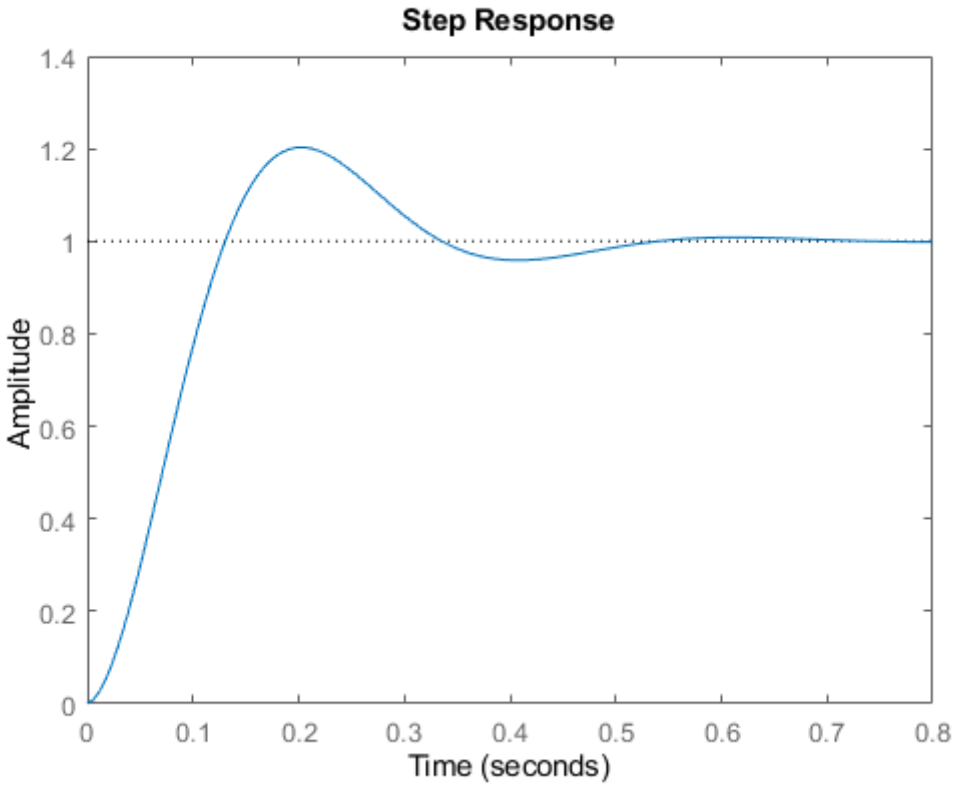

Figure 7.2 Cascade outer loop step response

7.3. Optimization

The motor and end-effector cable attachment points are the variables that are modified during the optimization procedure. Changing these parameters can significantly impact the performance of the cable robot. Figure 7.3 below shows the configuration of the Cable Robot in the initial state.

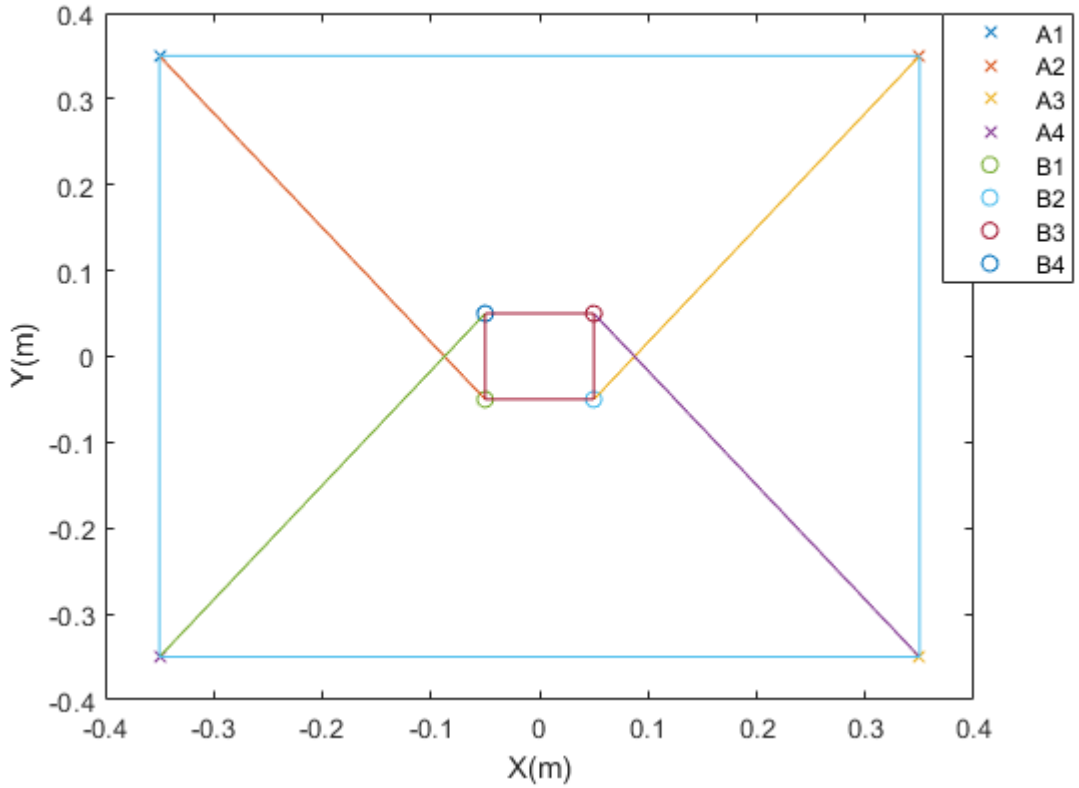

Figure 7.3 Initial condition of cable robot

Table 7.3 below shows the initial values of the three different cost functions before optimization, and the value after the individual and combined optimizations.

Table 7.3. Values of cost functions for each optimization

Cost Function	**Initial Condition**	**Dexterity Optimization**	**Stiffness Optimization**	**Workspace Optimization**	**Combined Optimization**
Dexterity	0.1575	0.1587	0.1575	0.0571	0.1575
Stiffness	0.0041	0.0038	0.0041	5.4722e-04	0.0041
Workspace Index	0.8431	0.8101	0.8431	1.0000	0.8431

First the dexterity optimization is carried out. Figure 7.4 shows the new configuration of the cable robot.

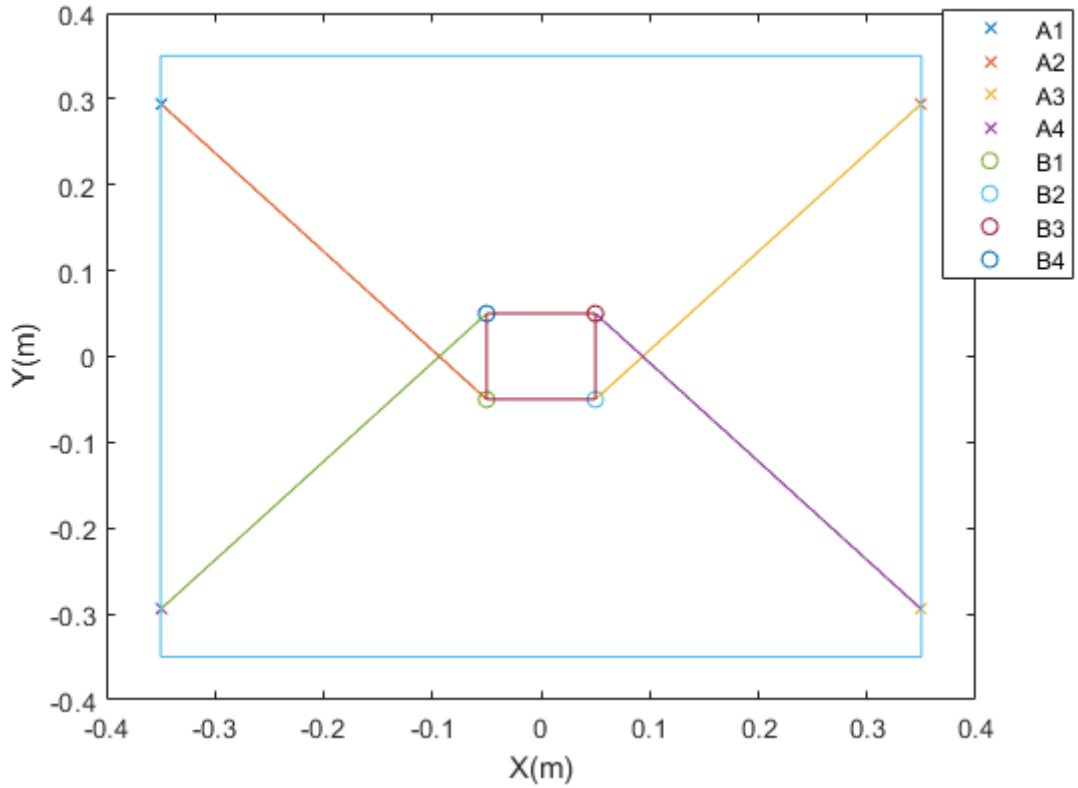

Figure 7.4 Dexterity optimized cable robot configuration

Then the stiffness optimization is carried out. Figure 7.5 shows the new configuration of the cable robot. It is the same as the initial configuration. It converges to this arrangement even with different initial conditions.

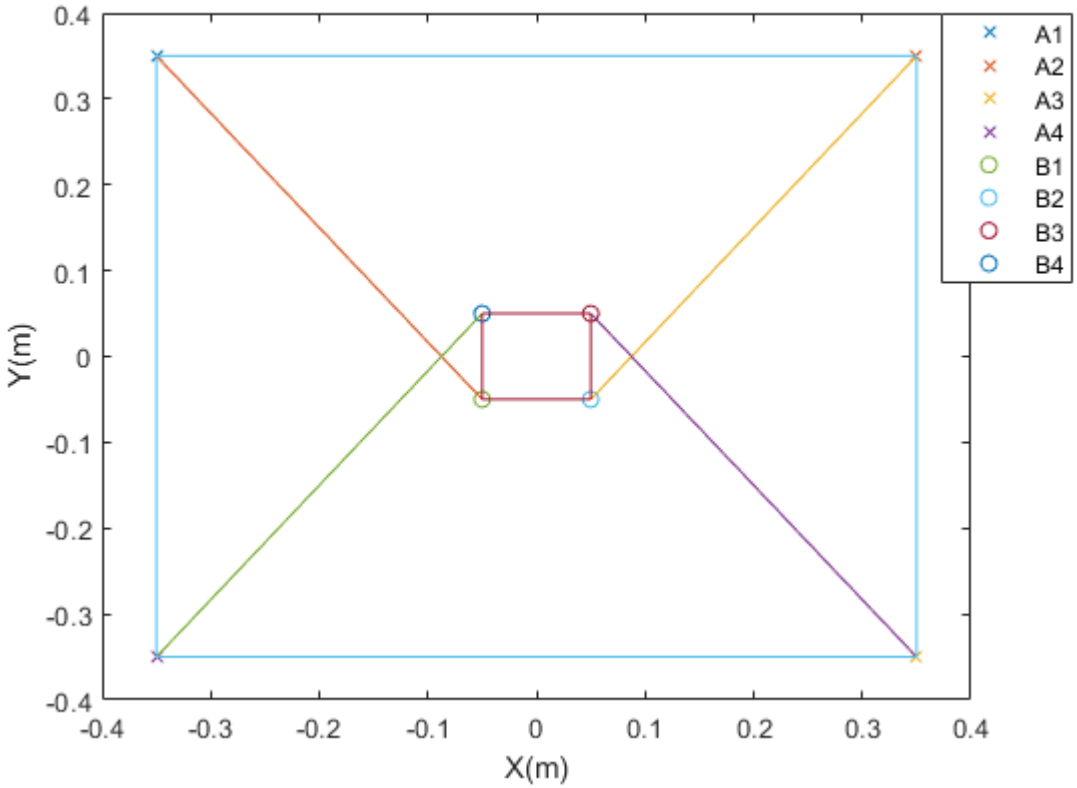

Figure 7.5 Stiffness optimized cable robot configuration

Lastly, the workspace optimization is carried out. Figure 7.6 shows the new configuration of the cable robot, as well as the old and new workspace indicated on the model in Figures 7.7 and 7.8.

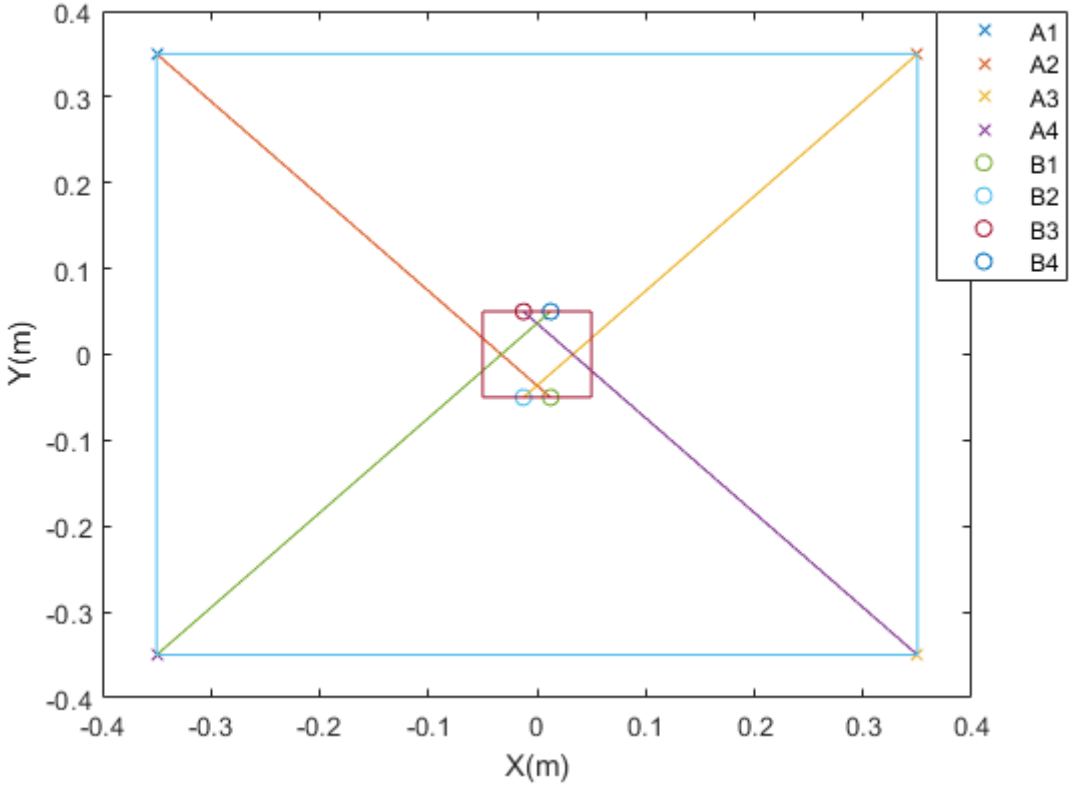

Figure 7.6 Workspace optimized cable robot structure

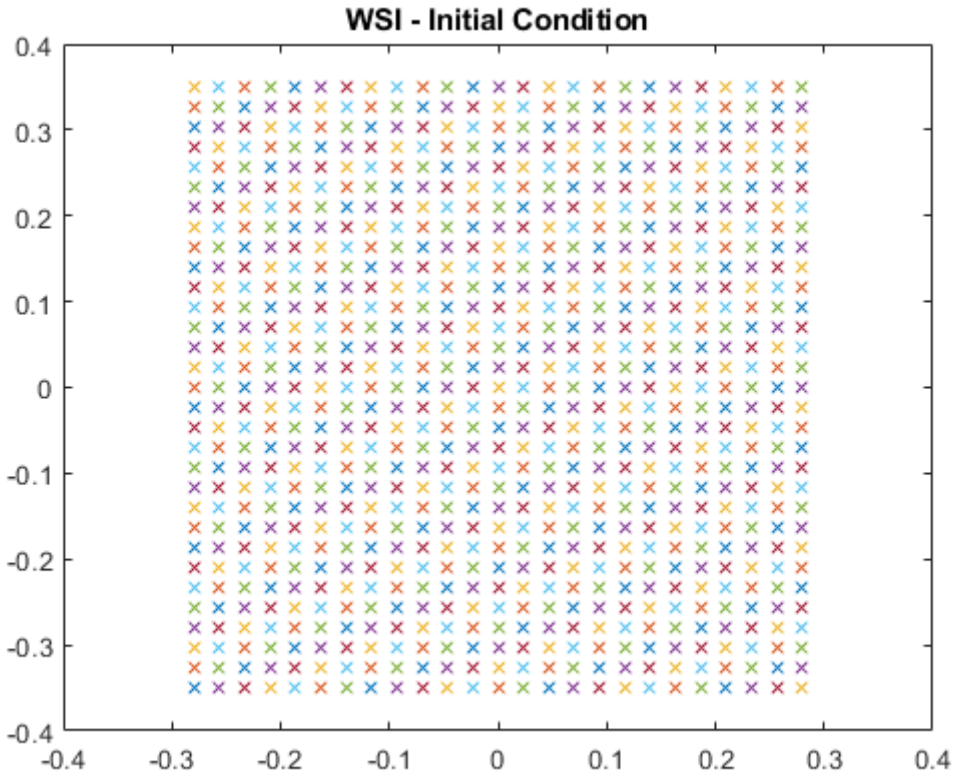

Figure 7.7 Feasible region of workspace before optimization

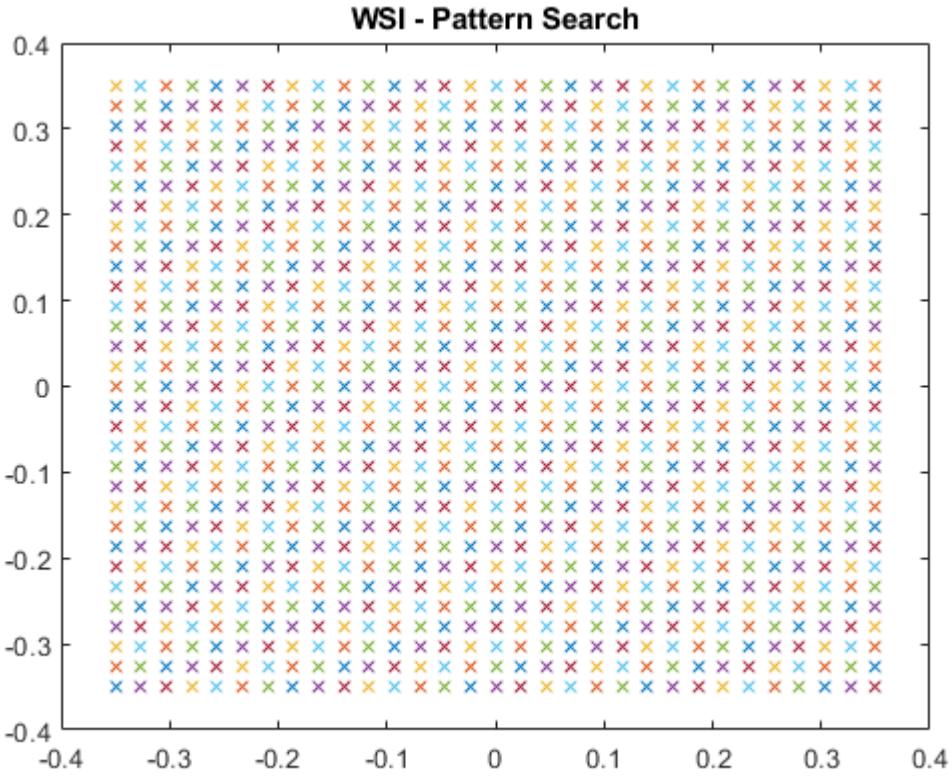

Figure 7.8 Feasible region of workspace after optimization

The last procedure is the combined optimization problem which is run using the fgoalattain command. The fgoalattain command uses the same algorithms as the fmincon command, however the workspace cost function is not compatible with these algorithms. Based on the values obtained in Table 7.3 above, the combined optimization yields the best result at the initial configuration shown in Figure 7.3. where the motor and cable connections are at the edges of the base and end-effector respectively. Both the dexterity and stiffness cost functions are near their maximum values observed in the individual optimizations.

7.4. Trajectory Generation

For the first set of simulation, two types of standard trajectories are generated. These are a Circle and Rectangle trajectory. These are generated using the Trajectory generation

algorithm 1. The different trajectory generation approaches are compared. Figures 7.9 and 7.10 below show the circular and rectangular trajectories.

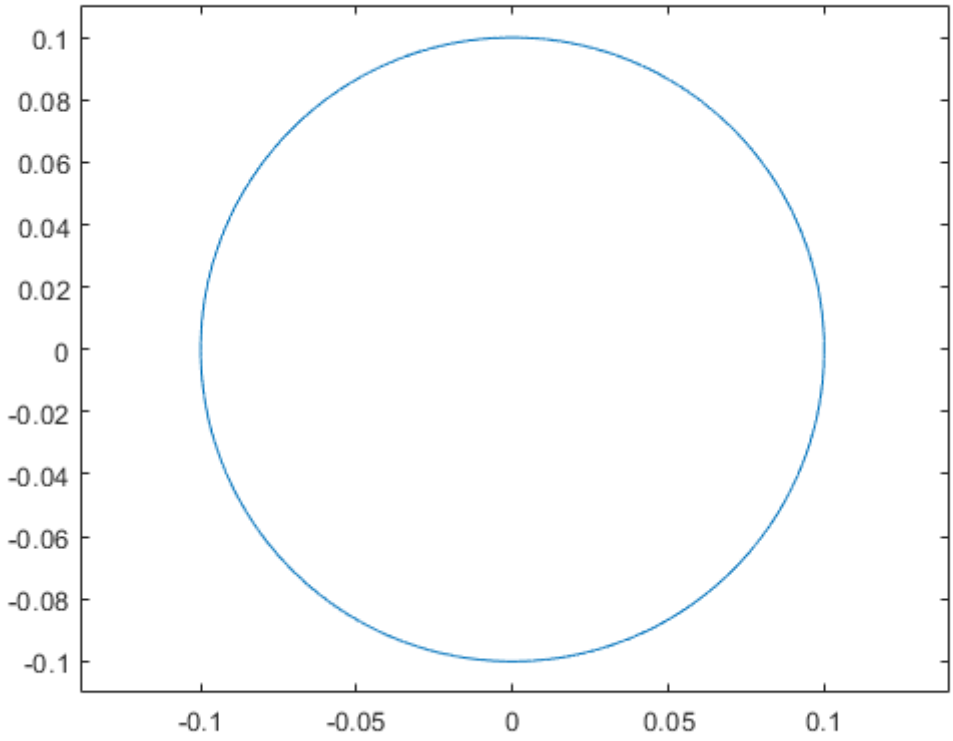

Figure 7.9 Circular Trajectory

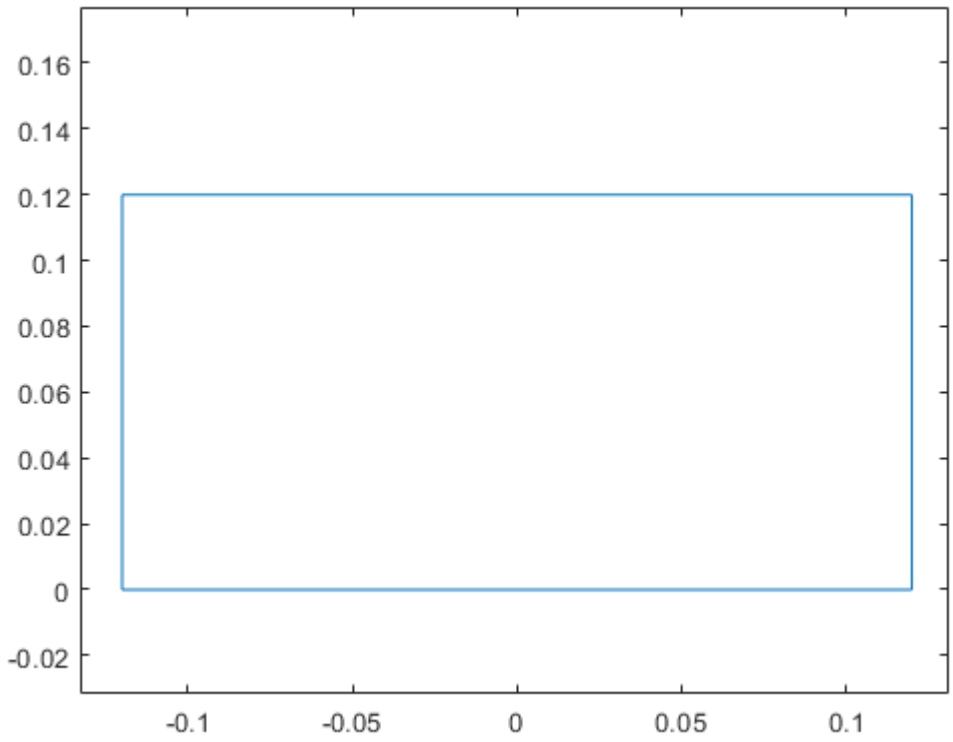

Figure 7.10 Rectangle Trajectory

The second set of simulations uses Trajectory generation algorithm 3 to generate a non-standard shape which is a spiral. A spiral has a constantly changing arc radius which is not possible to generate using algorithm 1. In the third set of simulations, Trajectory generation algorithm 2 is used to generate the trajectories for different sets of random data.

7.5. Simulation

This section includes the simulation results for different scenarios described below.

7.5.1. Trajectory Generation 1 – Circle

The circular trajectory is a basic trajectory that can be generated easily using a single arc segment. Figure 7.11 below shows the Trapezoidal Acceleration Profile for the circular trajectory.

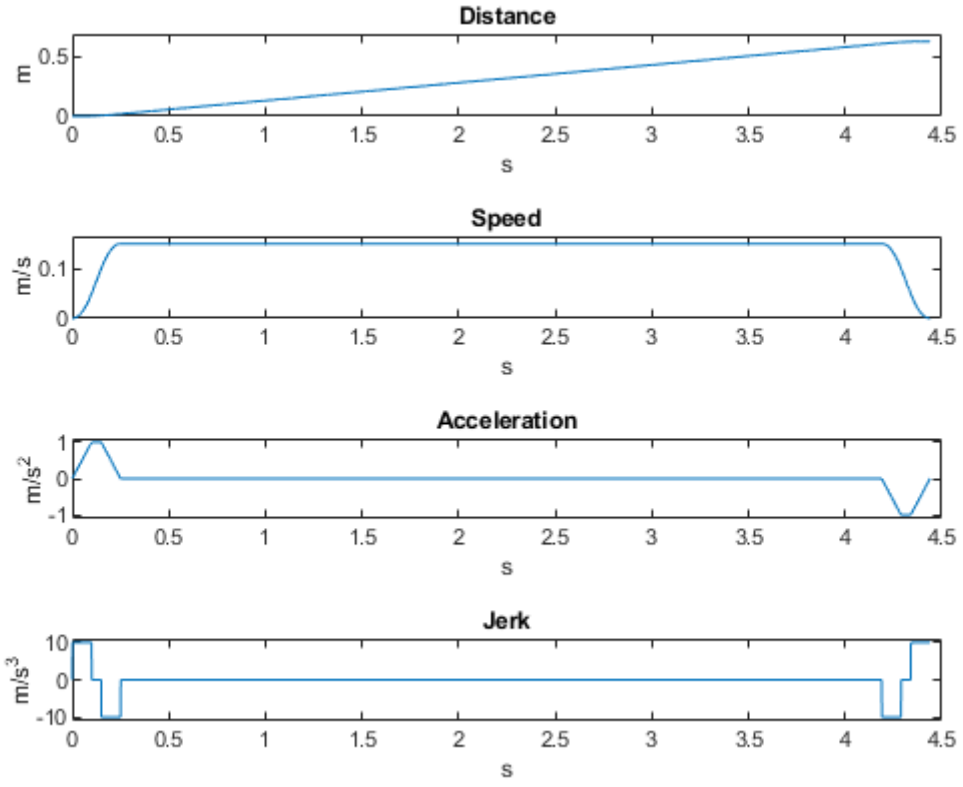

Figure 7.11 Circle – Trapezoidal Acceleration

Now, the simulations are carried out with the dexterity optimized configuration motor and cable connection points using the controller that was designed in section 7.2. Figure 7.12 – 7.15 show the simulation resuls.

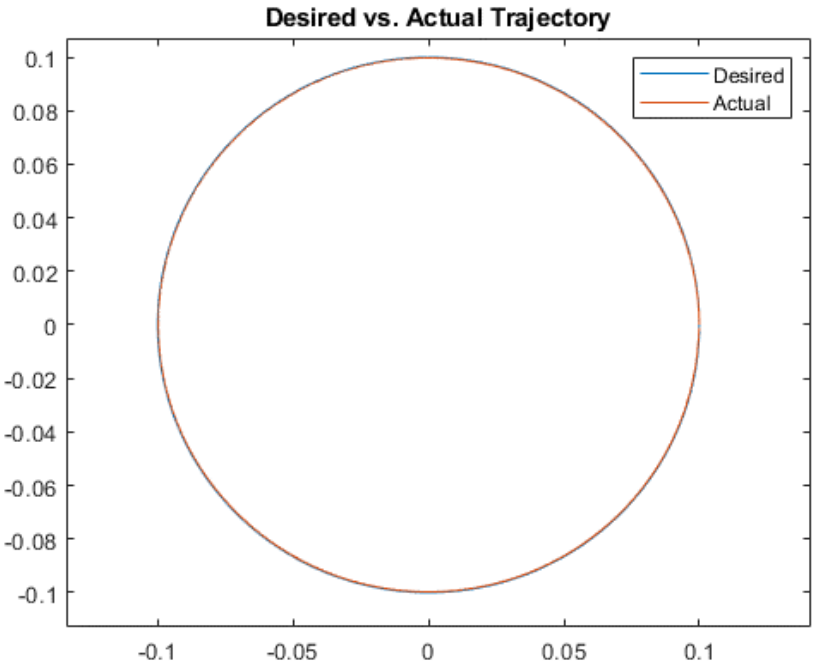

Figure 7.12 Circle – Real vs Actual Trajectory

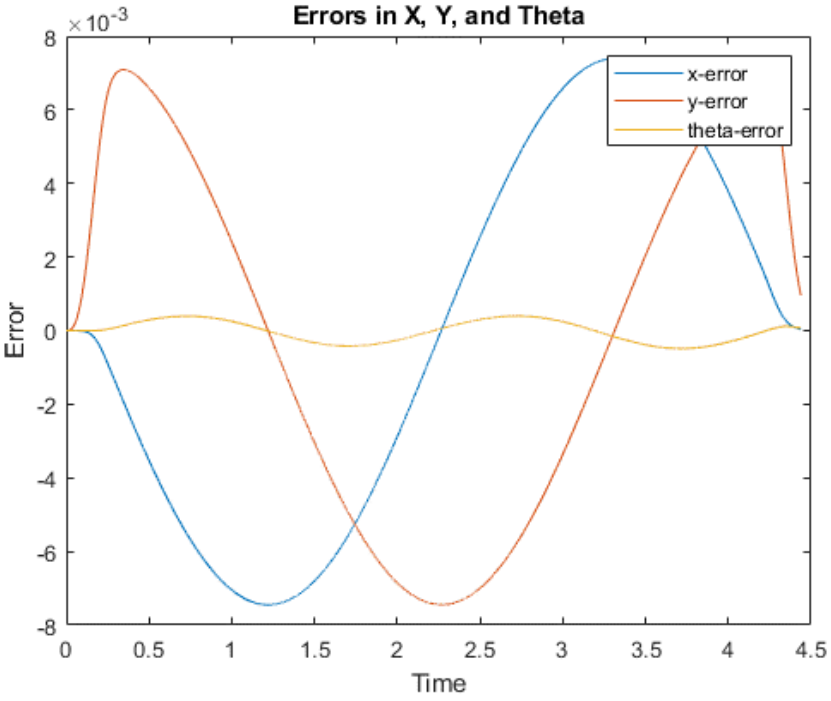

Figure 7.13 Circle – Tracking Errors

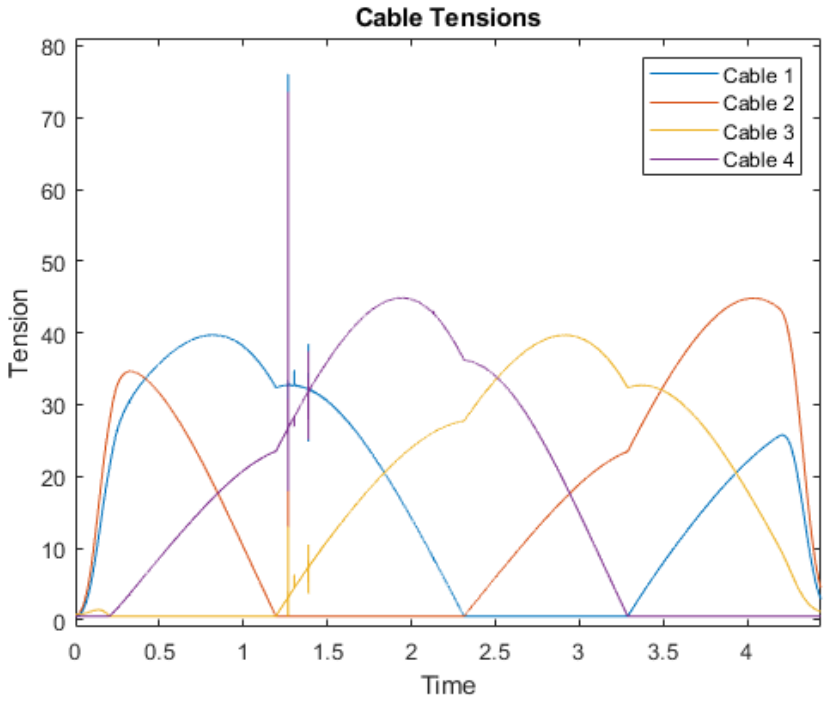

Figure 7.14 Circle – Desired Tensions

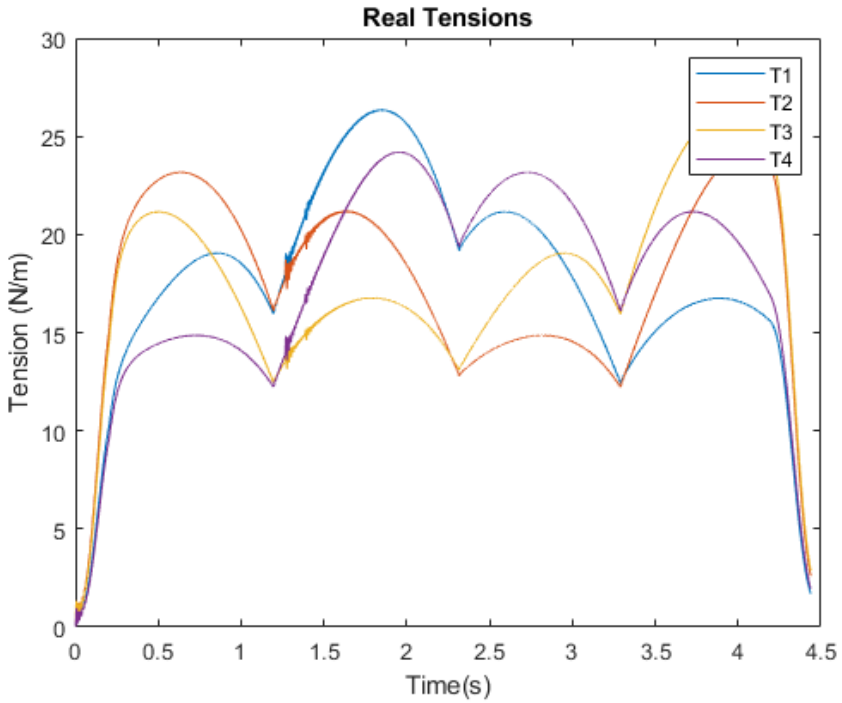

Figure 7.15 Circle – Real Tension

7.5.2. Trajectory Generation 1 – Rectangle

The rectangular trajectory consists of 4 individual linear segments. In the first simulation it is generated using the Point-to-Point approach where each segment begins and ends with zero velocity. Figure 7.16 below shows the Trapezoidal Acceleration Profile for the rectangular trajectory.

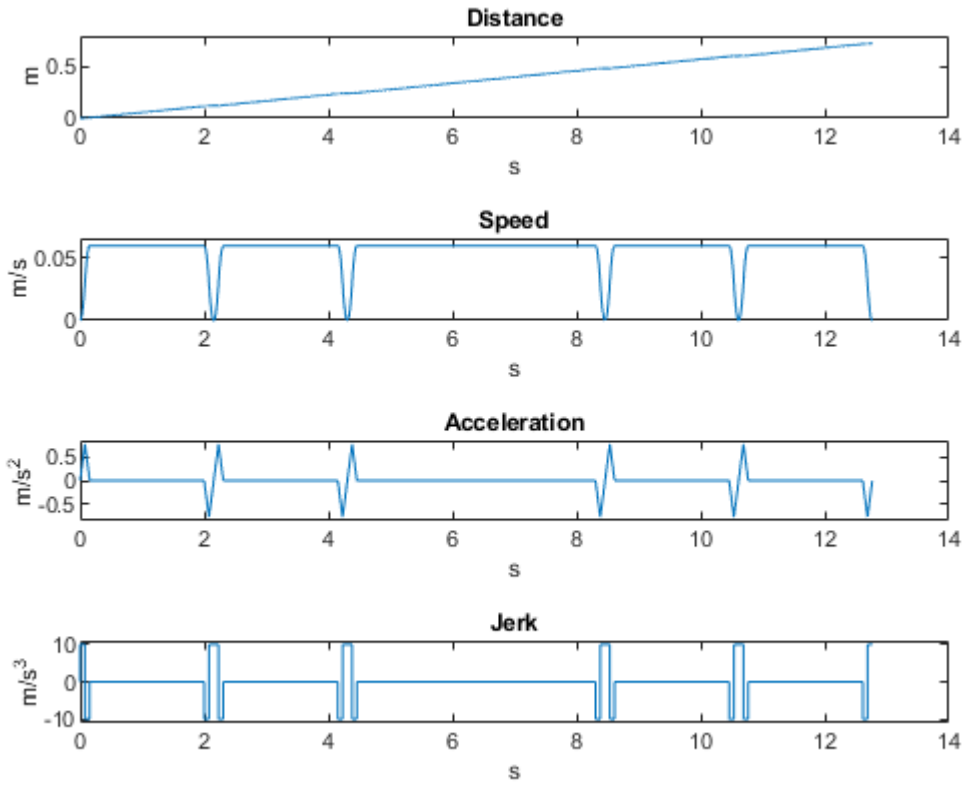

Figure 7.16 Rectangle – P2P – Trapezoidal Acceleration

Now, this simulation is carried out with the standard configuration of the cable robot with the controller designed in section 7.2. Figure 7.17 – 7.21 show the simulation results.

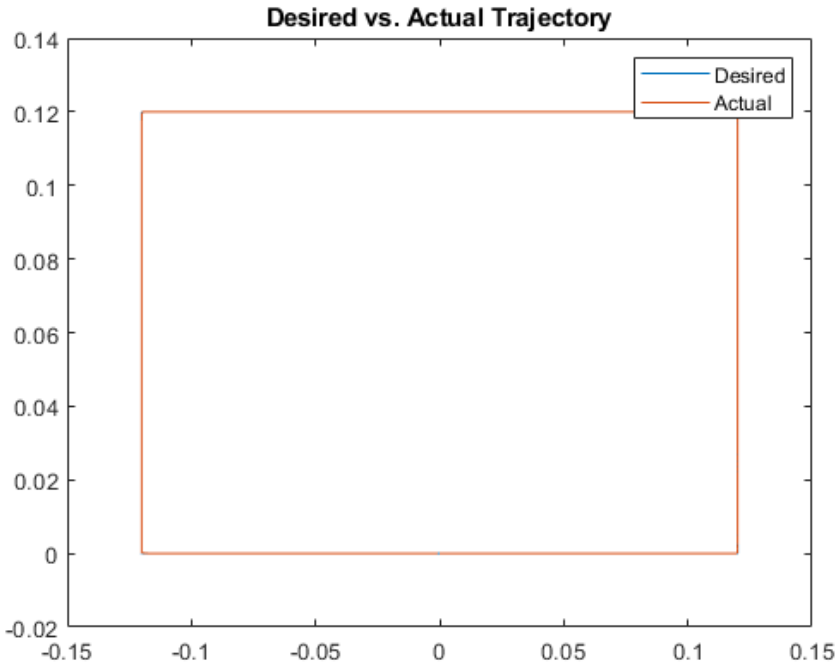

Figure 7.17 Rectangle – P2P – Desired/Real trajectory

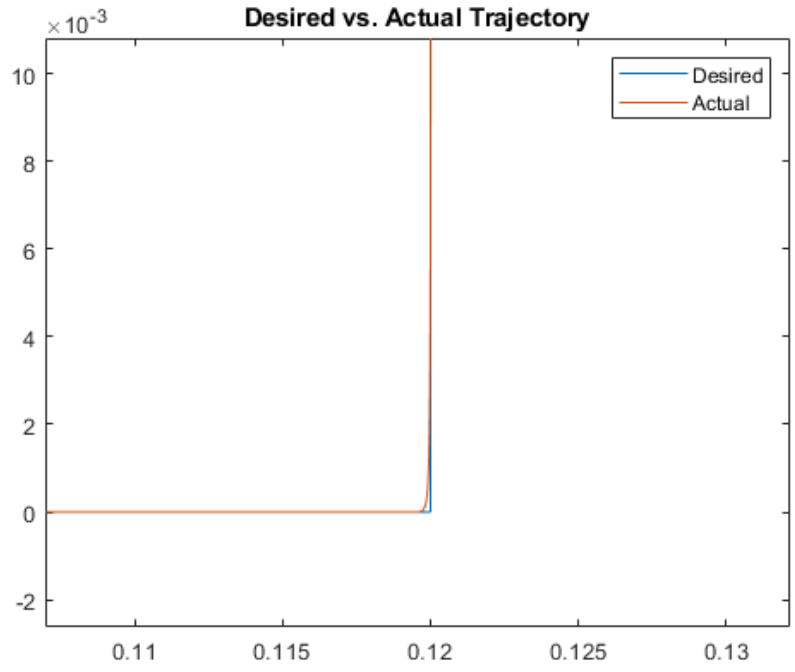

Figure 7.18 Rectangle – P2P – Corner zoom

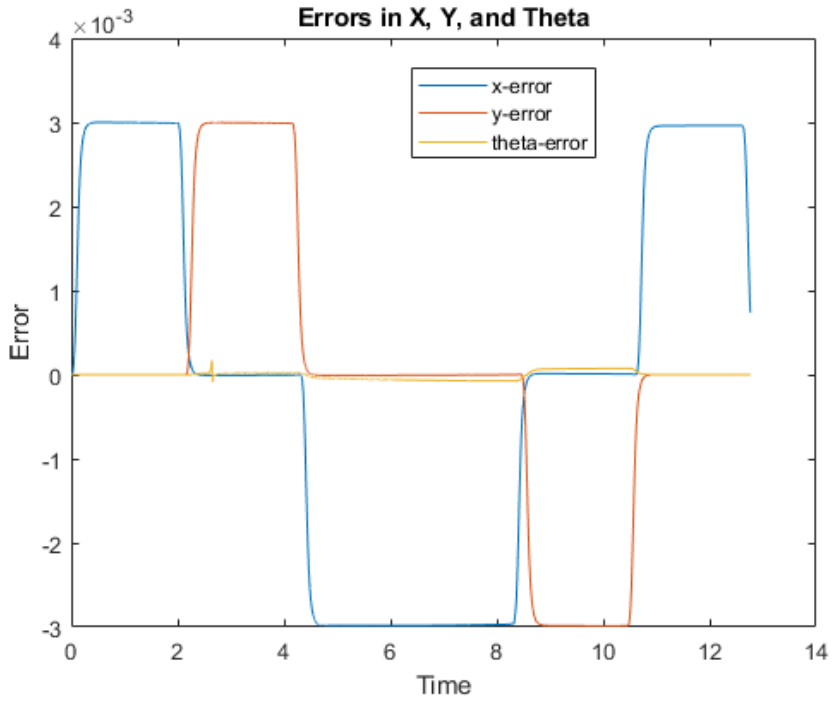

Figure 7.19 Rectangle – P2P – Tracking Errors

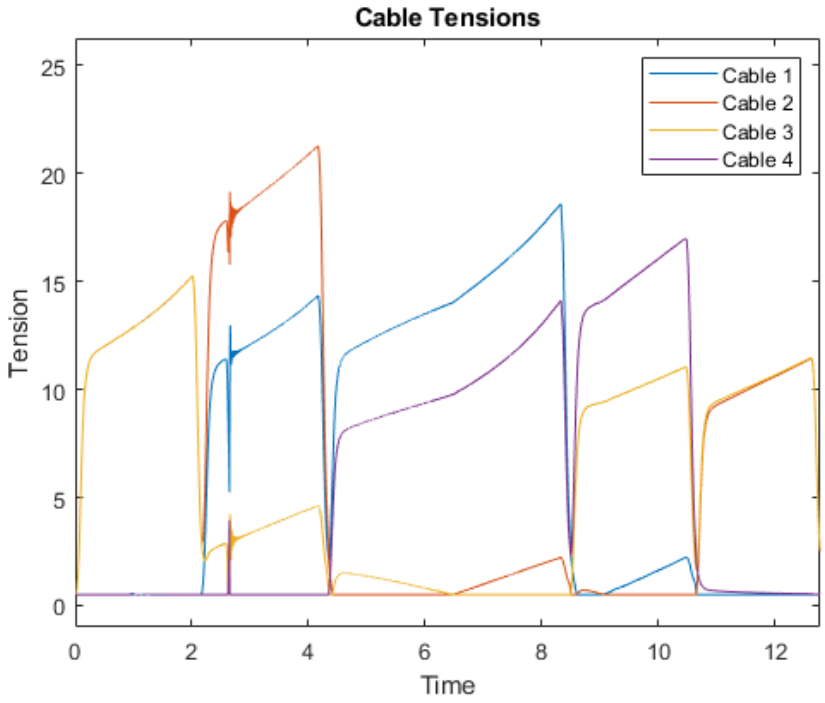

Figure 7.20 Rectangle – P2P – Desired Tensions

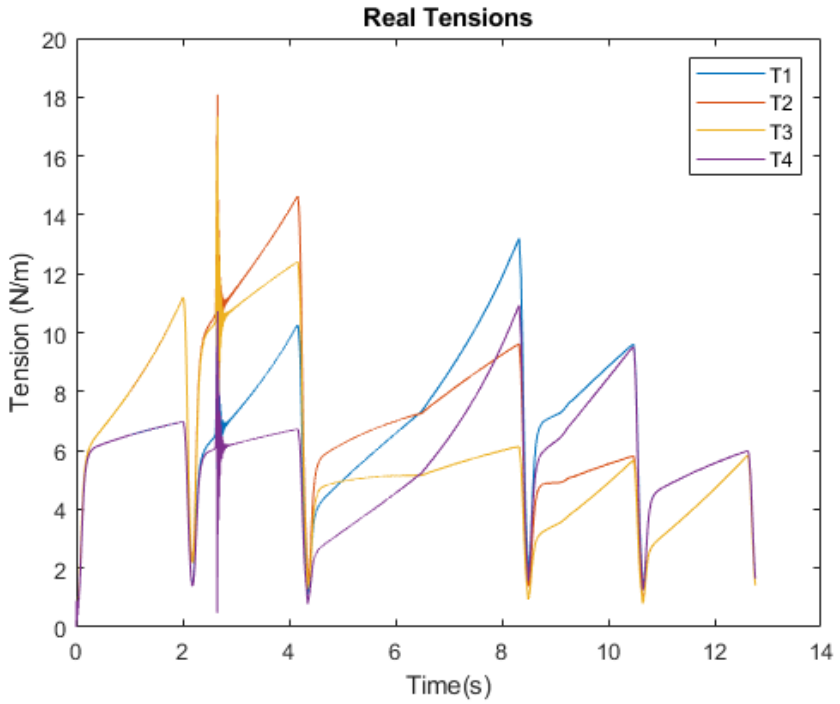

Figure 7.21 Rectangle – P2P – Real Tensions

Now the same trajectory is generated using the Continuous approach where the velocity at the corners does not go to zero. Figure 7.22 below shows the Trapezoidal Acceleration Profile for the trajectory.

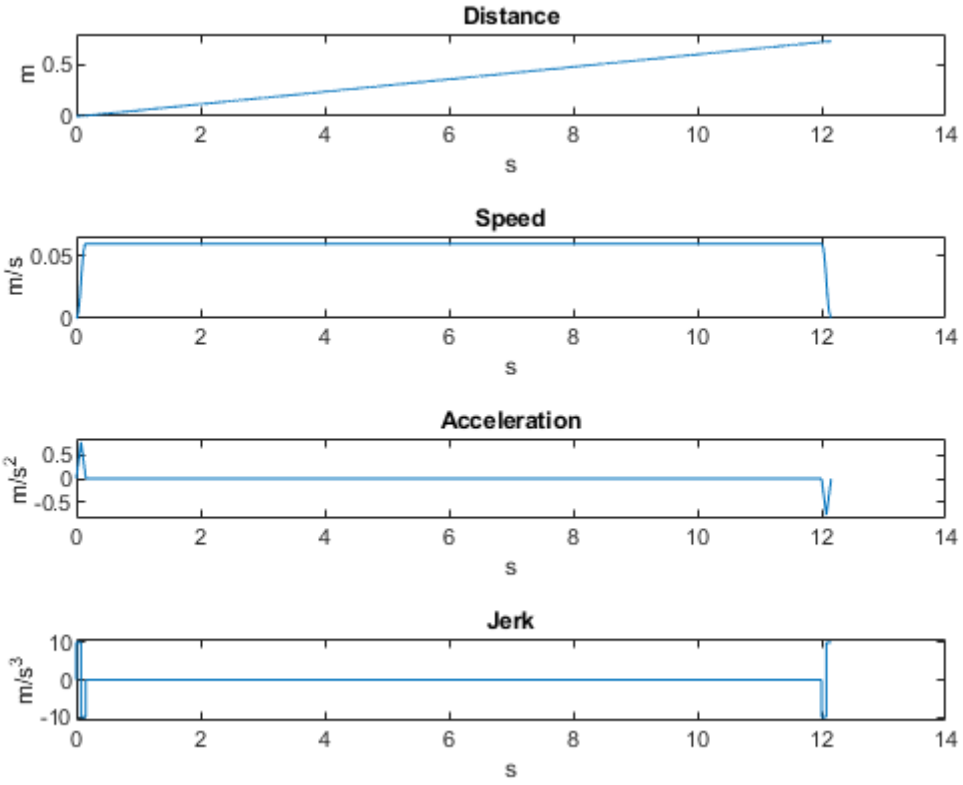

Figure 7.22 Rectangle – Continuous – Trapezoidal Acceleration

Figure 7.23 – 7.27 below show the simulation results.

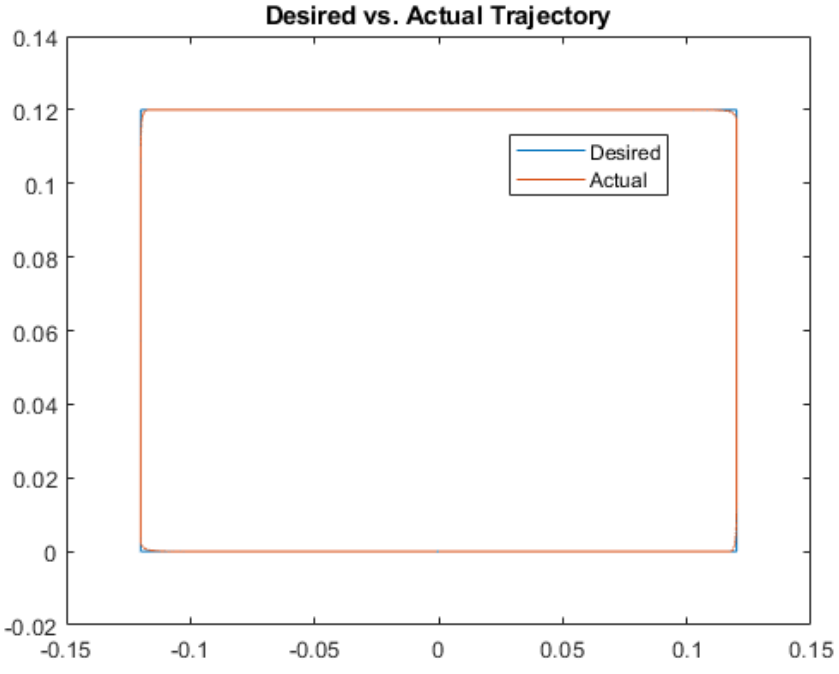

Figure 7.23 Rectangle – Continuous – Desired vs Actual Trajectory

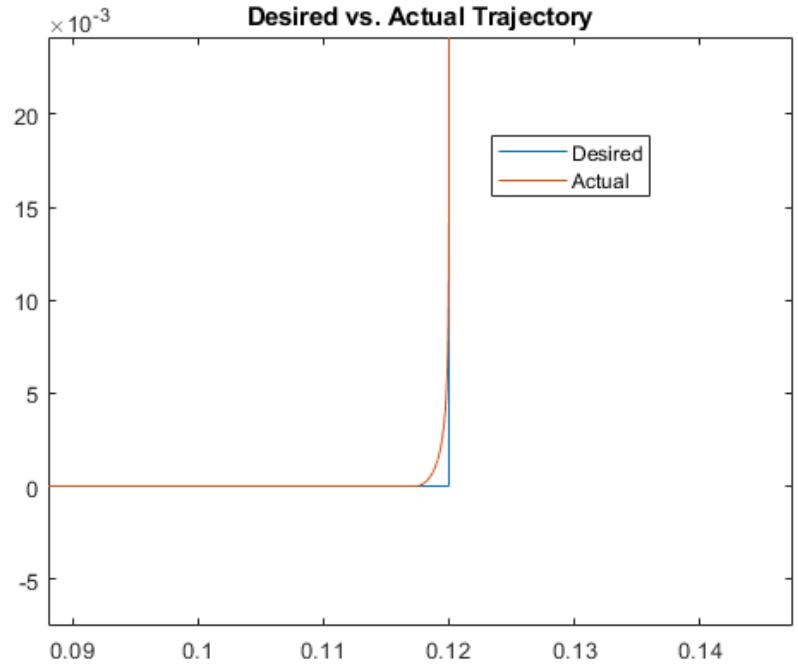

Figure 7.24 Rectangle – Continuous – Corner Zoom

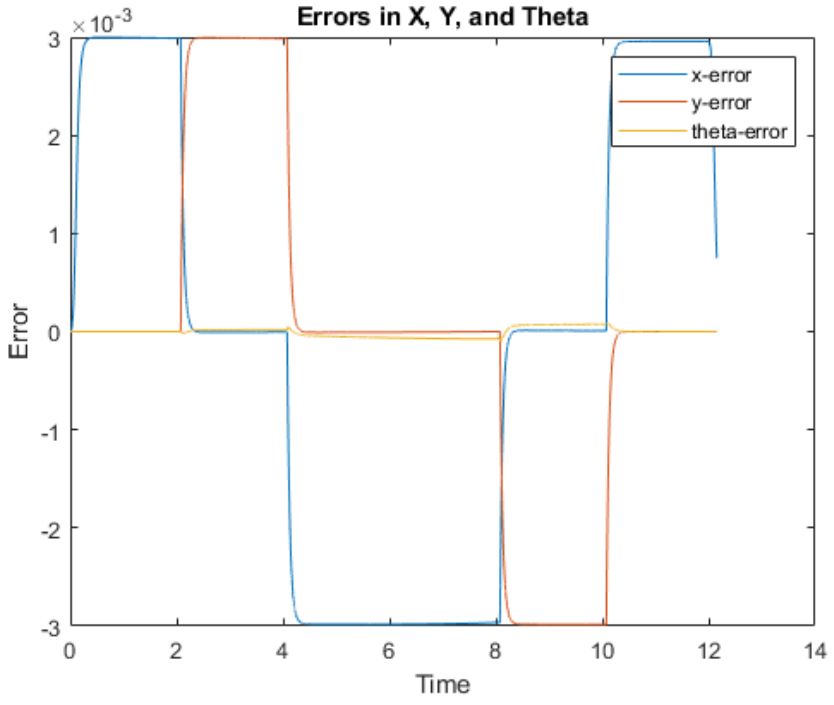

Figure 7.25 Rectangle – Continuous – Tracking Errors

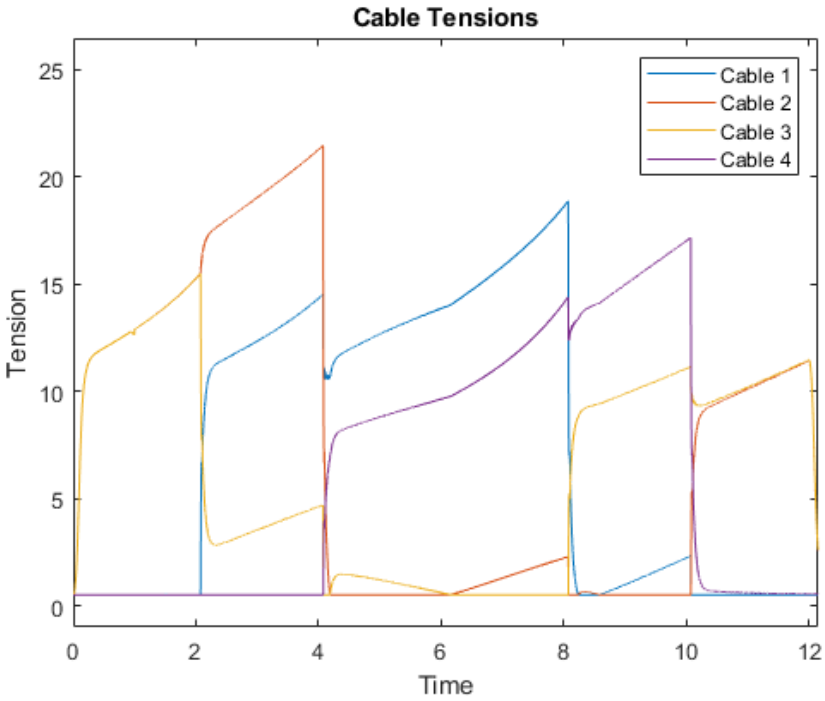

Figure 7.26 Rectangle – Continuous – Desired Tensions

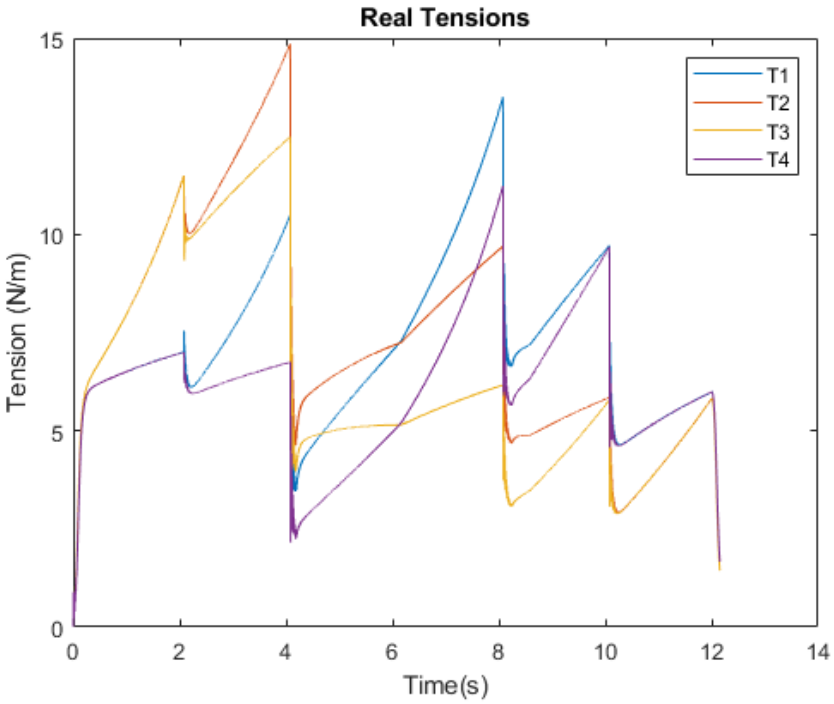

Figure 7.27 Rectangle – Continuous – Real Tensions

As observed, the high velocity at the corner does not yield good performance. To remedy this, the rectangular trajectory is generated using Microsplines where the edges of the

segments are joined using splines as given in section 4.1.3. These splines also calculate the proper spline travel time to preserve the simulation time. Figure 7.28 below shows the Trapezoidal Acceleration Profile for the trajectory.

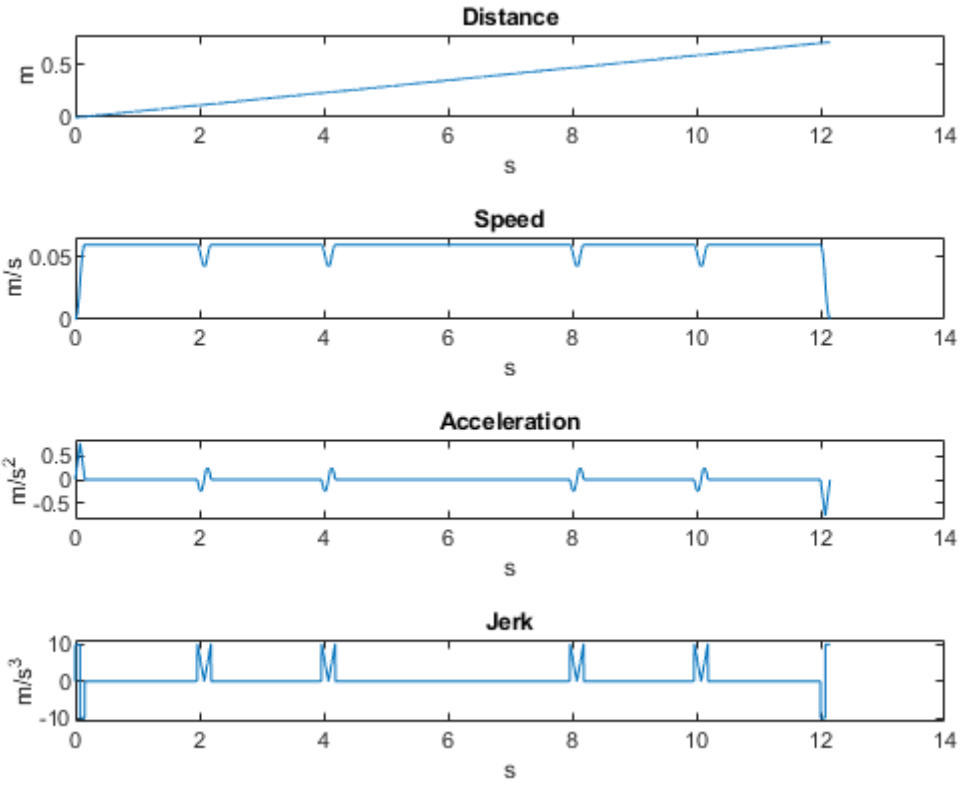

Figure 7.28 Rectangle – Splines – Trapezoidal Acceleration

Figure 7.29 – 7.33 show the simulation results.

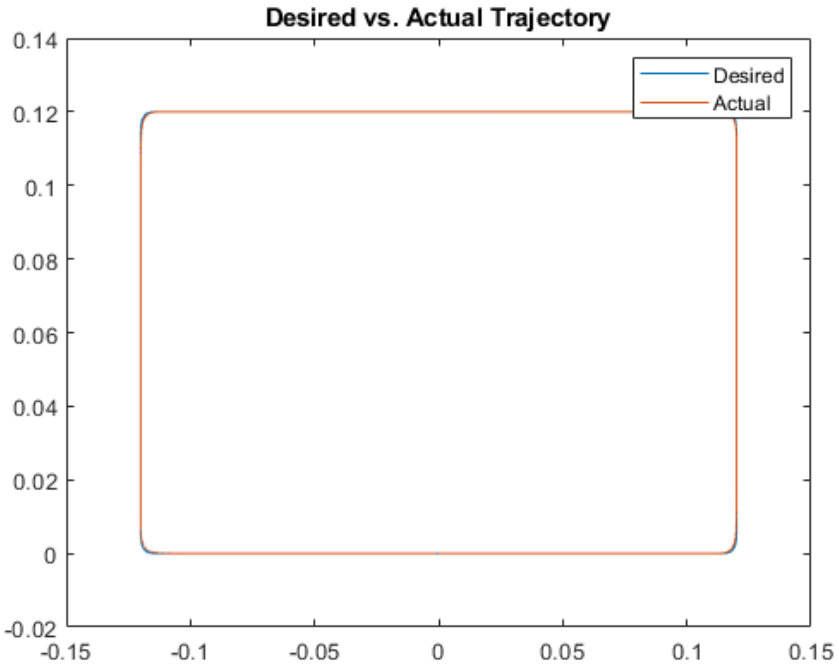

Figure 7.29 Rectangle – Splines – Desired vs Real Trajectory

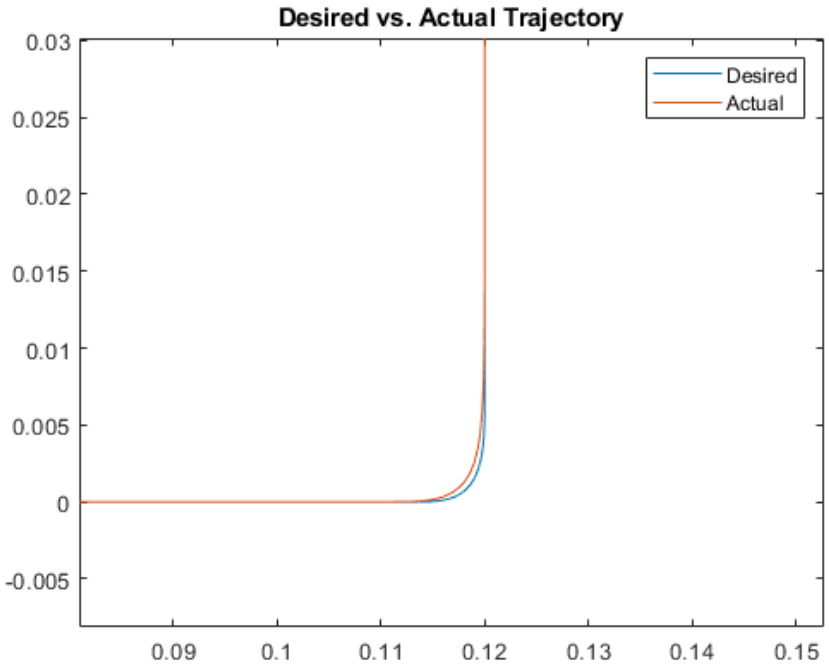

Figure 7.30 Rectangle – Splines – Corner Zoom

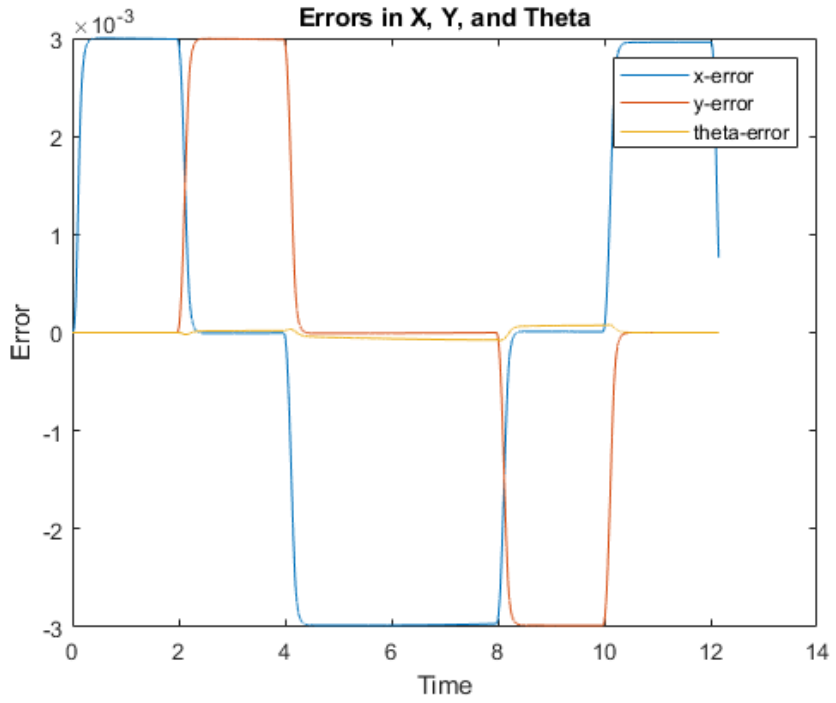

Figure 7.31 Rectangle – Splines – Tracking Errors

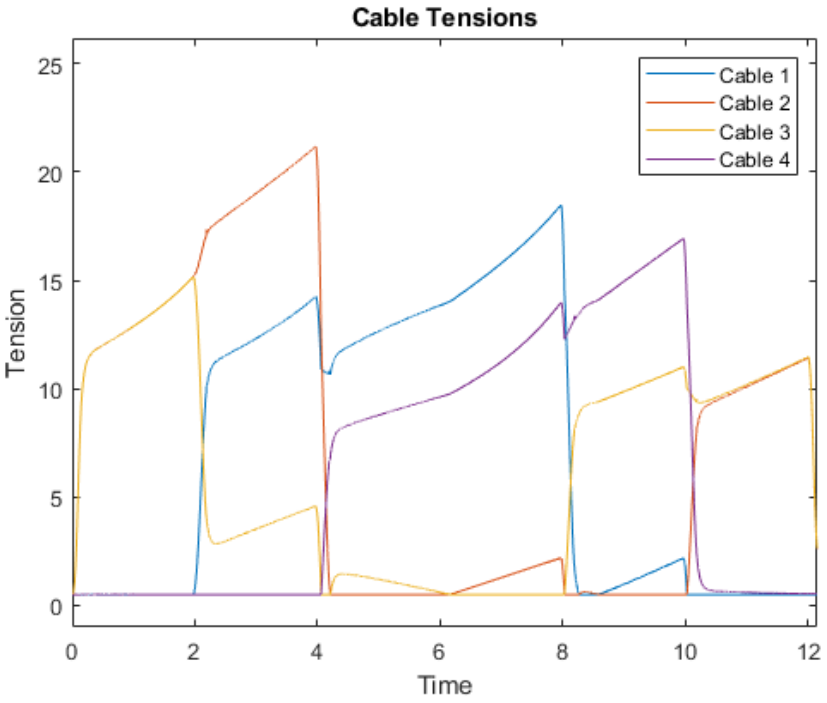

Figure 7.32 Rectangle – Splines – Desired Tensions

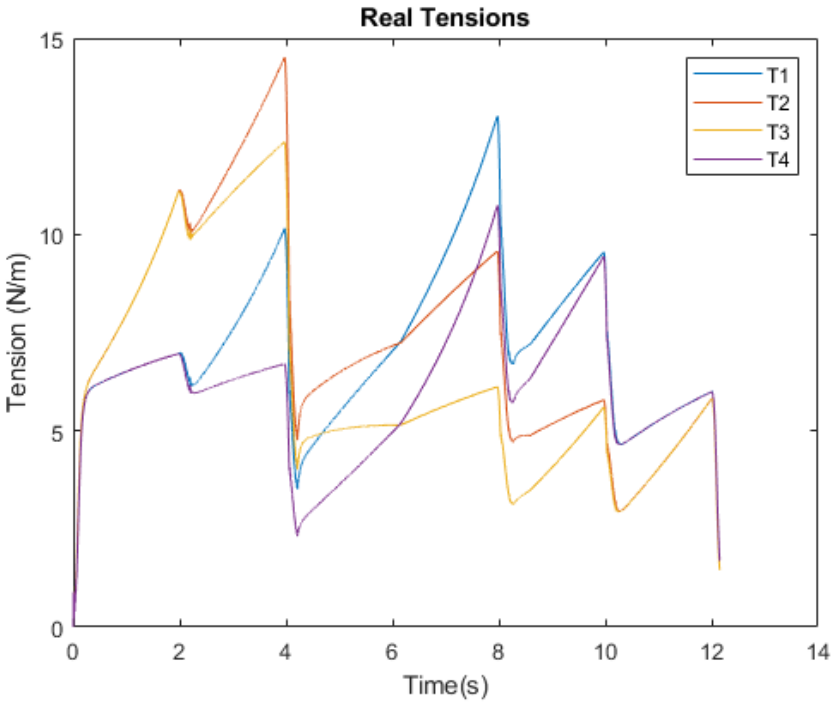

Figure 7.33 Rectangle – Splines – Real Tensions

7.5.3. Trajectory Generation 3 – Spiral

Now the spiral type trajectory consists of arc segments with constantly changing radii. This kind of trajectory generation cannot be handled with simple interpolation. The trajectory generation algorithm defined in section 4.4 is used to generate this trajectory by giving it a series of reference points.

Each point is connected with a 5^{th} order spline as given in section 4.4, according to the desired velocity and acceleration profile. Figure 7.34 below shows the reference points and the generated trajectory.

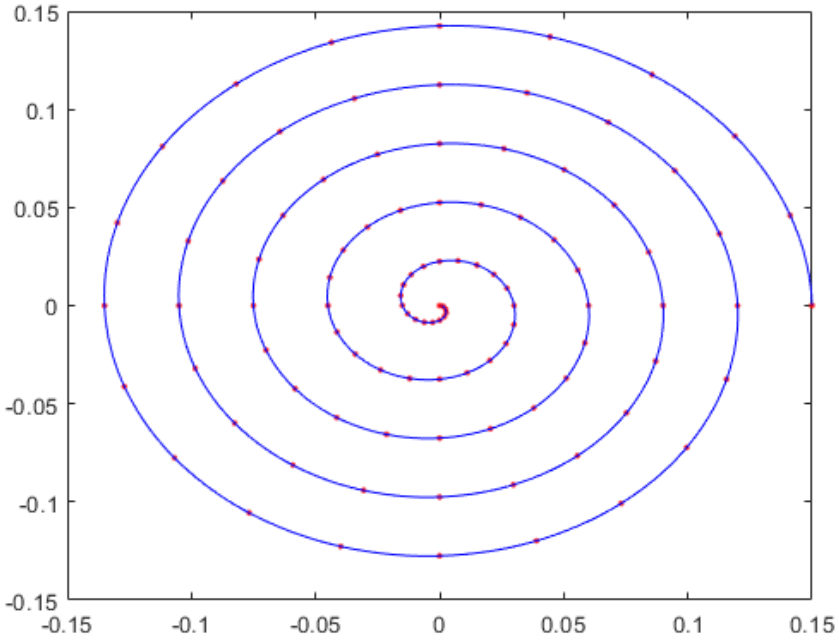

Figure 7.34 Spiral – Reference points

Figure 7.35 below shows the Trapezoidal Acceleration profile for the desired trajectory.

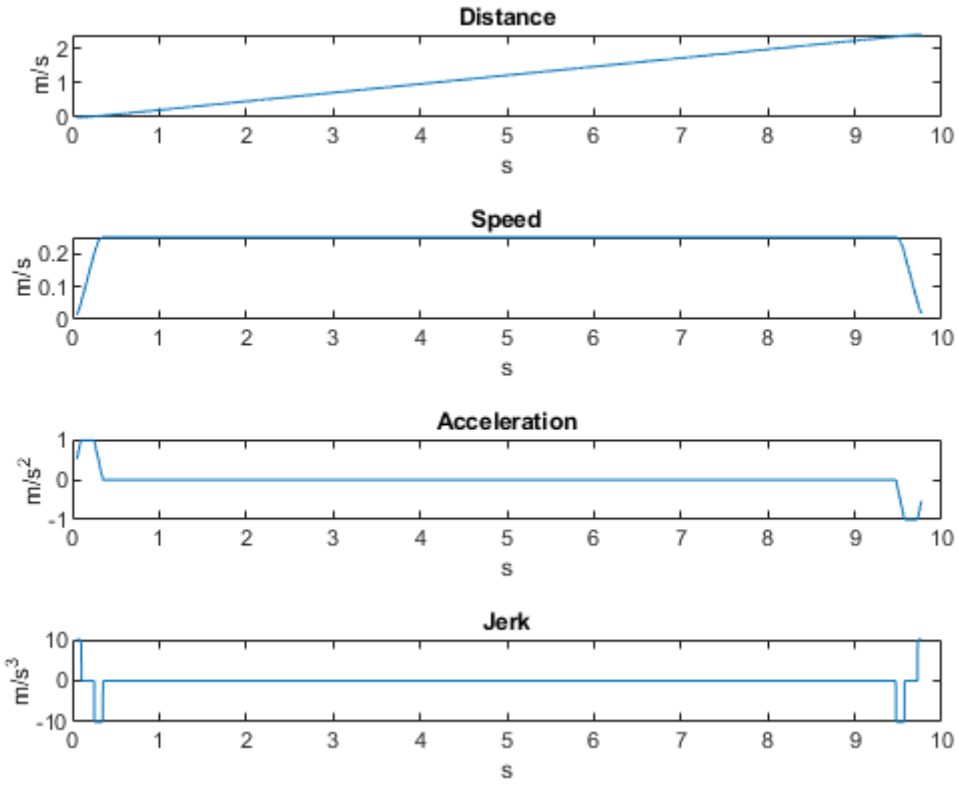

Figure 7.35 Spiral – Trapezoidal Acceleration

Figures 7.36 – 7.39 below show the simulation results.

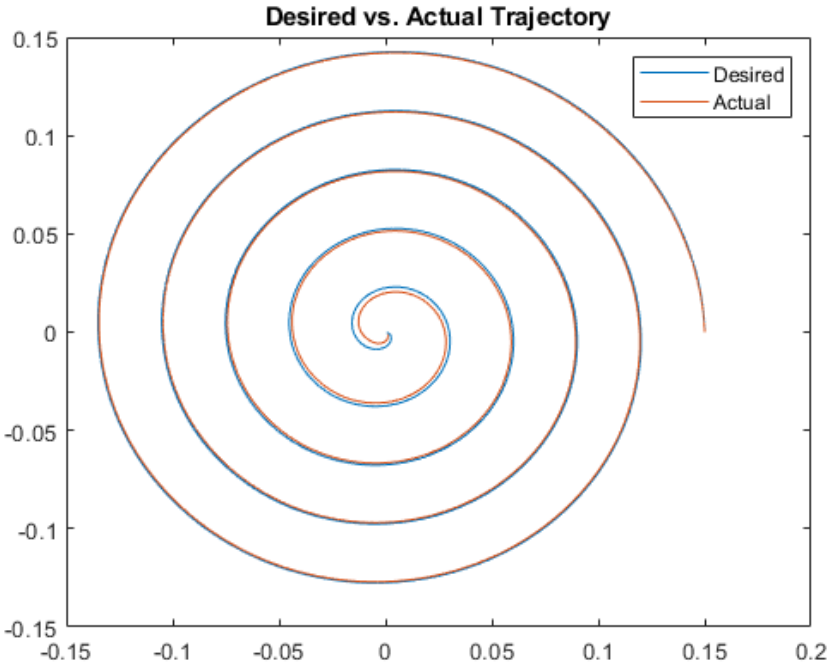

Figure 7.36 Spiral – Desired vs Actual Trajectory

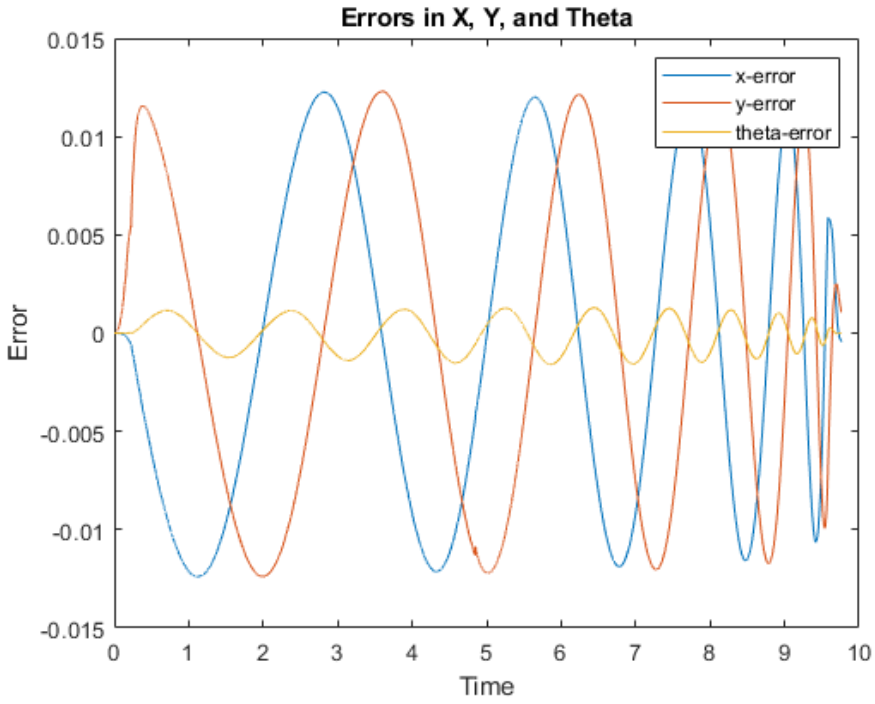

Figure 7.37 Spiral – Tracking Errors

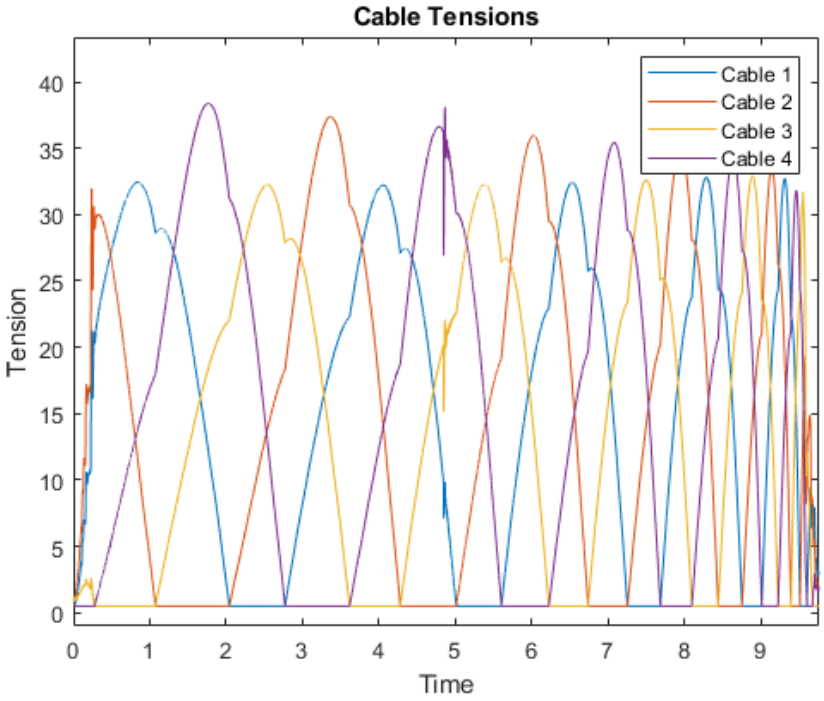

Figure 7.38 Spiral – Desired Tensions

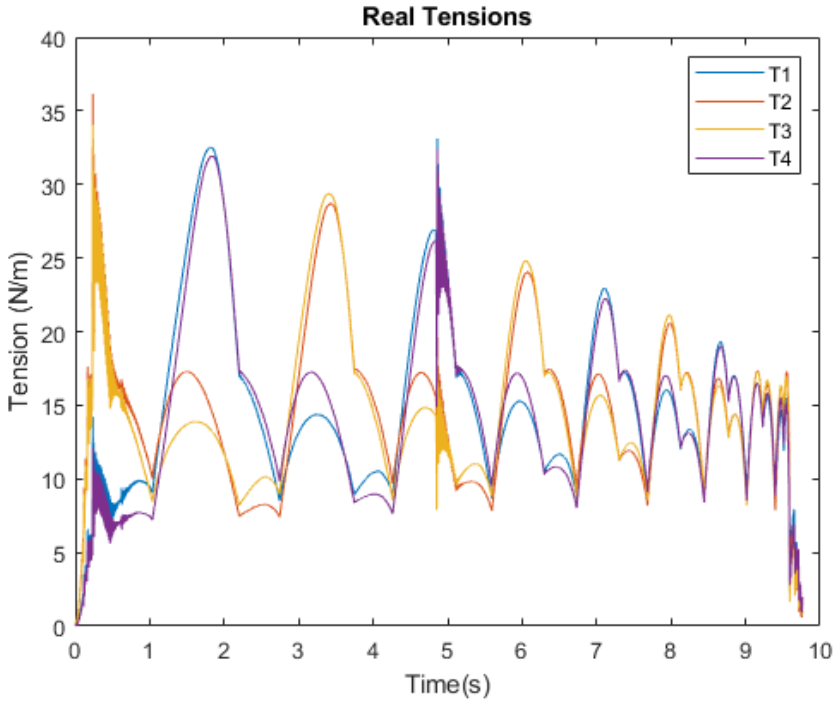

Figure 7.39 Spiral – Real Tensions

7.5.4. Trajectory Generation 2 – Custom Data

While the previous two algorithms are capable of generating a variety of different trajectories for standard and non-standard shapes, they are not able to simulate a trajectory where the acceleration and velocity are constantly changing. Often it might be required to simulate a custom trajectory, which can be fed to the cable robot but it needs to be compatible.

Three example trajectories can be called as Custom Trajectory A, Custom Trajectory B, and Custom Trajectory C. These trajectories are resampled at the desired control loop frequency or sampling rate and then connected with splines and linear segments to the desired start/end point as given in section 4.3. After Custom Trajectory A is selected, Figure 7.40 – 7.44 show the simulation results.

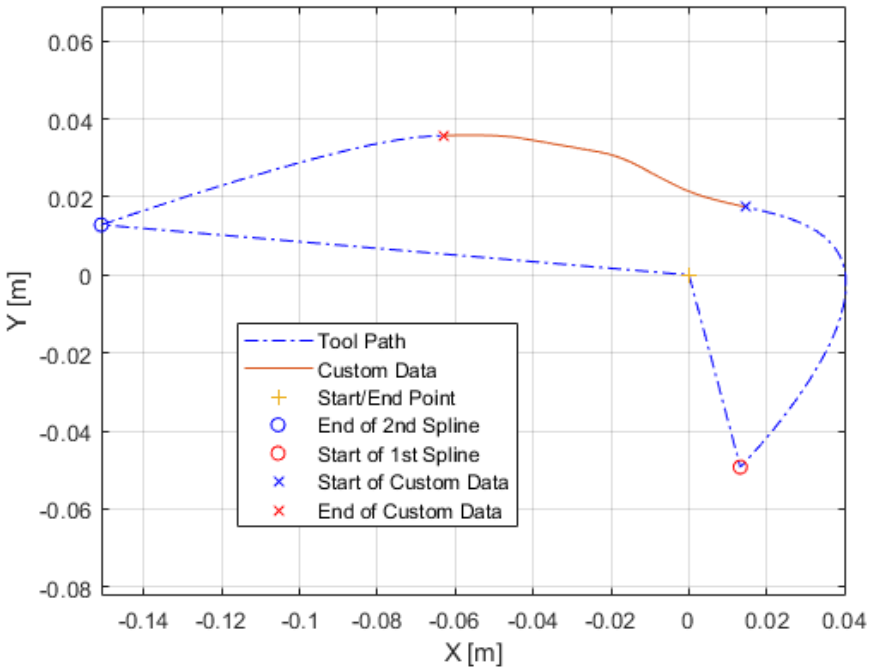

Figure 7.40 Custom Data A – Trajectory

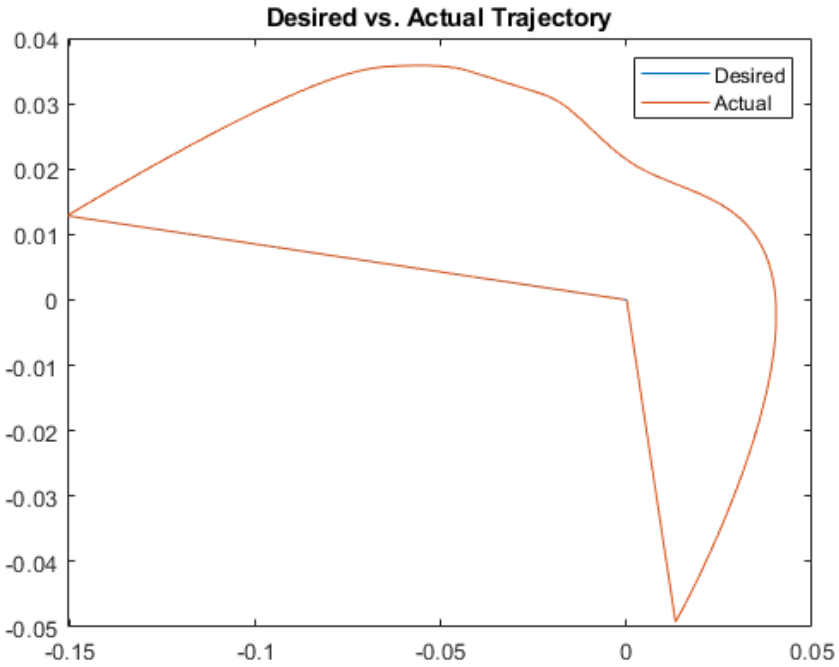

Figure 7.41 Custom Data A – Desired vs Real Trajectory

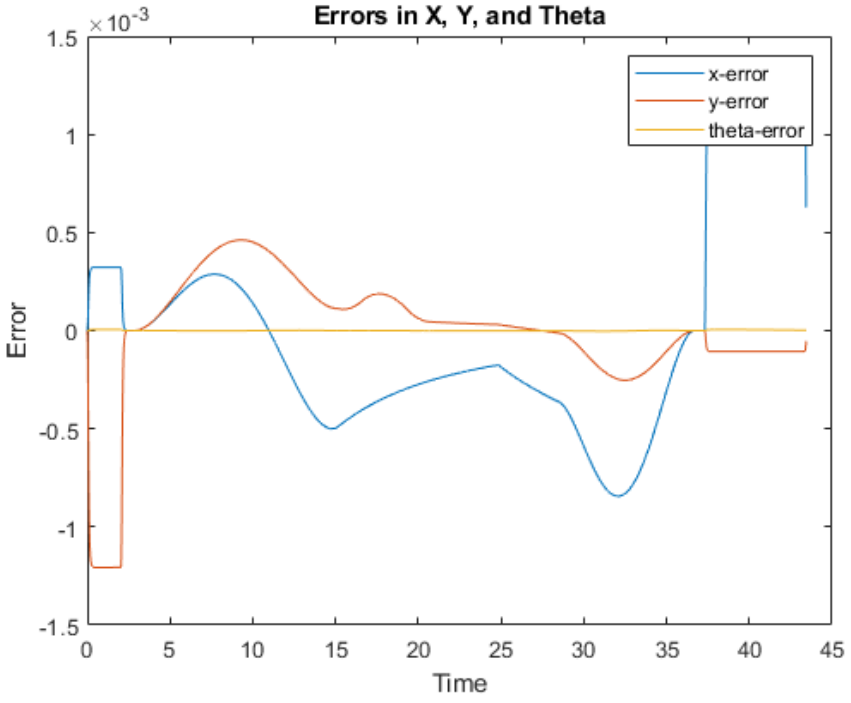

Figure 7.42 Custom Data A – Tracking Errors

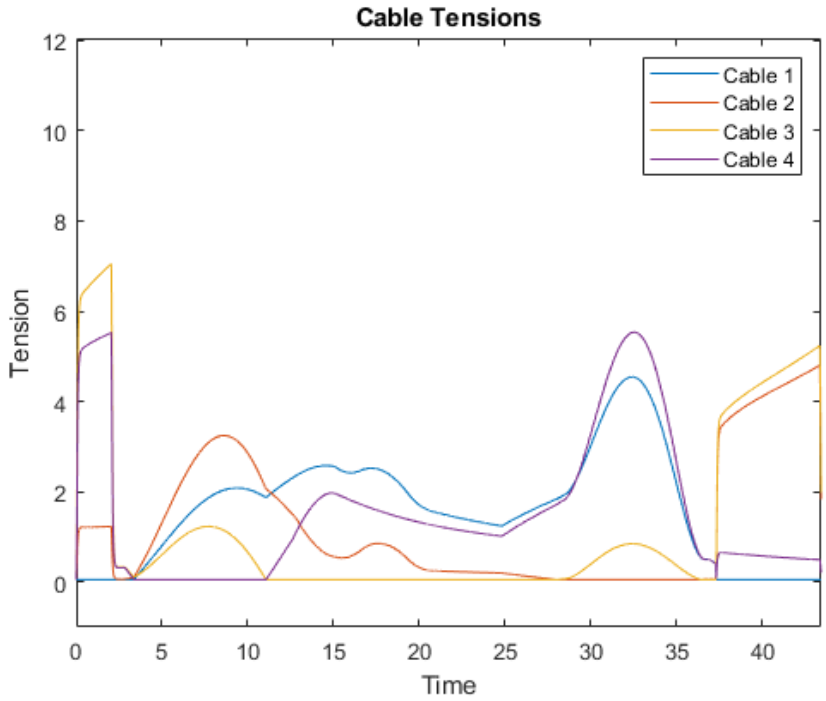

Figure 7.43 Custom Data A – Desired Tensions

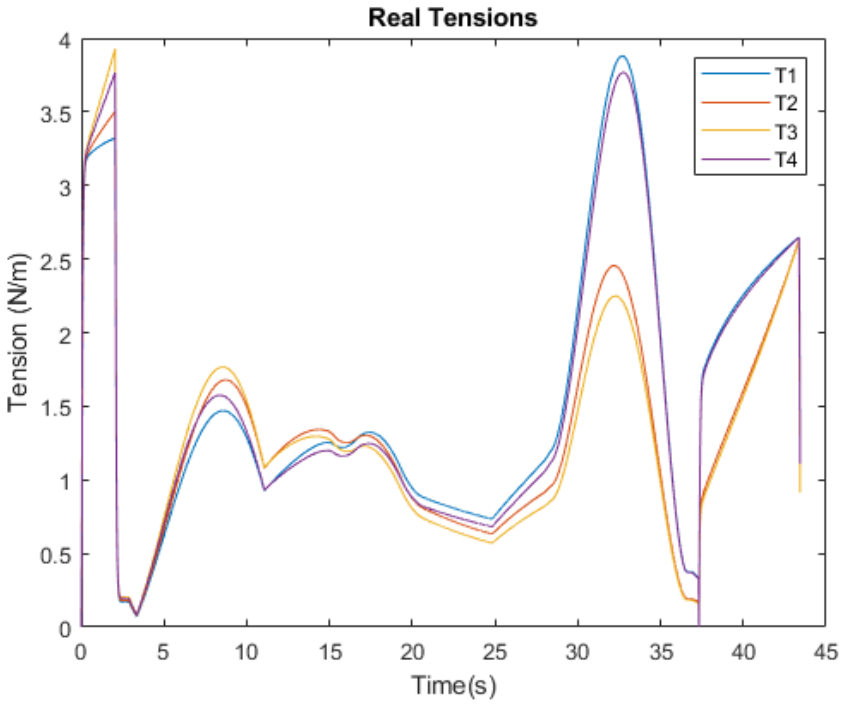

Figure 7.44 Custom Data A – Real Tensions

After Custom Trajectory B is selected, Figure 7.45 – 7.49 show the simulation results.

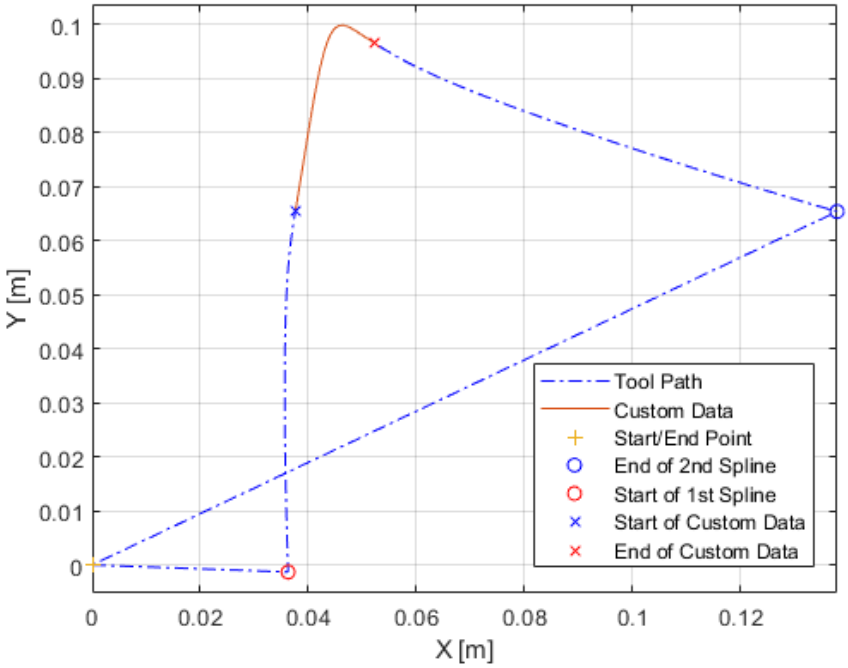

Figure 7.45 Custom Data B – Trajectory

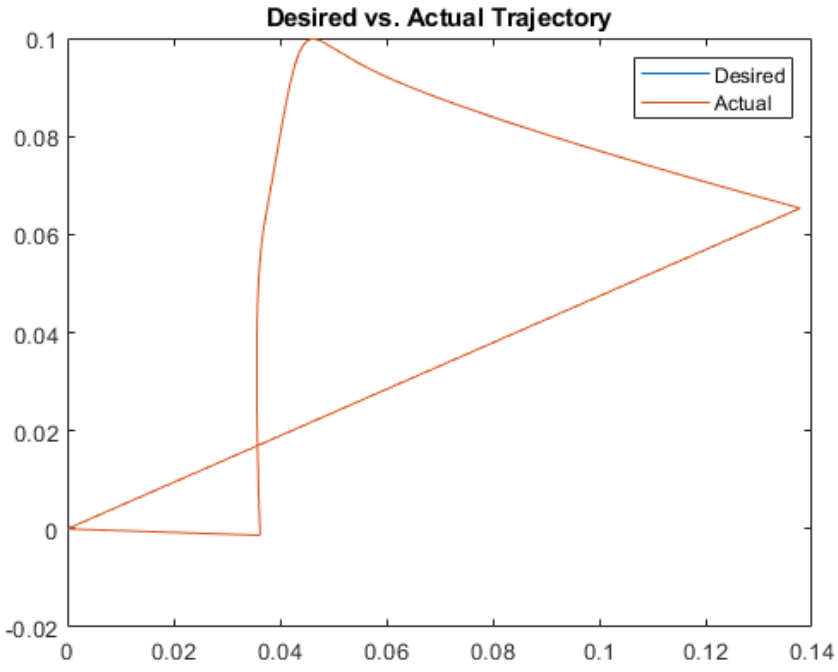

Figure 7.46 Custom Data B – Desired vs Actual Trajectory

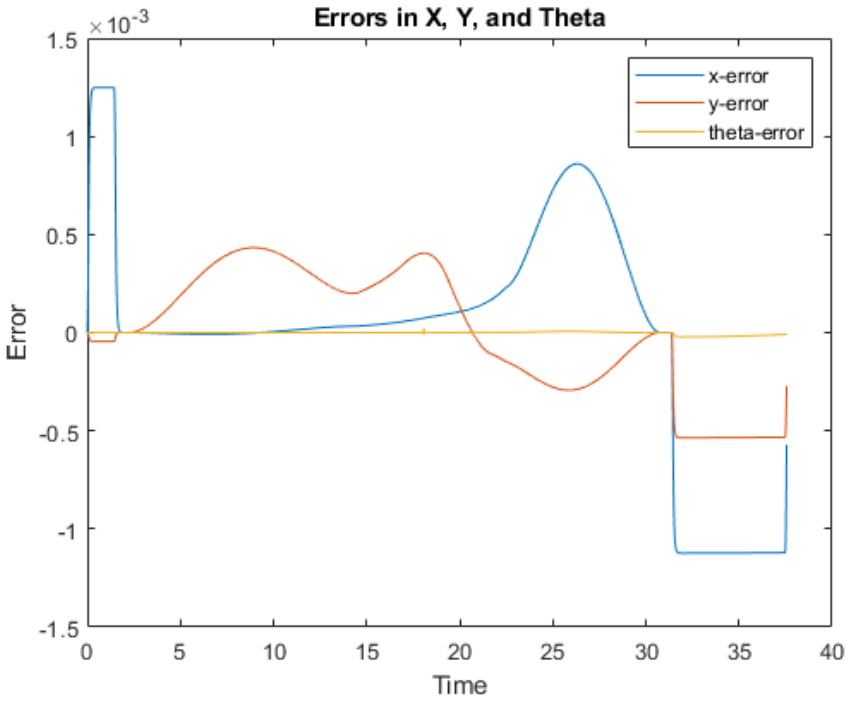

Figure 7.47 Custom Data B – Tracking Errors

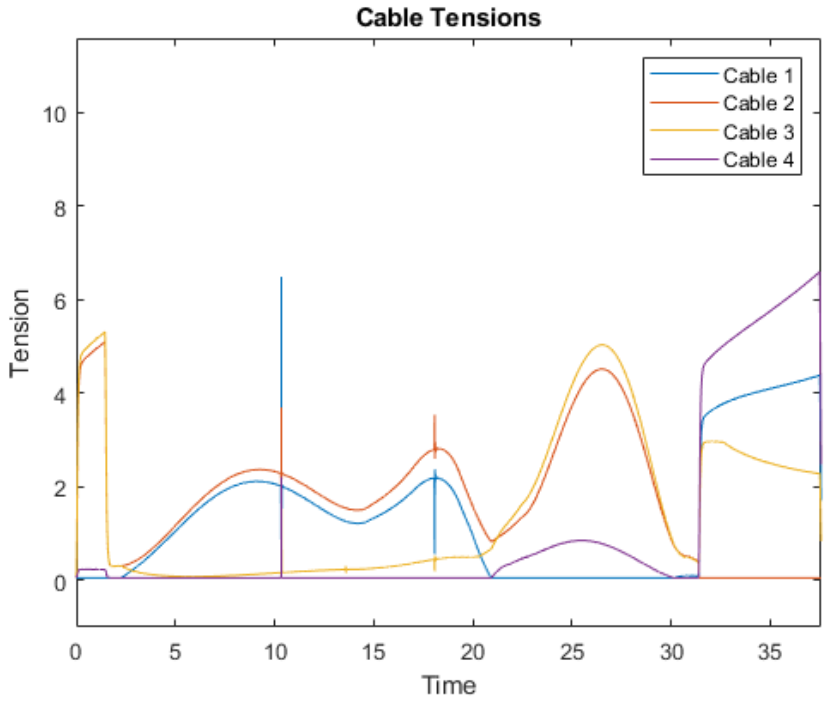

Figure 7.48 Custom Data B – Desired Tensions

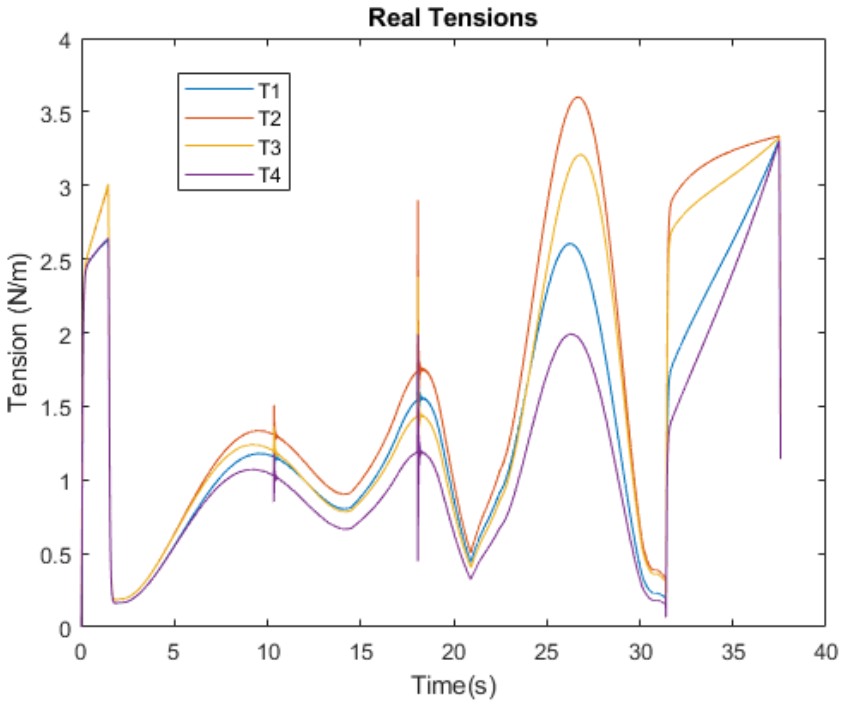

Figure 7.49 Custom Data B – Real Tensions

After Custom Trajectory C is selected, Figure 7.50 – 7.54 show the simulation results.

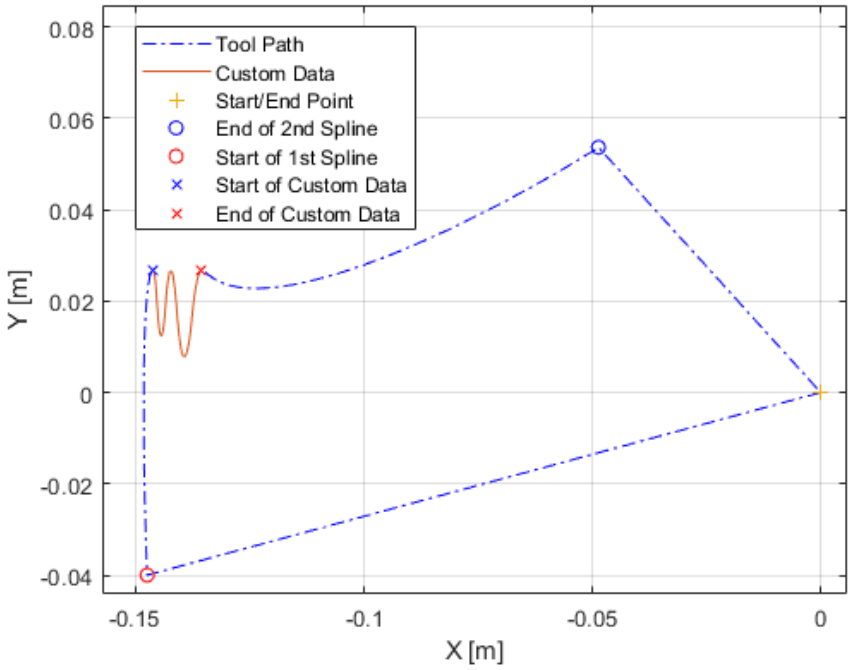

Figure 7.50 Custom Data C – Trajectory

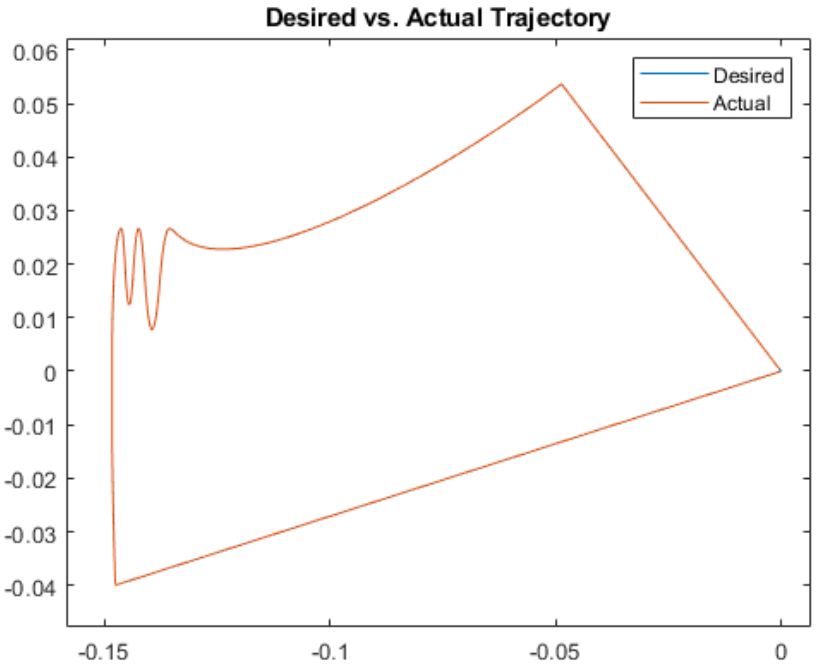

Figure 7.51 Custom Data C – Desired vs. Actual Trajectory

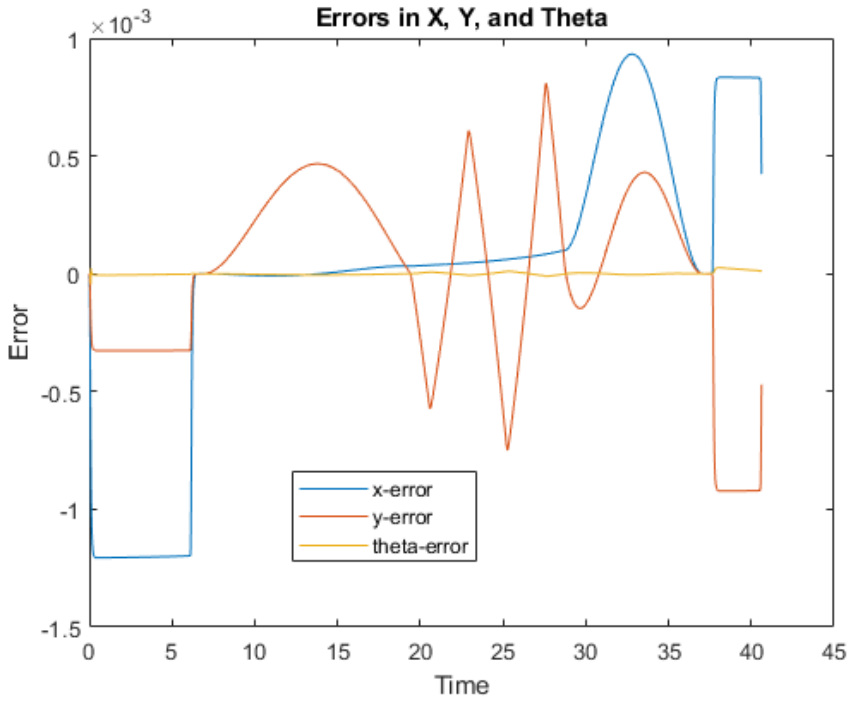

Figure 7.52 Custom Data C – Tracking Errors

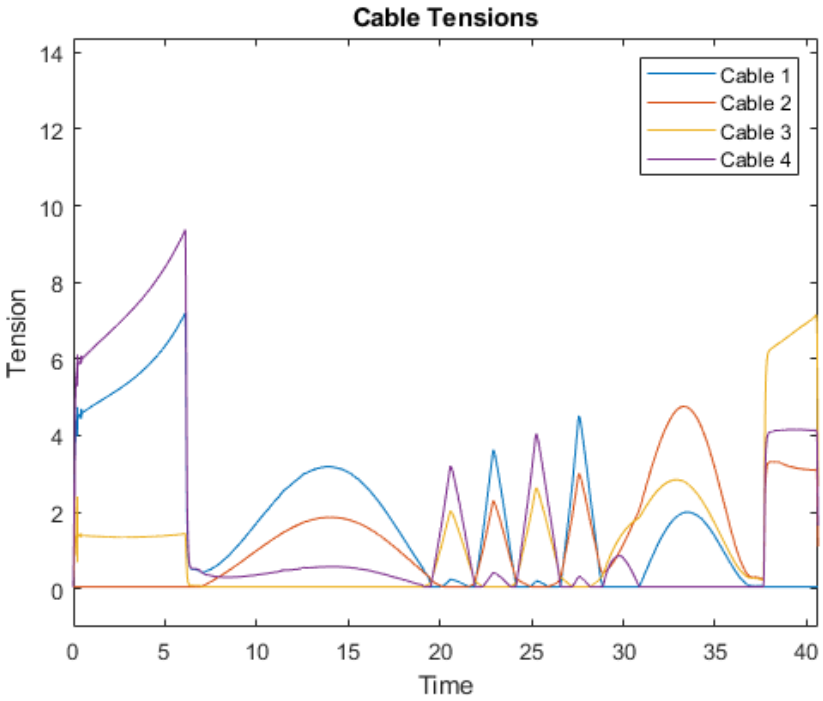

Figure 7.53 Custom Data C – Desired Tensions

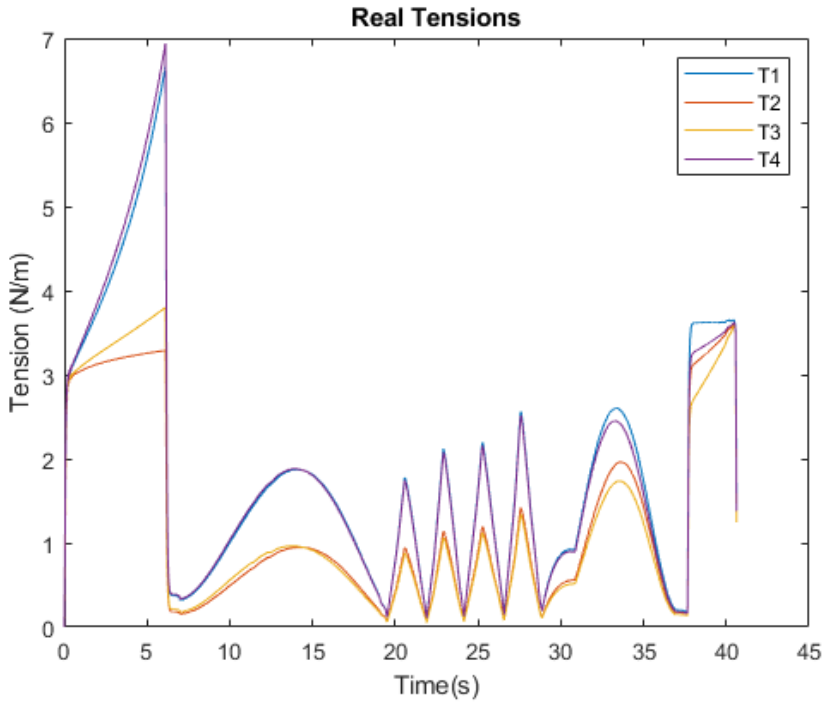

Figure 7.54 Custom Data C – Actual Tensions

7.5.5. Simulations with Increasing Speed

Another set of simulations is where the Circular Trajectory is used is to simulate the circular motion with increasing velocities. The simulations are caried out the results are given below.

Circle Speed: 0.06m/s^2

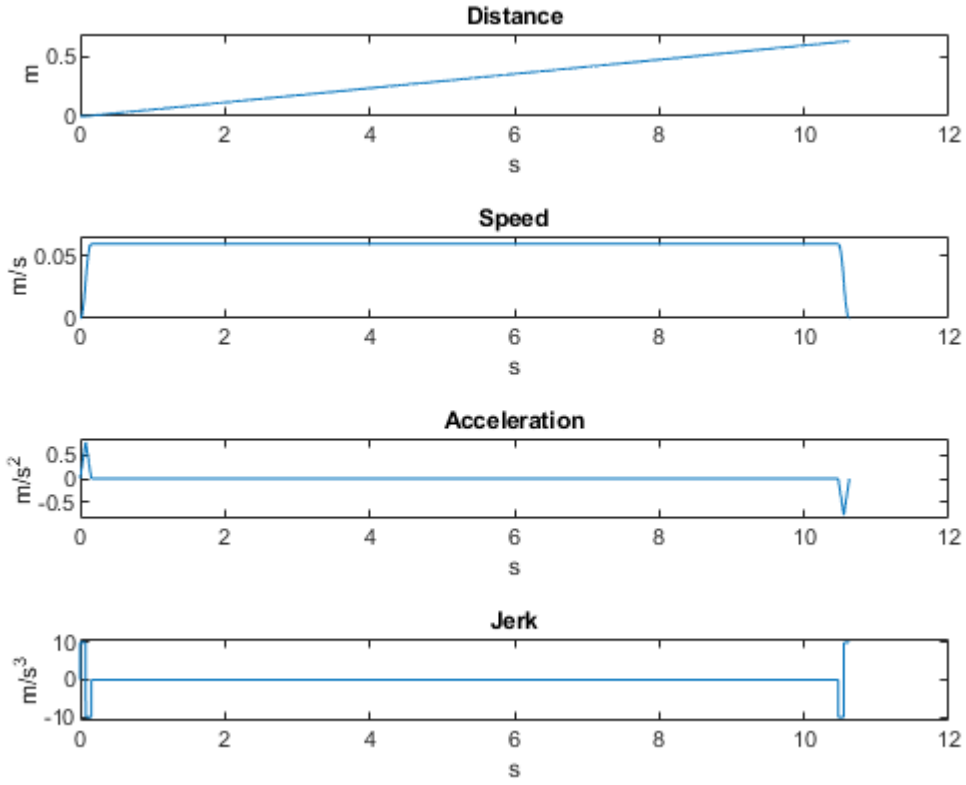

Figure 7.55 Circle 0.06 m/s^2 – Trapezoidal Acceleration

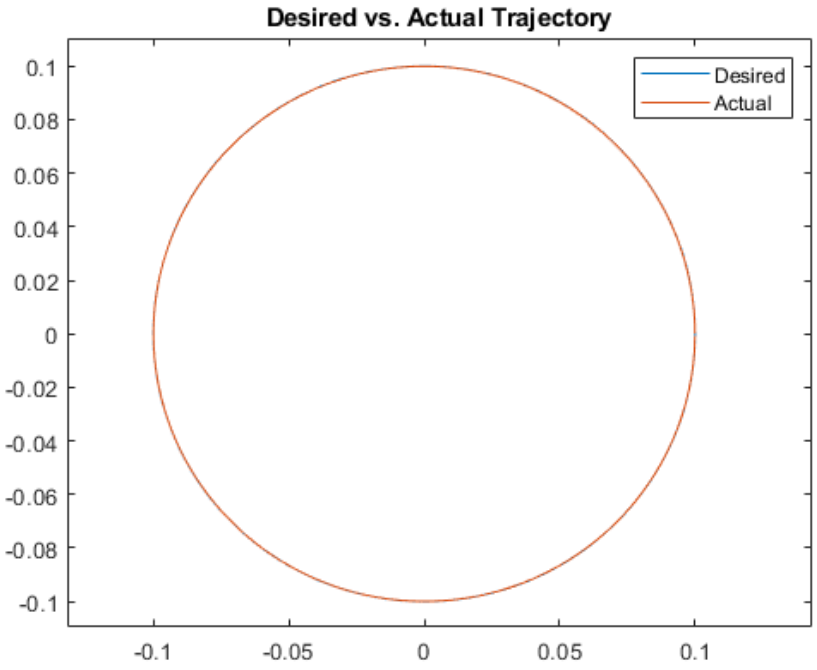

Figure 7.56 Circle 0.06 m/s^2 – Desired vs Actual Trajectory

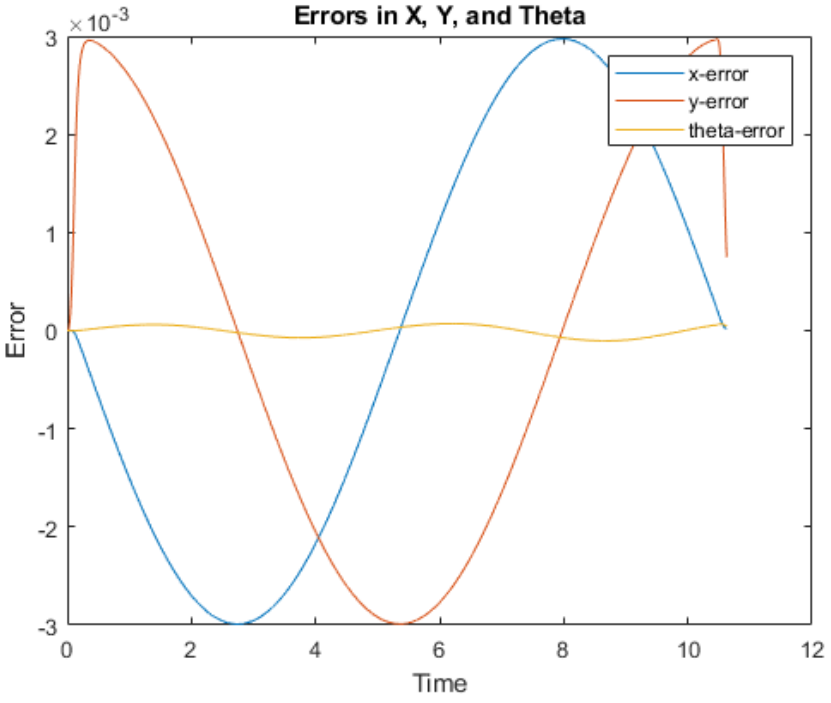

Figure 7.57 Circle 0.06 m/s^2 – Tracking Errors

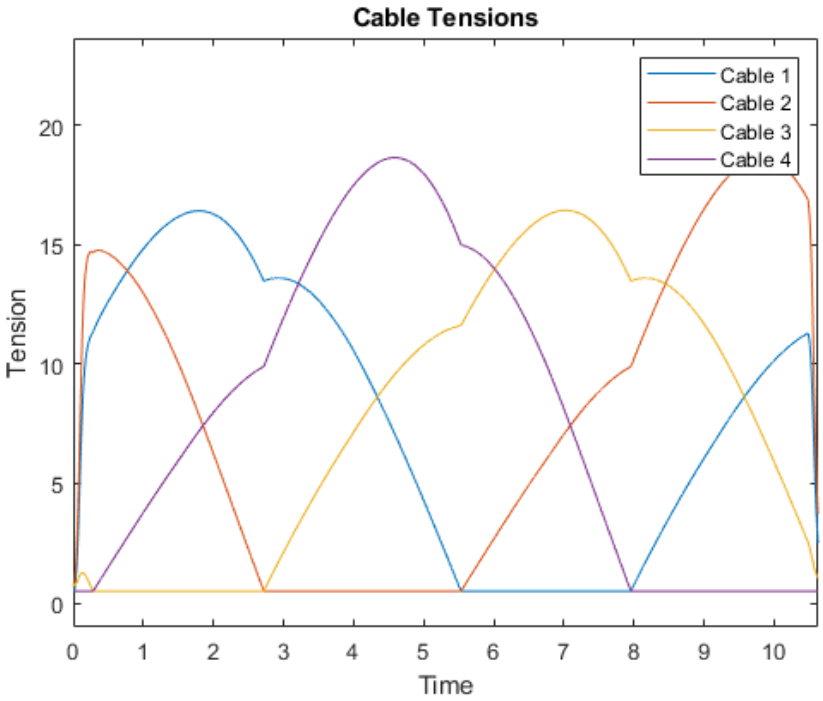

Figure 7.58 Circle 0.06 m/s^2 – Desired Tensions

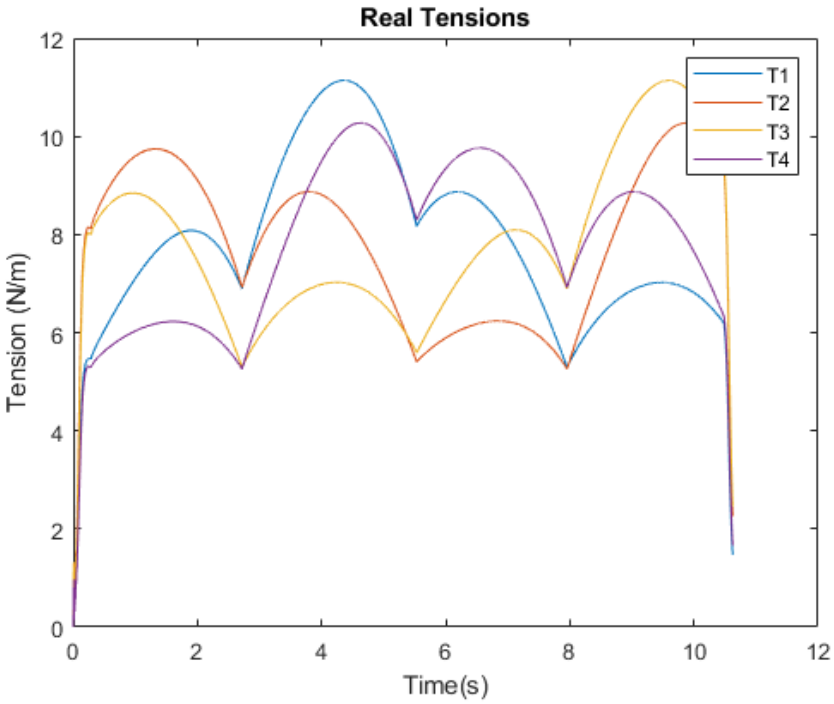

Figure 7.59 Circle 0.06 m/s^2 – Actual Tensions

Circle Speed: 0.12m/s^2

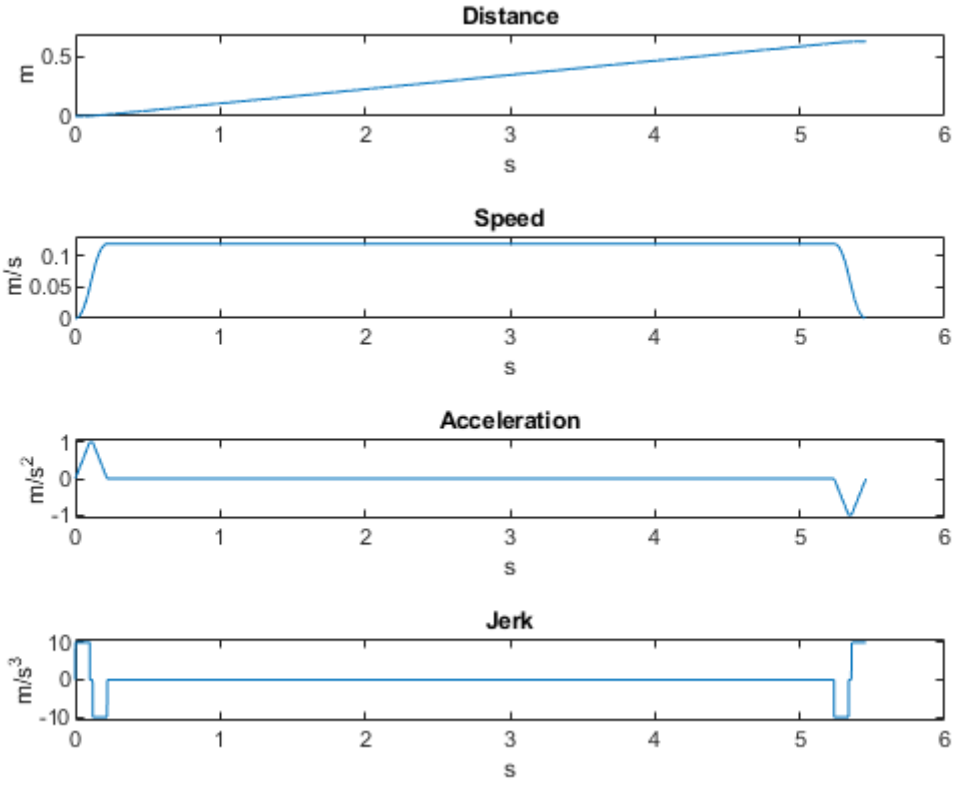

Figure 7.60 Circle 0.12 m/s^2 – Trapezoidal Acceleration

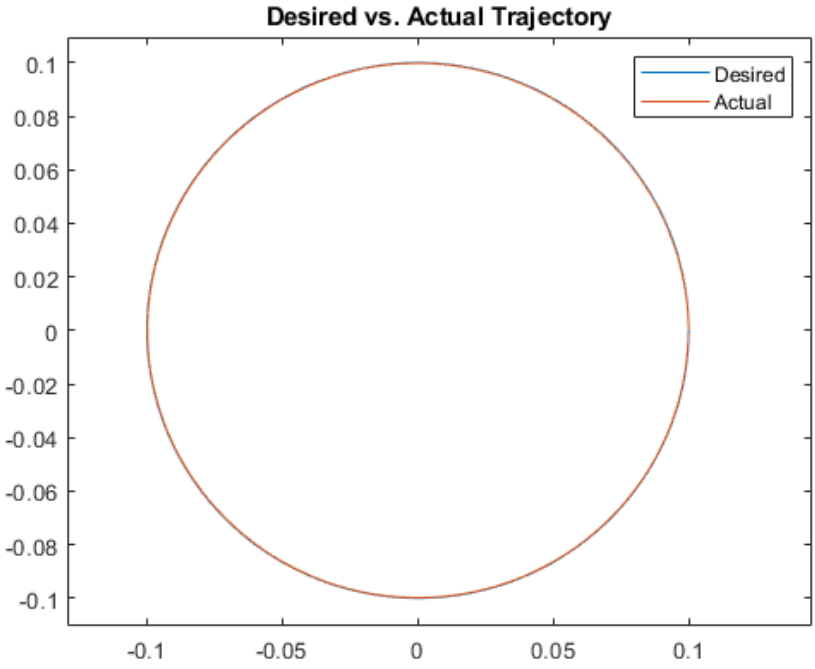

Figure 7.61 Circle 0.12 m/s^2 – Desired vs Actual Trajectory

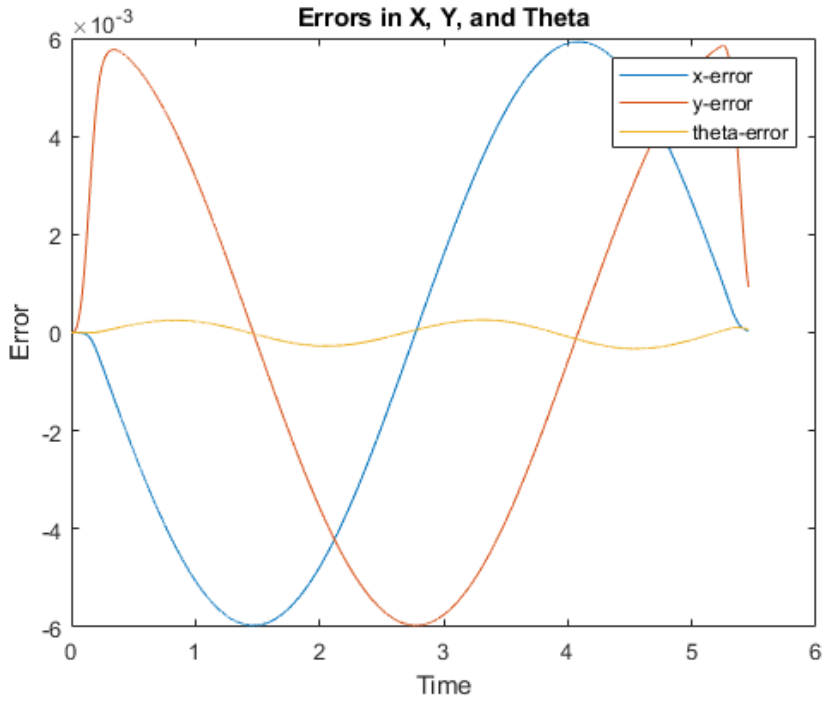

Figure 7.62 Circle 0.12 m/s^2 – Tracking Errors

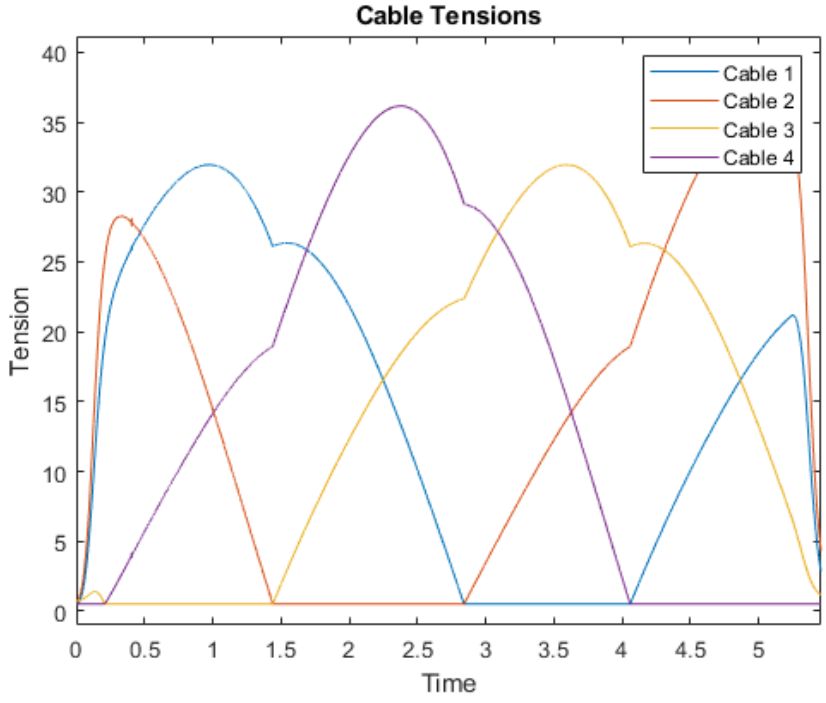

Figure 7.63 Circle 0.12 m/s^2 – Desired Tensions

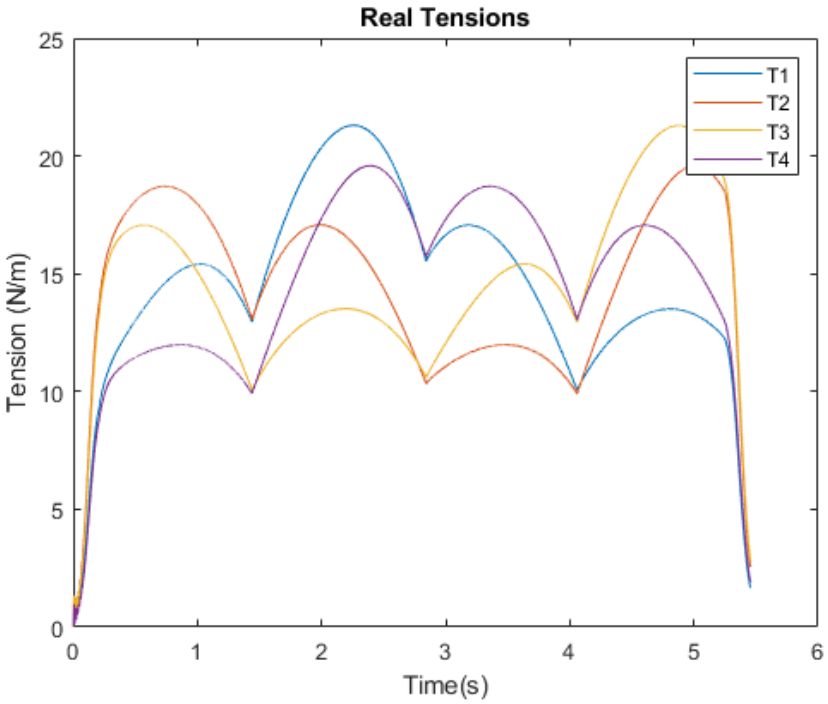

Figure 7.64 Circle 0.12 m/s^2 – Real Tensions

Circle Speed: 0.18m/s^2

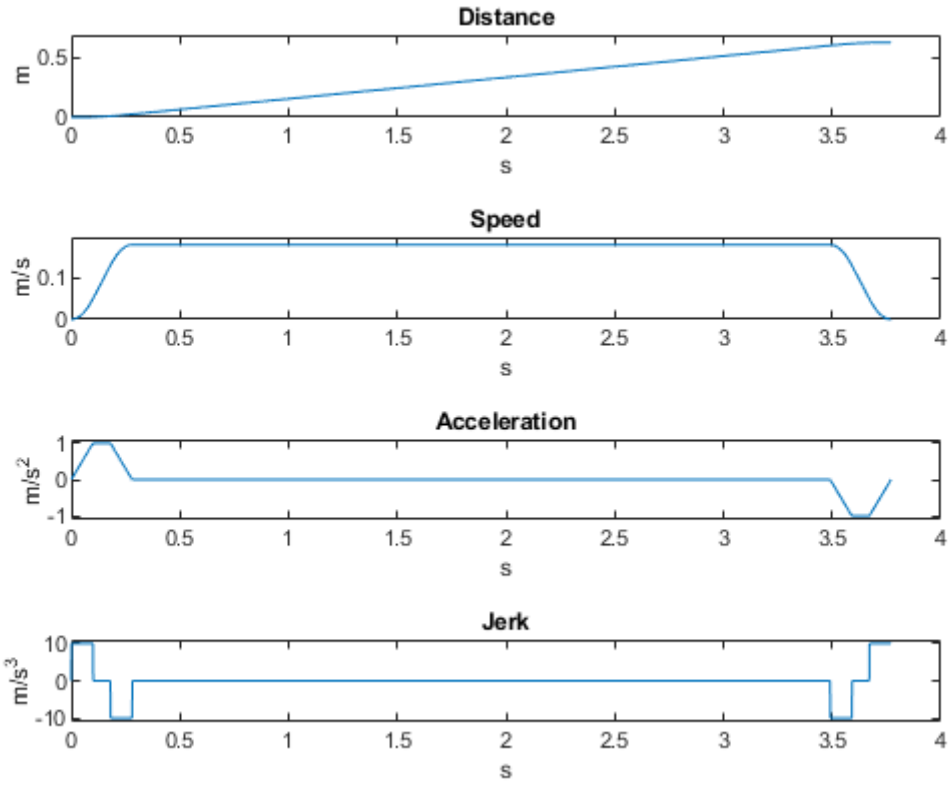

Figure 7.65 Circle 0.18 m/s^2 – Trapezoidal Acceleration

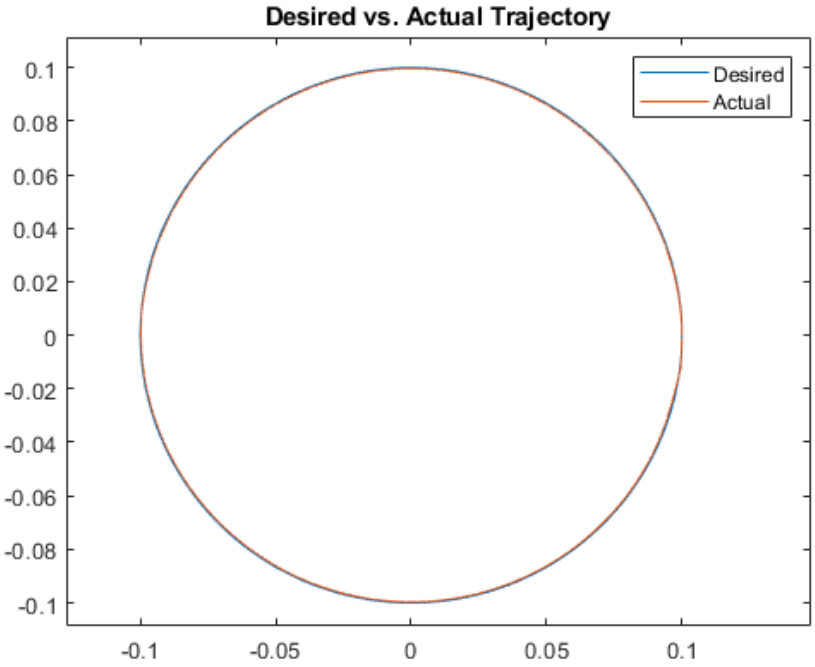

Figure 7.66 Circle 0.18 m/s^2 – Desired vs Actual Trajectory

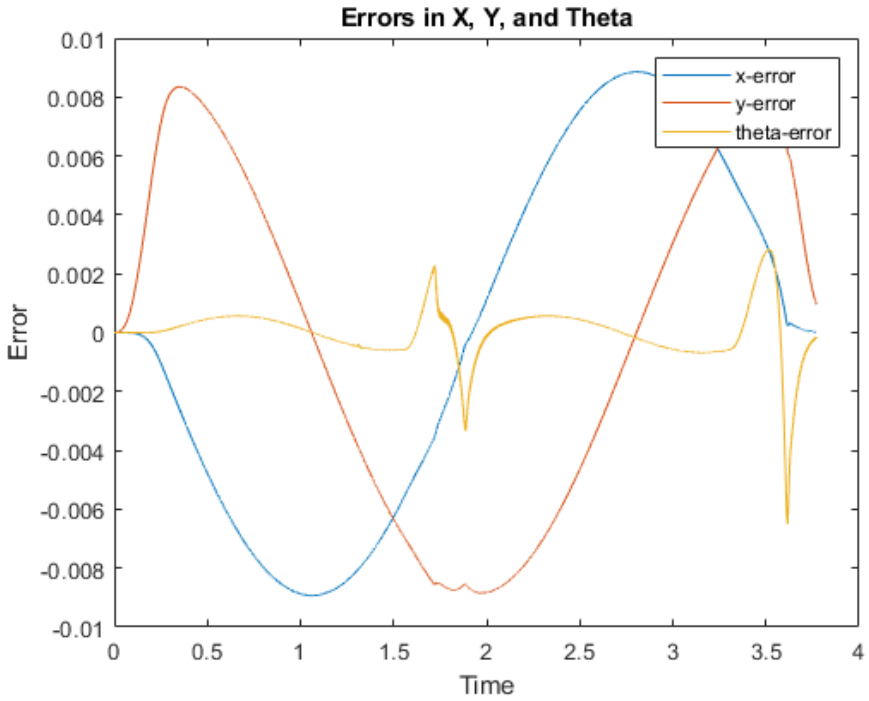

Figure 7.67 Circle 0.18 m/s^2 – Tracking Errors

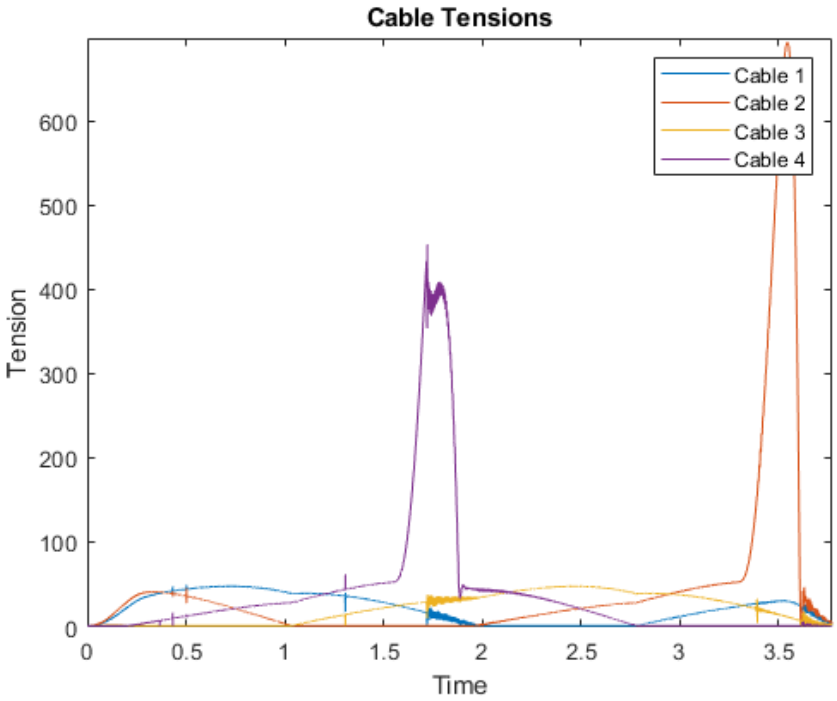

Figure 7.68 Circle 0.18 m/s^2 – Desired Tensions

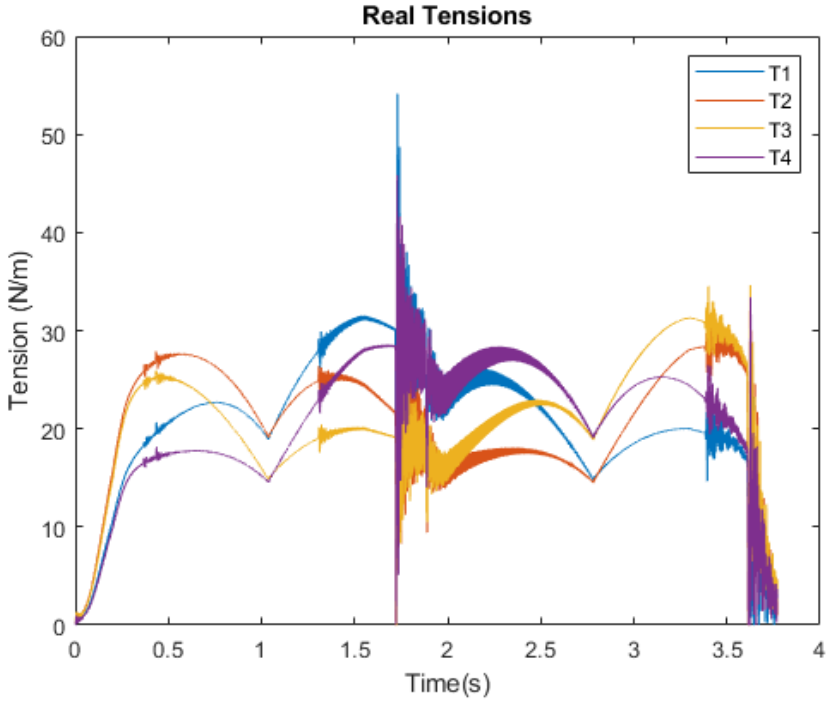

Figure 7.69 Circle 0.18 m/s^2 – Trapezoidal Acceleration

7.5.6. Simulations with Non-zero orientation angle

This series of simulations aim to show the performance of the cable robot when the reference angle for the end-effector. The circular trajectory with 0.06 m/s^2 velocity is chosen.

Angle: 5º

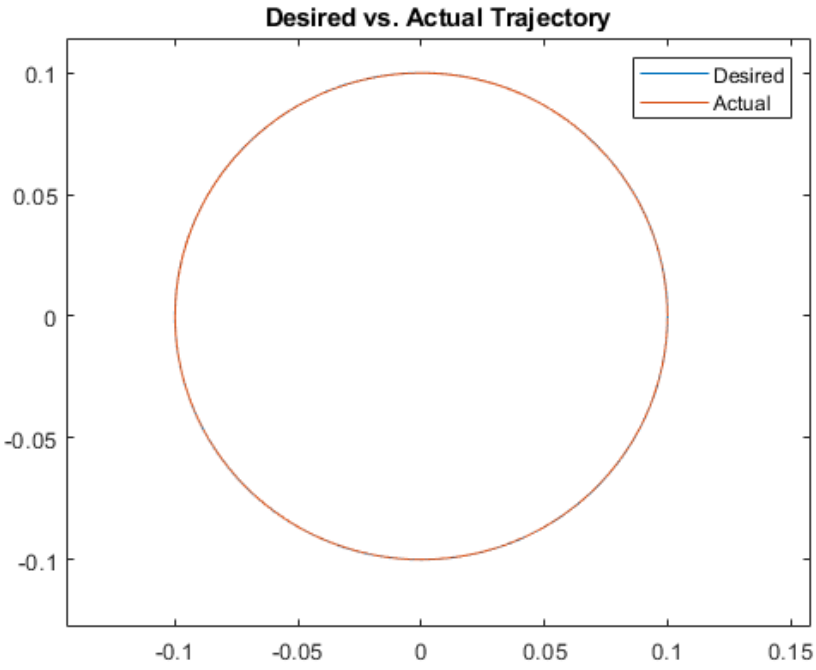

Figure 7.70 Circle – 5 degree – Desired vs Actual trajectory

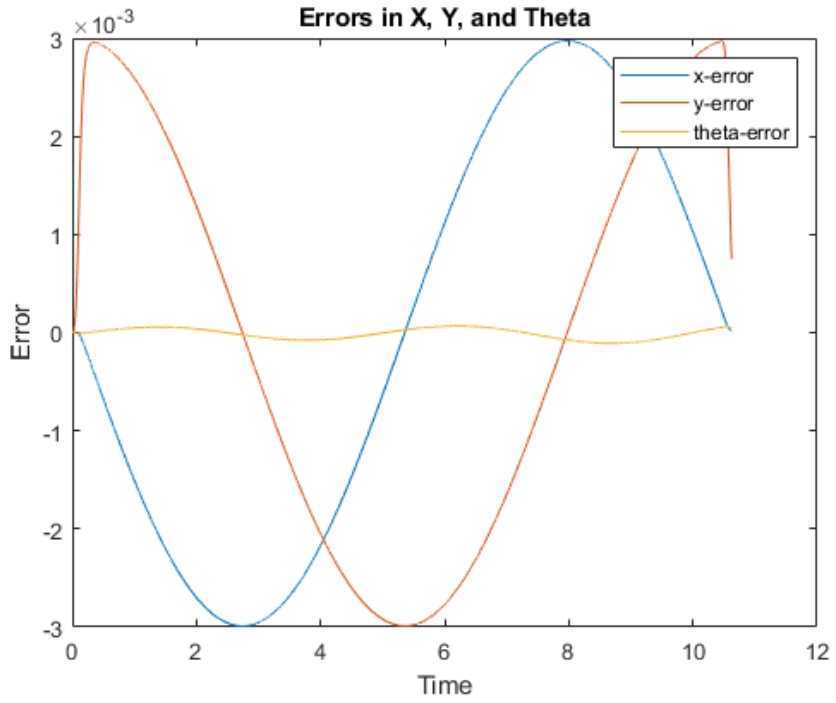

Figure 7.71 Circle – 5 degree – Tracking Errors

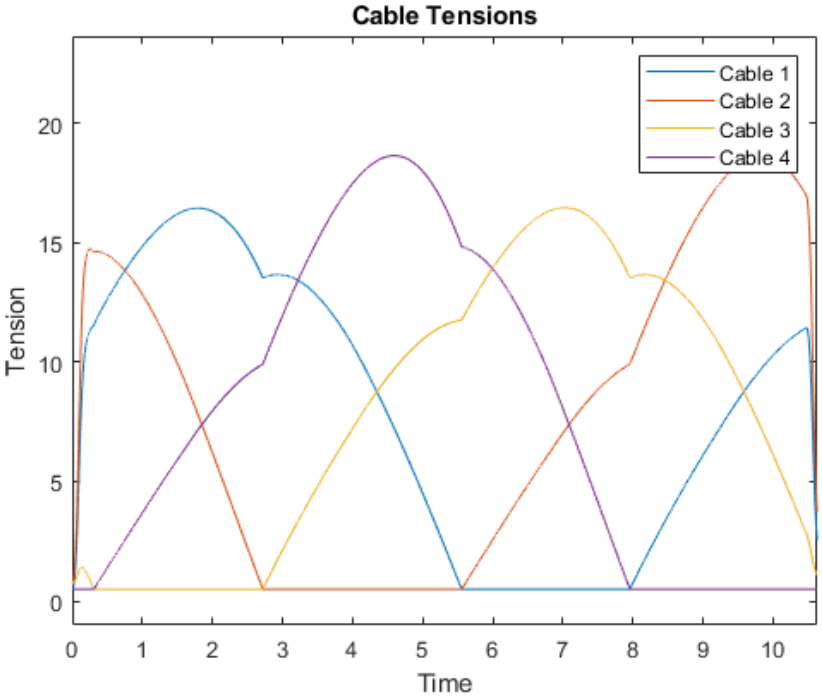

Figure 7.72 Circle – 5 degree – Desired Tensions

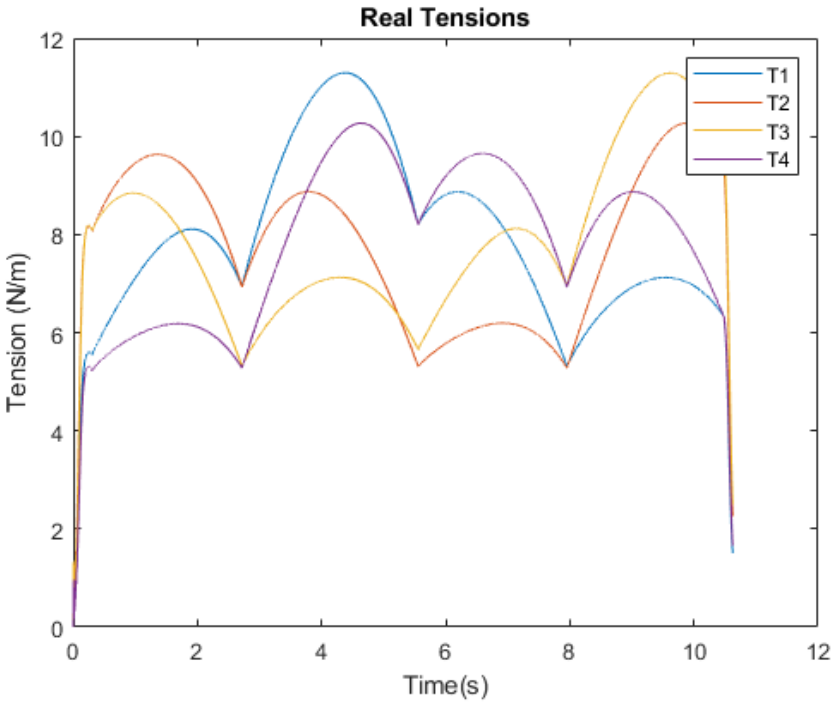

Figure 7.73 Circle – 5 degree –Actual Tensions

Angle: 15º

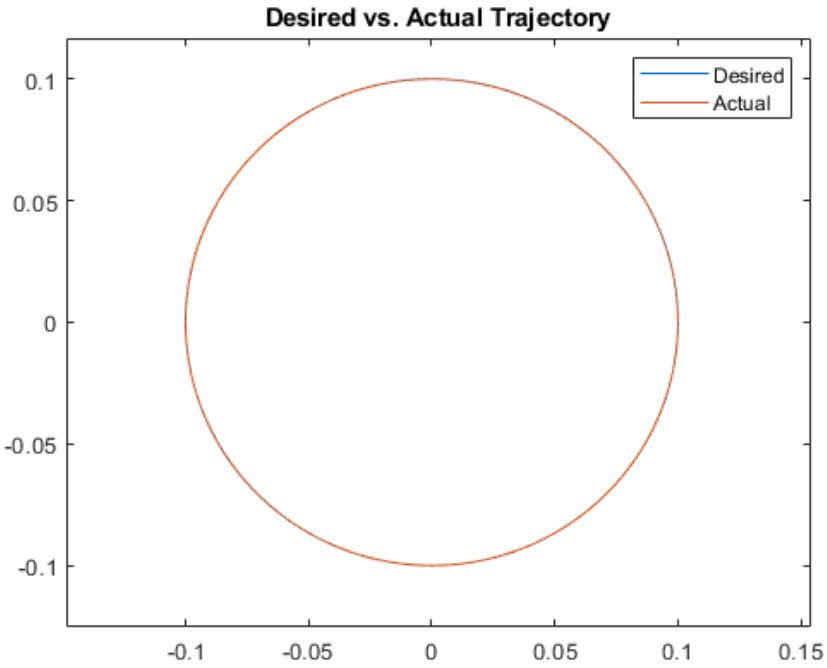

Figure 7.74 Circle – 15 degree – Desired vs Actual trajectory

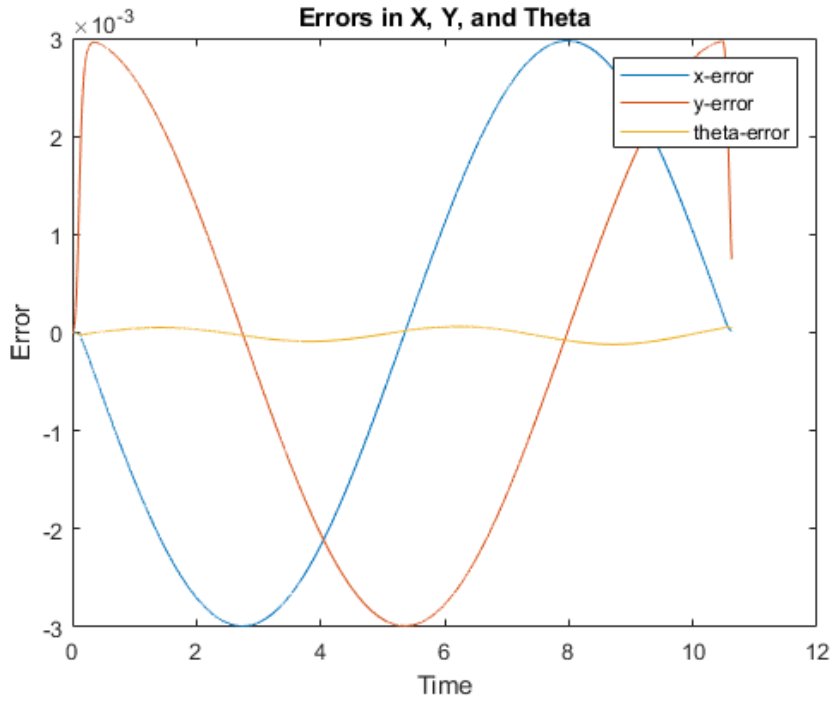

Figure 7.75 Circle – 15 degree – Tracking Errors

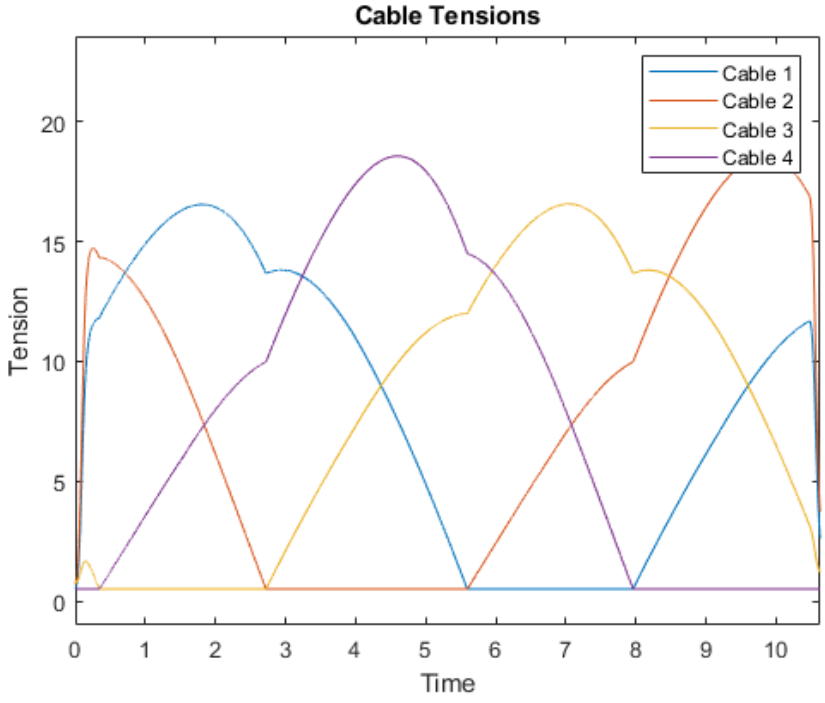

Figure 7.76 Circle – 15 degree – Tracking Errors

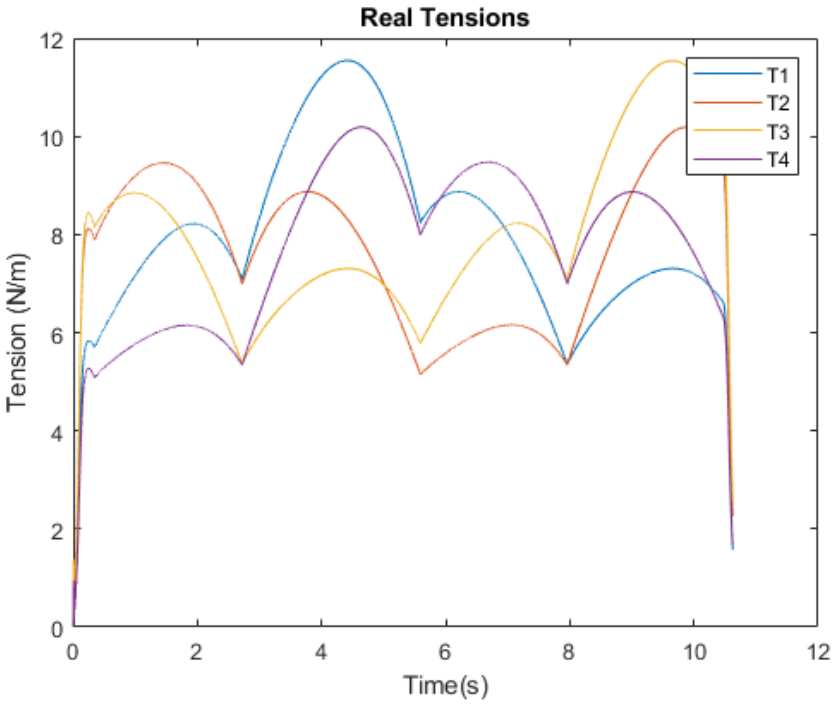

Figure 7.77 Circle – 15 degree – Real Tension

Angle: 30º

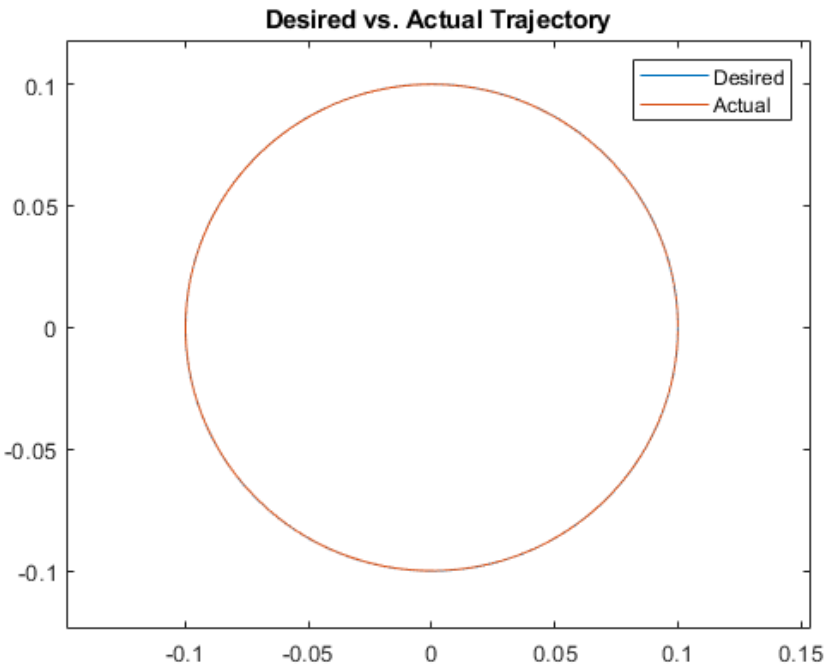

Figure 7.78 Circle – 30 degree – Desired vs Actual Trajectory

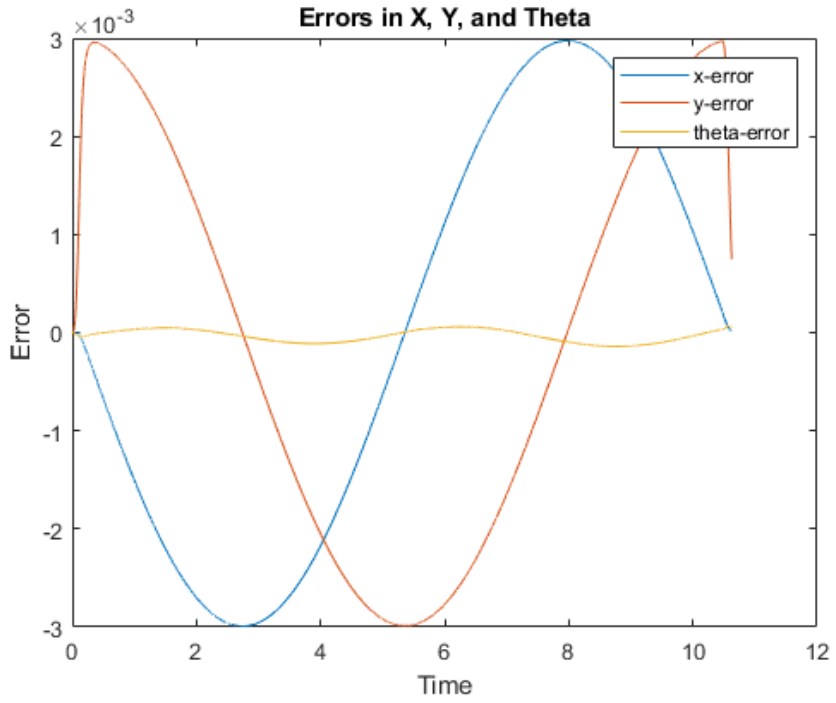

Figure 7.79 Circle – 30 degree – Tracking Errors

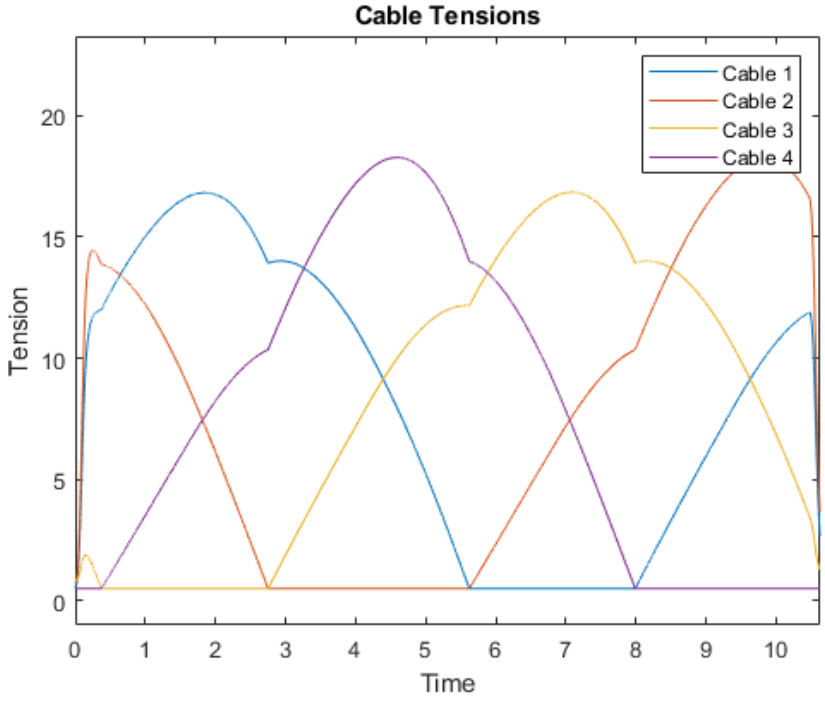

Figure 7.80 Circle – 30 degree – Desired Tensions

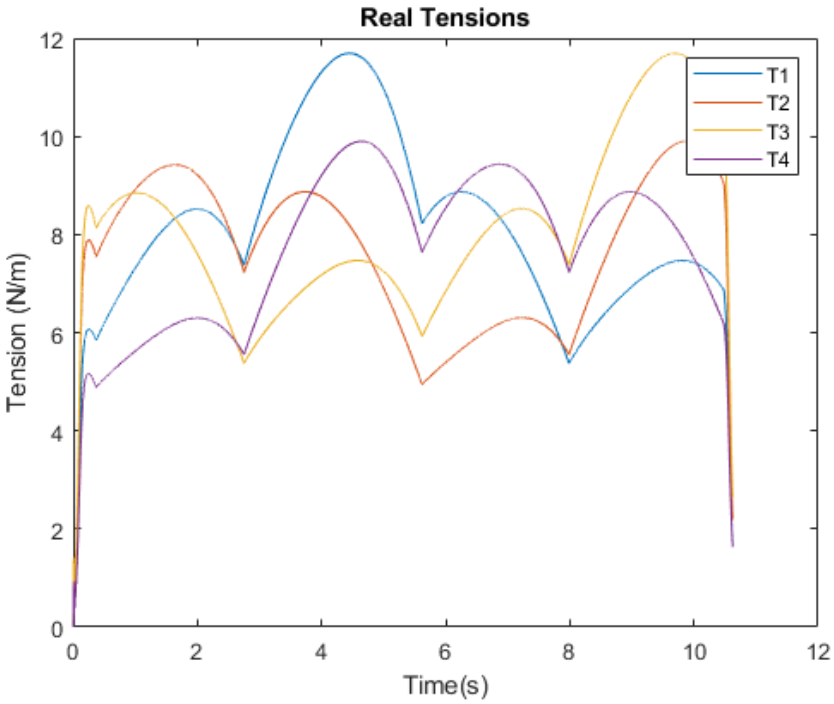

Figure 7.81 Circle – 30 degree – Real Tensions

Angle: 70º

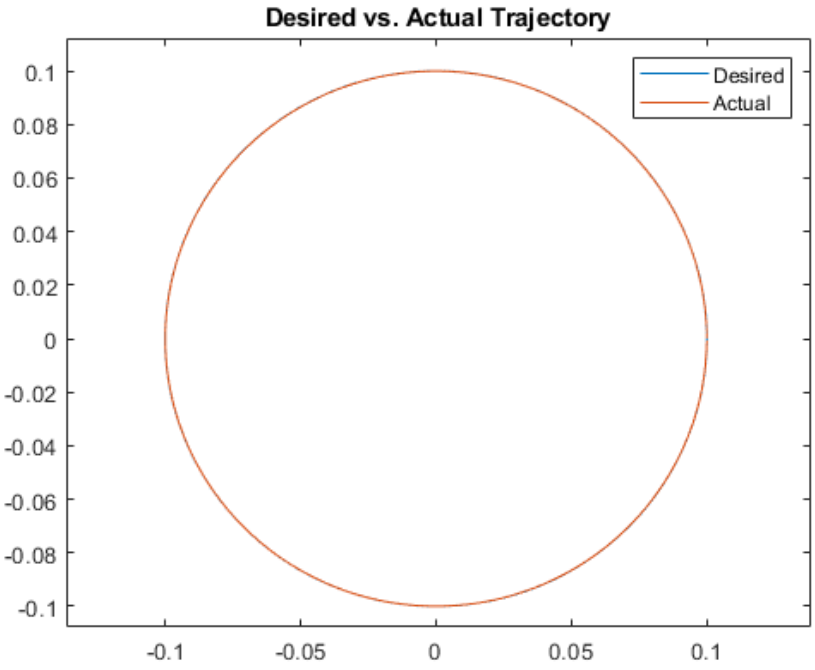

Figure 7.82 Circle – 70 degree – Desired vs Actual Trajectory

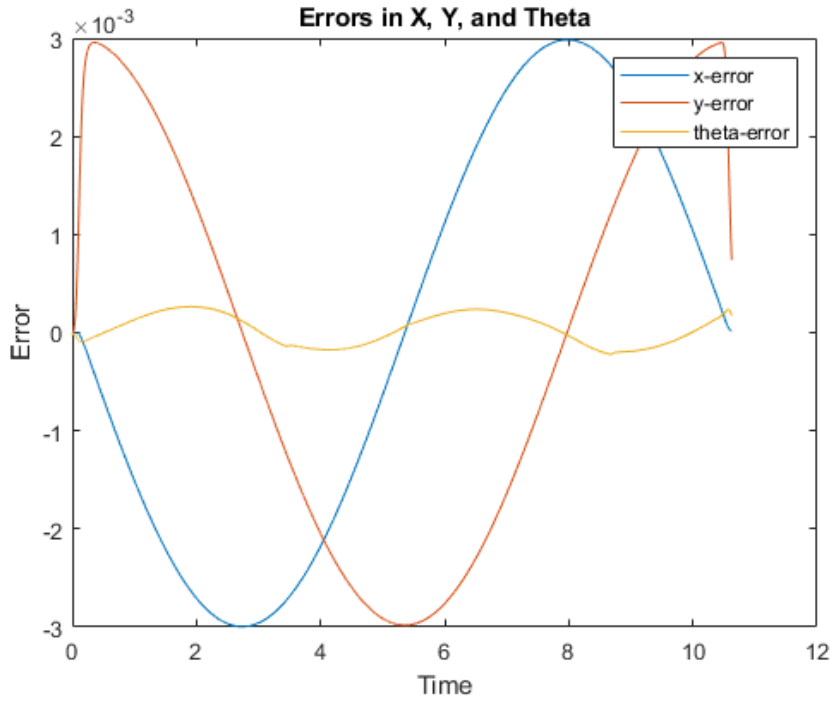

Figure 7.83 Circle – 70 degree – Tracking Errors

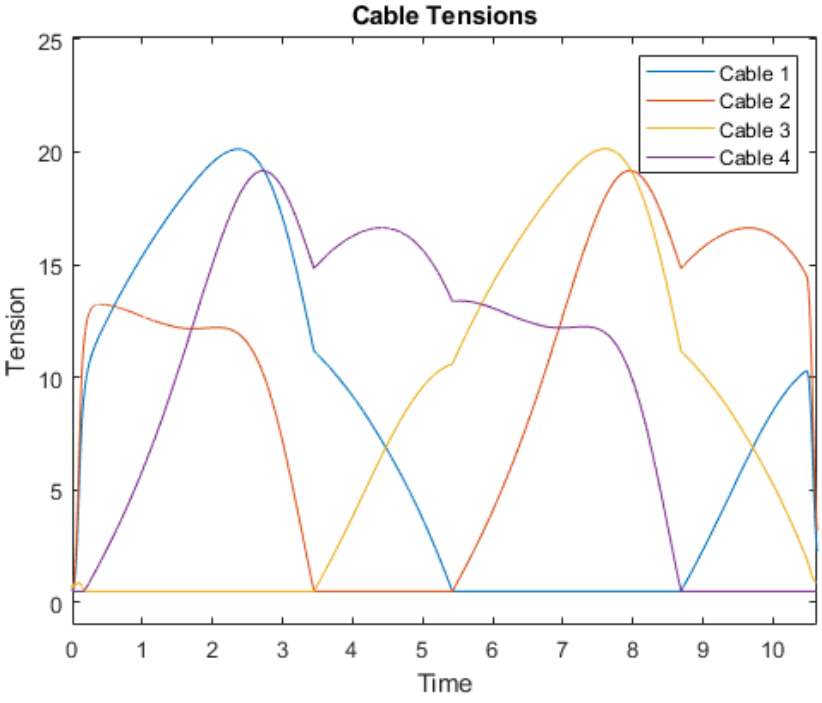

Figure 7.84 Circle – 70 degree – Desired Tensions

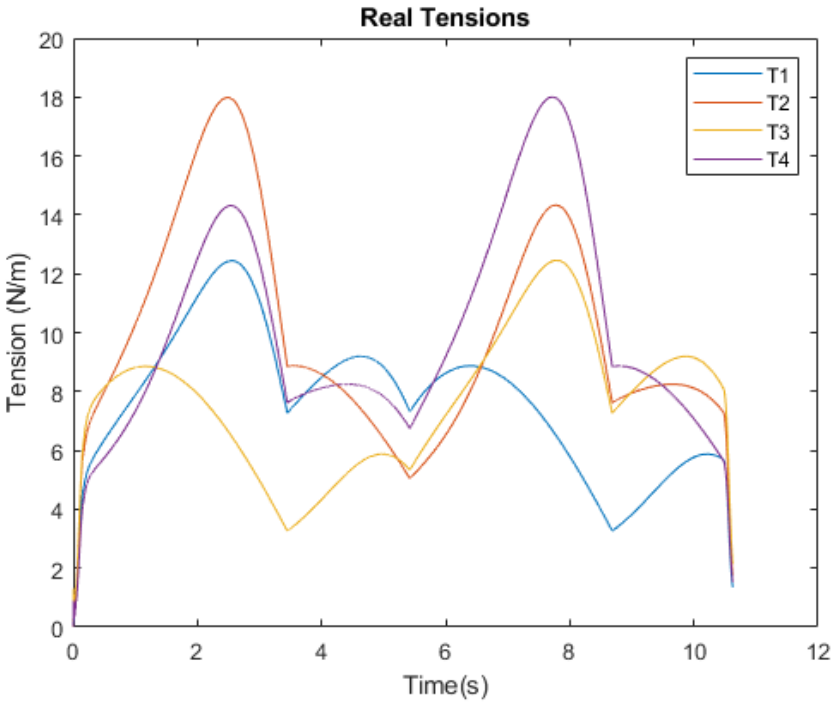

Figure 7.85 Circle – 70 degree – Real Tensions

At higher angles the cable robot is no longer stable.

7.5.7. Results with Non-Optimized Configuration

A non-optimized configuration is shown below in Figure 7.86.

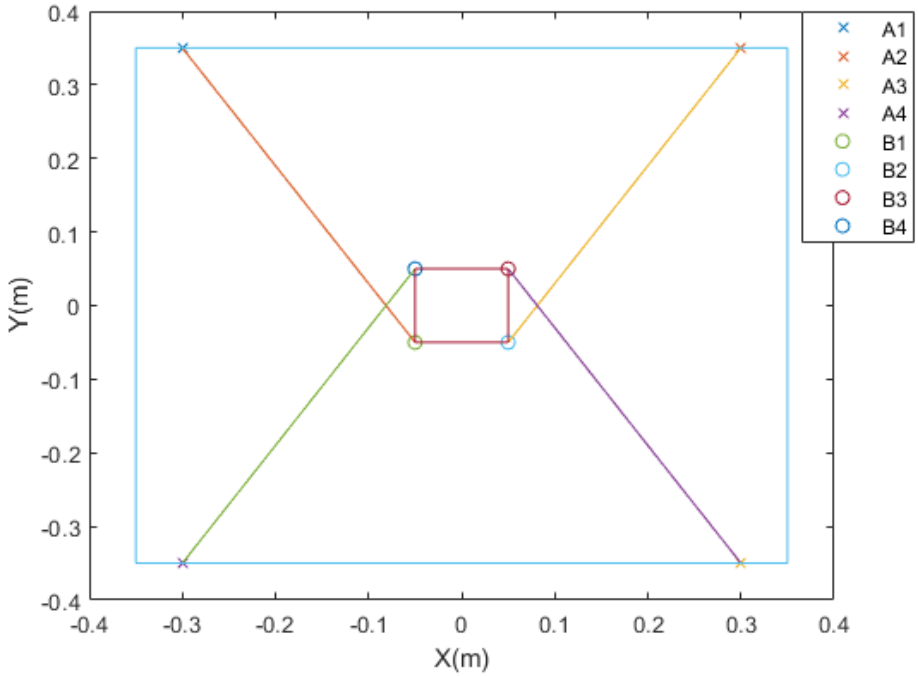

Figure 7.86 Non-optimized cable robot configurations

The simulation results for the circular trajectory at 0.18 m/s^2 velocity are given below.

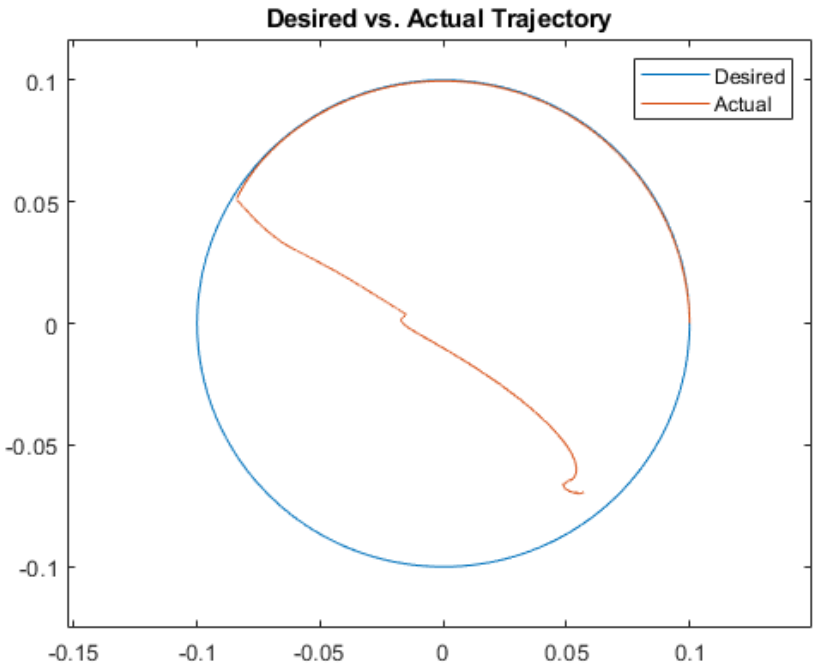

Figure 7.87 Circle – Non-Optimized – Desired vs Actual Trajectory

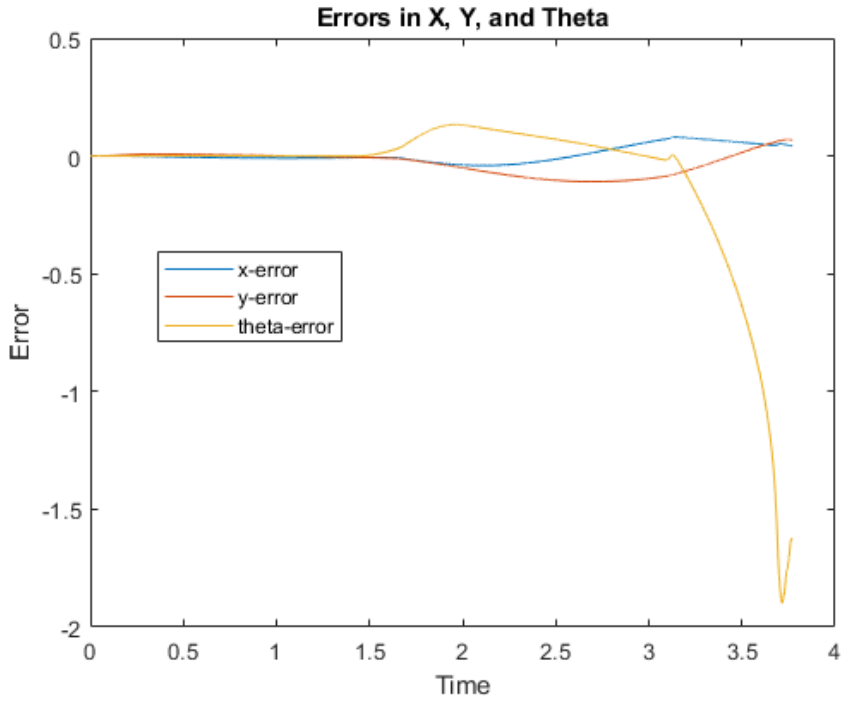

Figure 7.88 Circle – Non-Optimized – Tracking errors

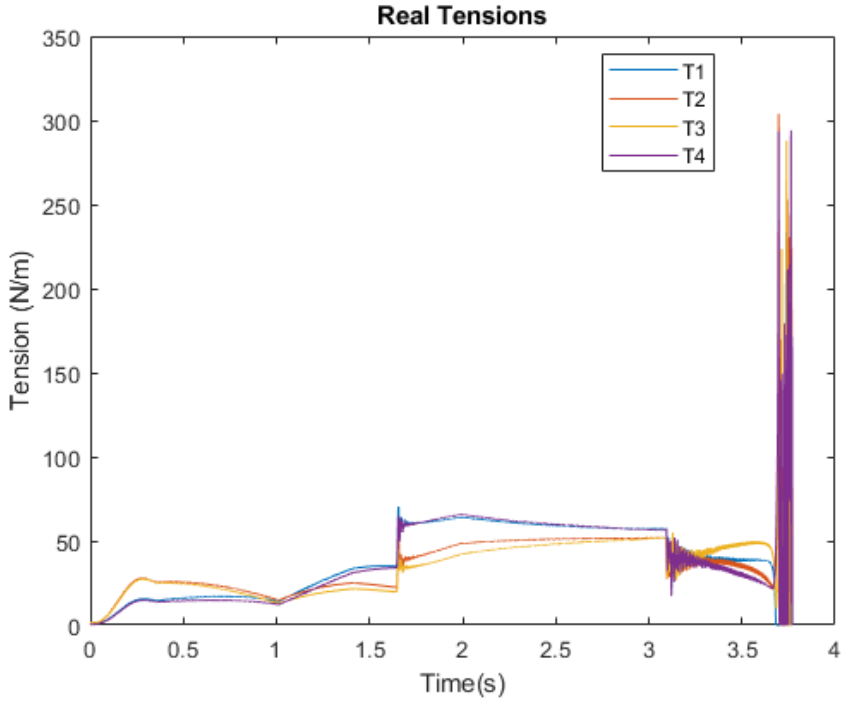

Figure 7.89 Circle – Non-Optimized – Real Tensions

With the unoptimized configuration, the cable robot is not stable enough for this trajectory.

7.6. Discussion

In section 7.5.1 the circular trajectory is generated using Trajectory Generation 1 algorithm which generates standard shapes using linear and circular interpolation methods. The circle trajectory is generated using the circular interpolation and the acceleration and velocity profiles are also shown for the generated trajectory. The control performance for trajectory tracking is good with very minimal error and the cable tensions are also within acceptable limits.

In section 7.5.2, a rectangular trajectory is generated consisting of linear segments joined together. Initially the trajectory is generated using a point-to-point approach where each segment begins and ends at zero velocity so there is no issue while cornering. The tracking error is very small in this simulation and the cable tensions are also small. The next simulation is carried out using the continuous approach where the corner point can have a set velocity. In this simulation, the tracking error is slightly larger as compared to the previous simulation and more deviation can be observed near the corners of the trajectory. The cable tensions are also larger. Lastly, the rectangular shape is generated using the Microsplines, which are used to join the ends of different segments using splines. The simulation then shows that the tracking error is very small, and the deviation from the path is minimal.

In the next part in section 7.5.3, Trajectory Generation 3 algorithm is used to design a non-standard shape trajectory, namely a spiral trajectory. The tracking performance is suitable. The cable tensions have minimal oscillations and are within an acceptable range. In section 7.5.4, Trajectory Generation 2 algorithm is used to develop a trajectory for some custom non-uniform data. For each scenario, the trajectory starts from a given start/end point, where it does a linear interpolation to the start of the first spline which is used to connect to the start of the custom data and then after the custom trajectory, another spline is used to come to a stop where another linear interpolation is used to get back to the start/end point. For each custom trajectory, A, B, and C, the tracking performance is good and the cable tensions are also within an acceptable value.

In section 7.5.5, simulations with increasing speed are carried out using the circular trajectory. With higher speeds we can observe the tracking error to increase. In section 7.5.6, the circular trajectory is tracked but with a non-zero reference theta (end-effector rotation angle) and the results show good performance until around 70 degrees, after which the robot becomes unstable at higher angles.

Lastly in section 7.5.7, the same circular trajectory is simulated using a non-optimized cable robot configuration which runs into some problems so the previous results which were obtained from the configuration after dexterity optimization improves the performance of the cable robot.

8. CONCLUSION AND RECOMMENDATIONS

Based on the simulations conducted, the different trajectory generation algorithms are successfully able to generate trajectories for the cable robot. Trajectory Algorithm 1 is able to generate standard shapes but it is also able to generate splines in regions where a corner exists or two segments are being joined so as to prevent and sudden jerk or loss of performance at that point. Trajectory Algorithm 3, similar to 1 is used for trajectories with uniform velocity profiles but it is capable of fitting a trajectory on any custom shape. It was used to simulate a spiral where the radius is constantly changing. Lastly, Trajectory Algorithm 2 is able to take a custom trajectory with non-uniform speed profile and simulate it by connecting it with splines. All these trajectory generation algorithms from CNC literature were successfully implemented with the cable robot model.

The cable robot model itself was a planar 4 cable robot with cable elasticity added as well. The model was able to simulate the different scenarios at different speeds and angles while maintaining positive cable tension which is essential for a cable robot. The structure of the cable robot was also optimized using the fmincon command and the results indicated an increase in performance compared to an initial condition after optimization. Other than that, the cascade controller also worked well and the design of it along with the other elements was incorporated into the GUI interface which can easily be used to completely design and simulate a planar cable robot.

This type of robot can be used in many different applications such as medical rehabilitation devices, 3D printers, warehouse crane mechanisms, etc. Future recommendations include the addition of different types of controllers which can be tuned for different scenarios. The capability of generating animations can be added to the GUI. The model can also be expanded to a 3-D cable robot which would be capable of much more advanced maneuvers.

The combination of the cable robot design and the different trajectory generation techniques can be particularly useful for medical rehabilitation devices. For providing robot-aided physiotherapy, patients are required to perform different repetitive exercises. Robots can greatly aid this process reducing the requirement of trained professionals being physically present during these exercises. Simpler exercises can use trajectories like

linear and circular interpolation from trajectory generation algorithm 1, with the addition of splines creating smooth trajectories which would be particularly useful for patients undergoing therapy. However more complex exercises can require trajectories with complex structures which can be much better handled by trajectory generation algorithm 3. Lastly, the trajectory generation algorithm 2 could be useful if a medical professional wanted to record a set of exercises. The coordinates recorded from those exercise could then be used to generate a custom trajectory with non-uniform speed and acceleration which would allow the patient to receive much more complicated tasks.

REFERENCES

[1] S.R. Oh, S.K. Agrawal, Cable suspended planar robots with redundant cables: Controllers with positive tensions, IEEE Trans. Robot. 21 (2005) 457–465. https://doi.org/10.1109/TRO.2004.838029.

[2] P. Dion-Gauvin, C. Gosselin, Trajectory planning for the static to dynamic transition of point-mass cable-suspended parallel mechanisms, Mech. Mach. Theory. 113 (2017) 158–178. https://doi.org/10.1016/J.MECHMACHTHEORY.2017.03.003.

[3] X. Tang, An Overview of the Development for Cable-Driven Parallel Manipulator, Https://Doi.Org/10.1155/2014/823028. 2014 (2014). https://doi.org/10.1155/2014/823028.

[4] B. Zi, S. Qian, Design, analysis and control of cable-suspended parallel robots and its applications, Des. Anal. Control Cable-Suspended Parallel Robot. Its Appl. (2017) 1–299. https://doi.org/10.1007/978-981-10-1753-7/COVER.

[5] J.S. Albus, R. V. Bostelman, N. Dagalakis, The NIST ROBOCRANE, 10, No. 5 (1992) 709–724. https://www.nist.gov/publications/nist-robocrane (accessed April 21, 2024).

[6] S. Qian, B. Zi, W.W. Shang, Q.S. Xu, A review on cable-driven parallel robots, Chinese J. Mech. Eng. (English Ed. 31 (2018) 1–11. https://doi.org/10.1186/S10033-018-0267-9/FIGURES/19.

[7] D. Huamanchahua, A. Tadeo-Gabriel, R. Chavez-Raraz, K. Serrano-Guzman, Parallel Robots in Rehabilitation and Assistance: A Systematic Review, 2021 IEEE 12th Annu. Ubiquitous Comput. Electron. Mob. Commun. Conf. UEMCON 2021. (2021) 692–698. https://doi.org/10.1109/UEMCON53757.2021.9666501.

[8] Q. Chen, B. Zi, Z. Sun, Y. Li, Q. Xu, Design and development of a new cable-driven parallel robot for waist rehabilitation, IEEE/ASME Trans. Mechatronics. 24 (2020) 1497–1507. https://doi.org/10.1109/TMECH.2019.2917294.

[9] H. Jia, W. Shang, F. Xie, B. Zhang, S. Cong, Second-Order Sliding-Mode-Based Synchronization Control of Cable-Driven Parallel Robots, IEEE/ASME Trans. Mechatronics. 25 (2020) 383–394. https://doi.org/10.1109/TMECH.2019.2960048.

[10] H.D. Taghirad, Parallel Robots: Mechanics and Control, Parallel Robot. Mech. Control. (2013) 1–510. https://doi.org/10.1201/B16096.

[11] M.I. Hosseini, S.A. Khalilpour, H.D. Taghirad, Practical robust nonlinear PD controller for cable-driven parallel manipulators, Nonlinear Dyn. 106 (2021) 405–424. https://doi.org/10.1007/S11071-021-06758-9/TABLES/3.

[12] W. Kraus, P. Miermeister, V. Schmidt, A. Pott, Hybrid Position-Force Control of a Cable-Driven Parallel Robot with Experimental Evaluation, Mech. Sci. 6 (2015) 119–125. https://doi.org/10.5194/MS-6-119-2015.

[13] J.C. Santos, A. Chemori, M. Gouttefarde, Model predictive control of large-dimension cable-driven parallel robots, Mech. Mach. Sci. 74 (2019) 221–232. https://doi.org/10.1007/978-3-030-20751-9_19/COVER.

[14] W. Shang, F. Xie, B. Zhang, S. Cong, Z. Li, Adaptive Cross-Coupled Control of Cable-Driven Parallel Robots with Model Uncertainties, IEEE Robot. Autom. Lett. 5 (2020) 4110–4117. https://doi.org/10.1109/LRA.2020.2988430.

[15] A. Pott, (PDF) WireX – An Open Source Initiative Scientific Software for Analysis and Design of Cable-driven Parallel Robots, (n.d.). https://www.researchgate.net/publication/334164626_WireX_-_An_Open_Source_Initiative_Scientific_Software_for_Analysis_and_Design_of_Cable-driven_Parallel_Robots (accessed February 20, 2023).

[16] M. Zarebidoki, J.S. Dhupia, W. Xu, A Review of Cable-Driven Parallel Robots: Typical Configurations, Analysis Techniques, and Control Methods, IEEE Robot. Autom. Mag. 29 (2022) 89–106. https://doi.org/10.1109/MRA.2021.3138387.

[17] D. Sridhar, R.L. Williams, Kinematics and Statics Including Cable Sag for Large Cable-Suspended Robots, Proc. ASME Des. Eng. Tech. Conf. 5A-2016 (2016). https://doi.org/10.1115/DETC2016-60495.

[18] S.W. Hwang, J.H. Bak, J. Yoon, J.H. Park, J.O. Park, Trajectory generation to suppress oscillations in under-constrained cable-driven parallel robots, J. Mech. Sci. Technol. 30 (2016) 5689–5697. https://doi.org/10.1007/S12206-016-1139-9/METRICS.

[19] L. Kevac, M. Filipovic, A. Rakic, The trajectory generation algorithm for the cable-suspended parallel robot—The CPR Trajectory Solver, Rob. Auton. Syst. 94 (2017) 25–33. https://doi.org/10.1016/J.ROBOT.2017.04.018.

[20] X. Jiang, C. Gosselin, Trajectory generation for three-degree-of-freedom cable-suspended parallel robots based on analytical integration of the dynamic equations, J. Mech. Robot. 8 (2016). https://doi.org/10.1115/1.4031501/384152.

[21] M.A. Khosravi, H.D. Taghirad, R. Oftadeh, A positive tensions PID controller for a planar cable robot: An experimental study, Int. Conf. Robot. Mechatronics, ICRoM 2013. (2013) 325–330. https://doi.org/10.1109/ICROM.2013.6510127.

[22] P. Gholami, M.M. Aref, H. Taghirad, Adaptive Cascade Control of the KNTU CDRPM: A Cable Driven Redundant Parallel Manipulator, (2009).

[23] S.A. Khalilpour, R. Khorrambakht, M.J. Harandi, H.D. Taghirad, P. Cardou, Cascade Terminal Sliding Mode Control of a Deployable Cable Driven Robot, Proc. - 2019 6th Int. Conf. Control. Instrum. Autom. ICCIA 2019. (2019). https://doi.org/10.1109/ICCIA49288.2019.9030886.

[24] A. Fattah, S.K. Agrawal, On the Design of Cable-Suspended Planar Parallel Robots, J. Mech. Des. 127 (2005) 1021–1028. https://doi.org/10.1115/1.1903001.

[25] S. Seriani, M. Seriani, P. Gallina, Workspace optimization for a planar cable-suspended direct-driven robot, Robot. Comput. Integr. Manuf. 34 (2015) 1–7. https://doi.org/10.1016/J.RCIM.2015.01.004.

[26] M.H. Kassem, Design and Optimization of a Planar Cable Robot, (2020). https://dspace.aus.edu:8443/xmlui/handle/11073/19730 (accessed May 22, 2024).

[27] T. Rasheed, P. Long, A.S. Roos, S. Caro, Optimization based Trajectory Planning of Mobile Cable-Driven Parallel Robots, IEEE Int. Conf. Intell. Robot. Syst. (2019) 6788–6793. https://doi.org/10.1109/IROS40897.2019.8968133.

[28] C. Sun, H. Gao, Z. Liu, S. Xiang, H. Yu, N. Li, Z. Deng, Design and optimization of three-degree-of-freedom planar adaptive cable-driven parallel robots using the cable wrapping phenomenon, Mech. Mach. Theory. 166 (2021) 104475. https://doi.org/10.1016/J.MECHMACHTHEORY.2021.104475.

[29] S. Abdolshah, D. Zanotto, G. Rosati, S.K. Agrawal, Optimizing stiffness and dexterity of planar adaptive cable-driven parallel robots, J. Mech. Robot. 9 (2017). https://doi.org/10.1115/1.4035681/472662.

[30] M. Anson, A. Alamdari, V. Krovi, Orientation Workspace and Stiffness Optimization of Cable-Driven Parallel Manipulators With Base Mobility, J. Mech. Robot. 9 (2017). https://doi.org/10.1115/1.4035988/472687.

[31] B. Zhou, S. Li, B. Zi, B. Chen, W. Zhu, Multi-Objective Optimal Design of a Cable-Driven Parallel Robot Based on an Adaptive Adjustment Inertia Weight Particle Swarm Optimization Algorithm, J. Mech. Des. 145 (2023). https://doi.org/10.1115/1.4062458.

[32] T. Bruckmann, R. Boumann, Simulation and optimization of automated masonry construction using cable robots, Adv. Eng. Informatics. 50 (2021) 101388. https://doi.org/10.1016/J.AEI.2021.101388.

[33] H.H. Cheng, D. Lau, Cable Attachment Optimization for Reconfigurable Cable-Driven Parallel Robots Based on Various Workspace Conditions, IEEE Trans. Robot. 39 (2023) 3759–3775. https://doi.org/10.1109/TRO.2023.3288838.

[34] W.B. Lim, S.H. Yeo, G. Yang, S.K. Mustafa, Kinematic analysis and design optimization of a cable-driven universal joint module, IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, AIM. (2009) 1933–1938. https://doi.org/10.1109/AIM.2009.5229772.

[35] R.L. Williams, P. Gallina, Translational planar cable-direct-driven robots, J. Intell. Robot. Syst. Theory Appl. 37 (2003) 69–96. https://doi.org/10.1023/A:1023975507009.

[36] K. Erkorkmaz, Y. Altintas, High speed CNC system design. Part I: Jerk limited trajectory generation and quintic spline interpolation, Int. J. Mach. Tools Manuf. 41 (2001) 1323–1345. https://doi.org/10.1016/S0890-6955(01)00002-5.

[37] J. Bolboli, M.A. Khosravi, F. Abdollahi, Stiffness feasible workspace of cable-driven parallel robots with application to optimal design of a planar cable robot, Rob. Auton. Syst. 114 (2019) 19–28. https://doi.org/10.1016/J.ROBOT.2019.01.012.

[38] S. Torres-Mendez, A. Khajepour, Design Optimization of a Warehousing Cable-Based Robot, Proc. ASME Des. Eng. Tech. Conf. 5A (2015). https://doi.org/10.1115/DETC2014-34672.

APPENDICES

APPENDIX 1 –GRAPHICAL USER INTERFACE

This chapter describes the development of the GUI tool the cable robot codes have been integrated into. The interface is explained and the layout is provided. The first tab of the GUI is given below which is used to enter parameters related to the motor, cable, end-effector and other parameters. After entering the parameters, the button 'Update Variables' is pressed.

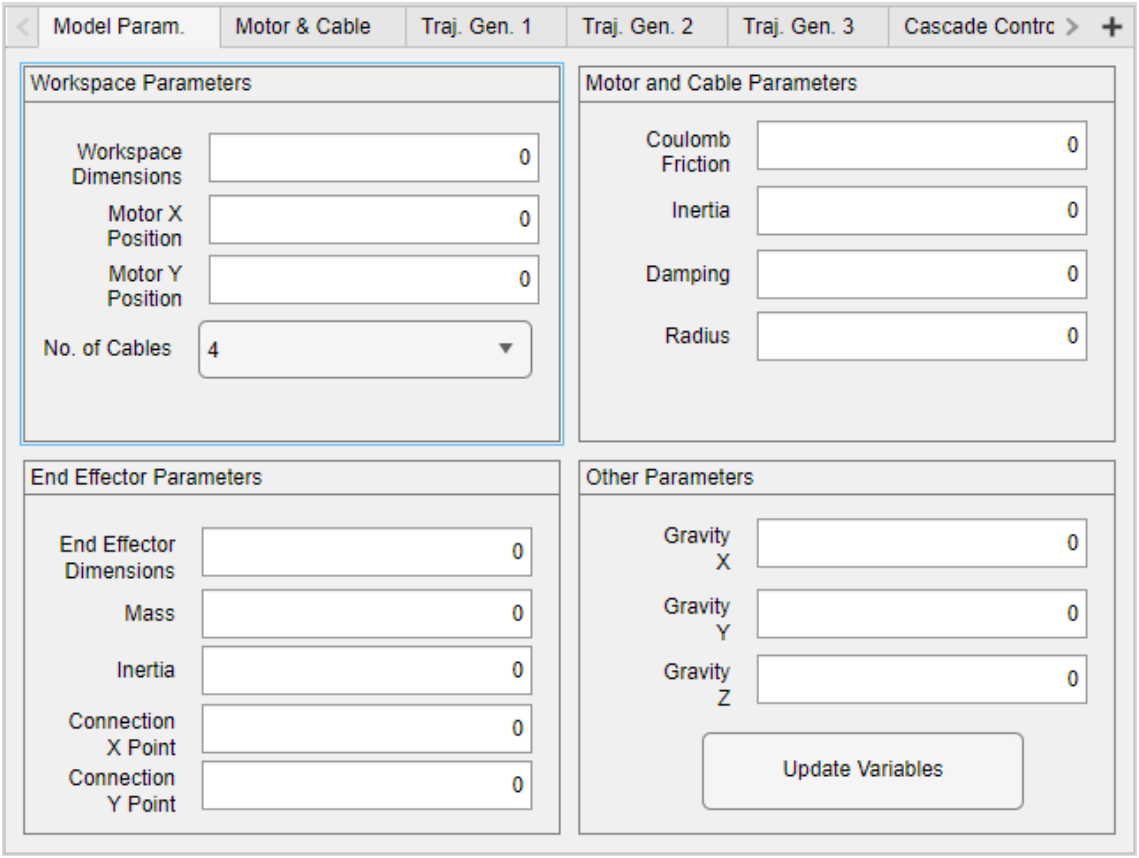

Figure Appendix 1.1 GUI Tab 1

The second tab of the GUI is given below. Additional parameters related to the motor and cable are entered. Point data for the motor torque-speed curve is also entered which is plotted. After data is entered, the 'Update Variables' button must be pressed.

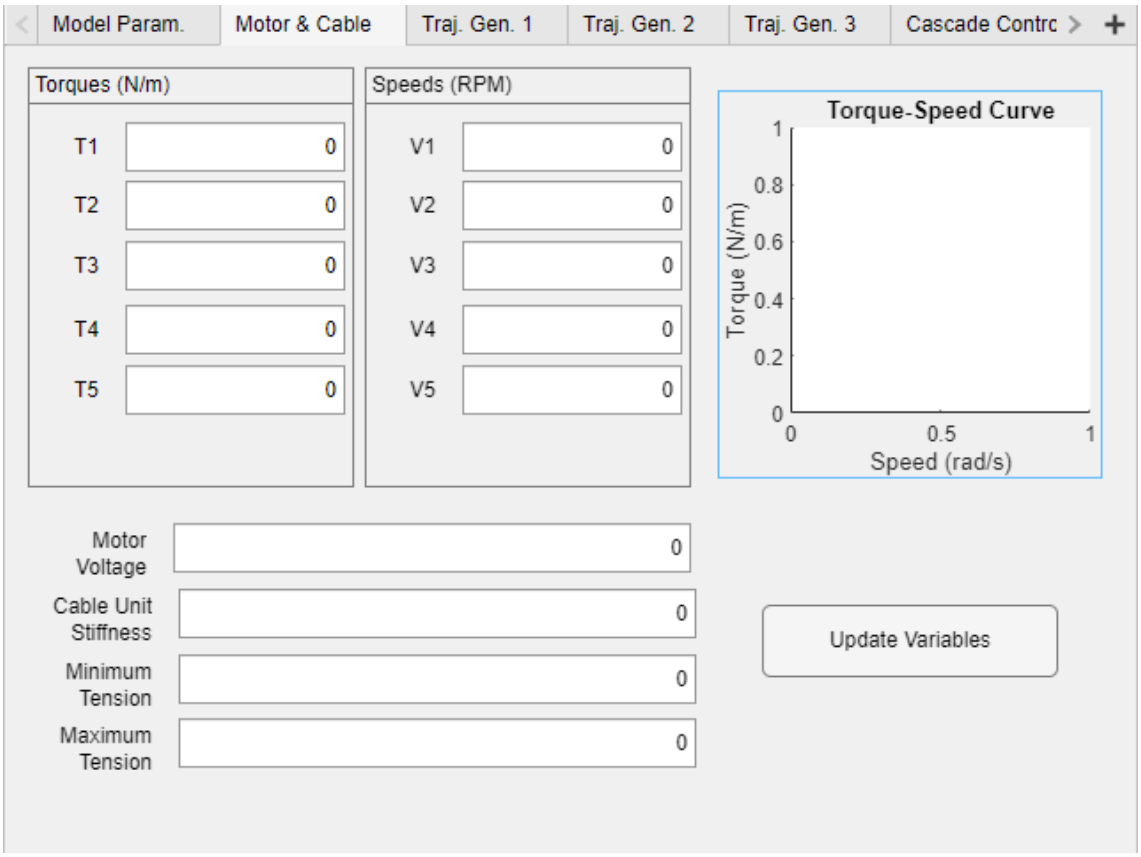

Figure Appendix 1.2 GUI Tab 2

The third tab of the application is for Trajectory Generation Algorithm 1. Here the desired trajectory can be selected, with the desired velocity and scale (for scaling the predefined dimensions) and when the ‘Generate’ button is pressed the desired trajectory is generated.

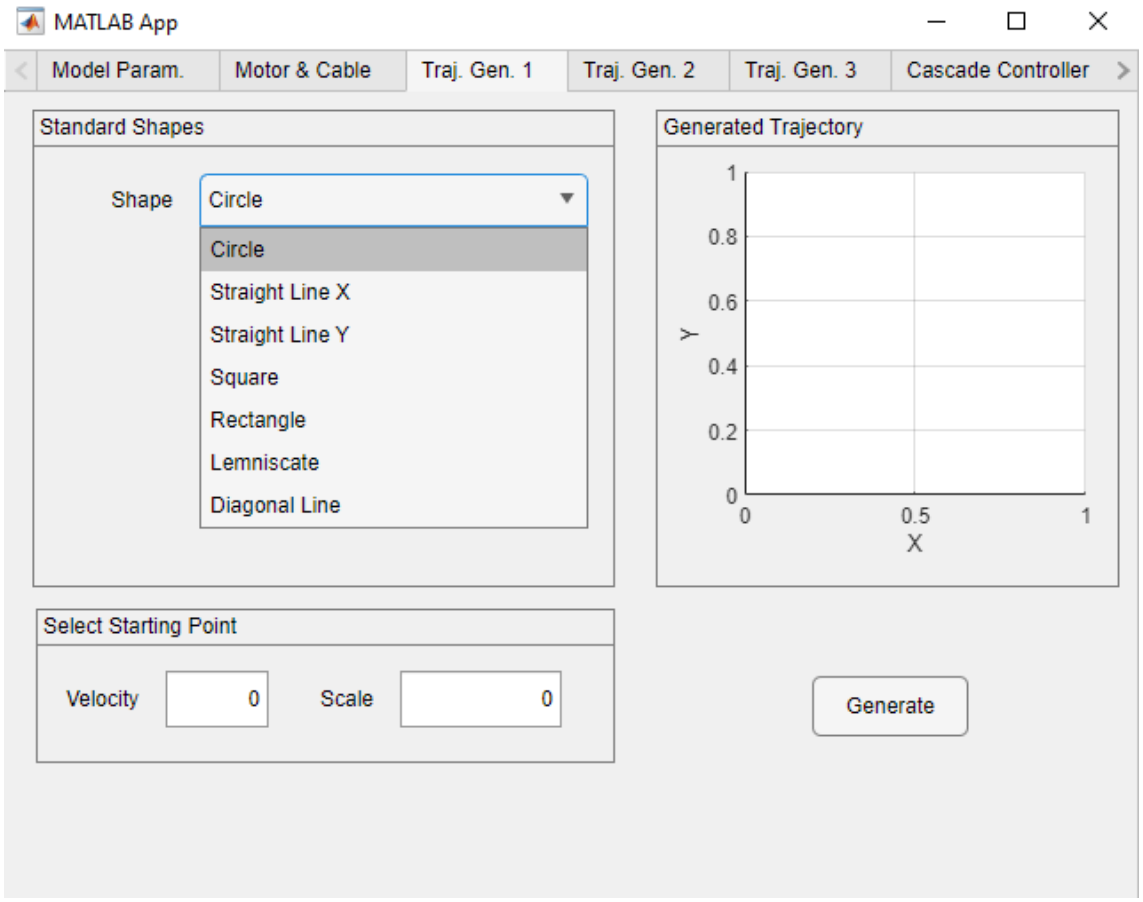

Figure Appendix 1.3 GUI Tab 3

In the fourth tab of the application, Trajectory Generation Algorithm 2 has been implemented. A .mat file containing the custom data can be loaded using the ‘Load Data’ and after specifying the proper scale, sampling time, and the start/end position, the desired trajectory is generated and displayed when the ‘Generate’ button is pressed.

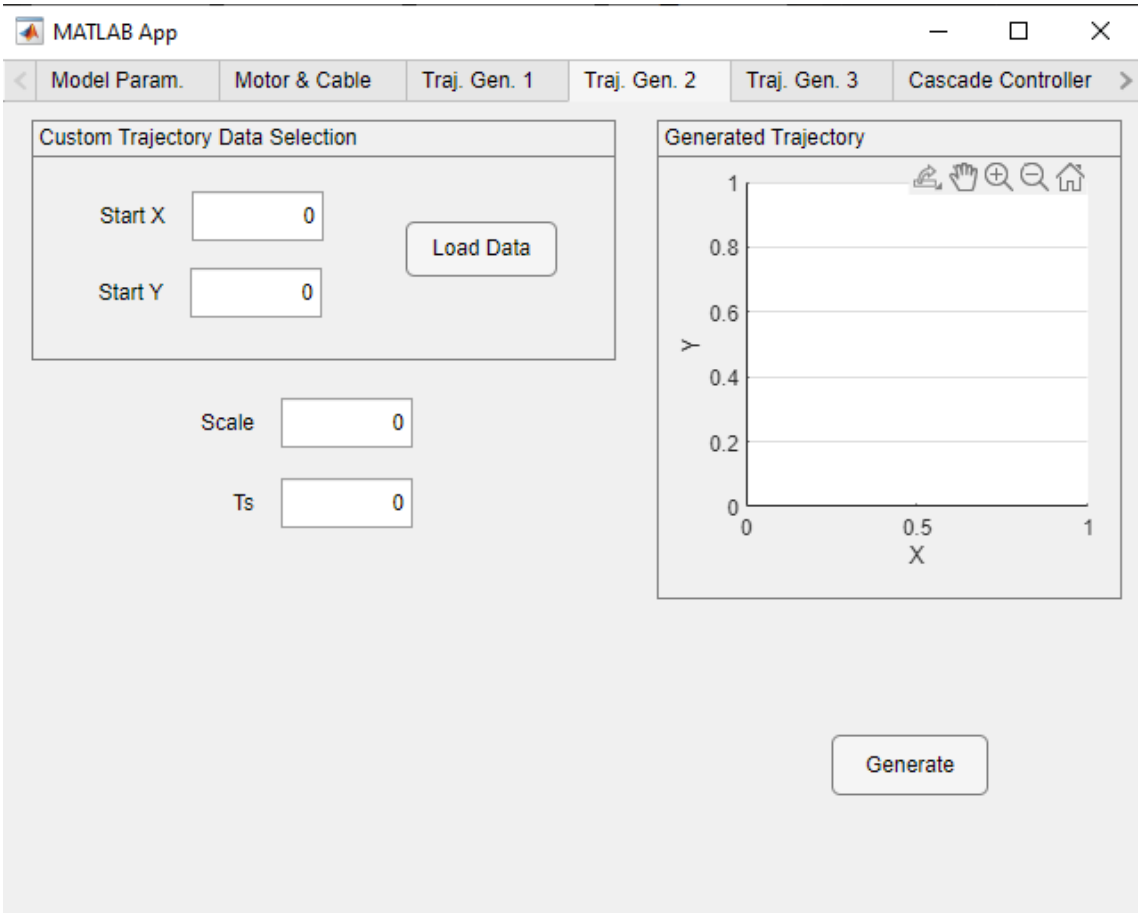

Figure Appendix 1.4 GUI Tab 4

The fifth tab contains the Trajectory Generation Algorithm 3. Here the desired parameters for the trapezoidal acceleration profile are entered and the ‘Load Data’ button is used to load a .mat file with the custom data for trajectory generation.

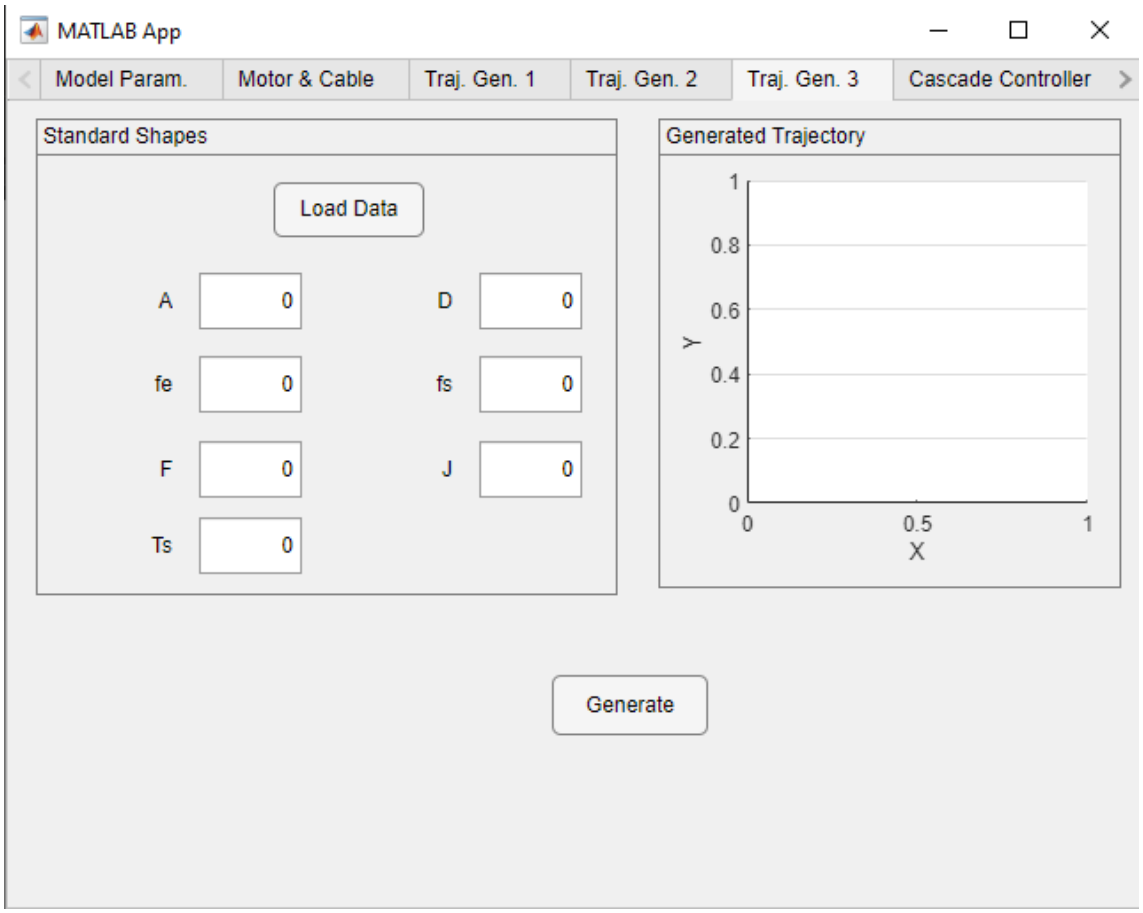

Figure Appendix 1.5 GUI Tab 5

Now the sixth tab is used for controller design. The required cascade controller can be automatically calculated based on the settling time and maximum overshoot specified by the user and the step response is plotted as well.

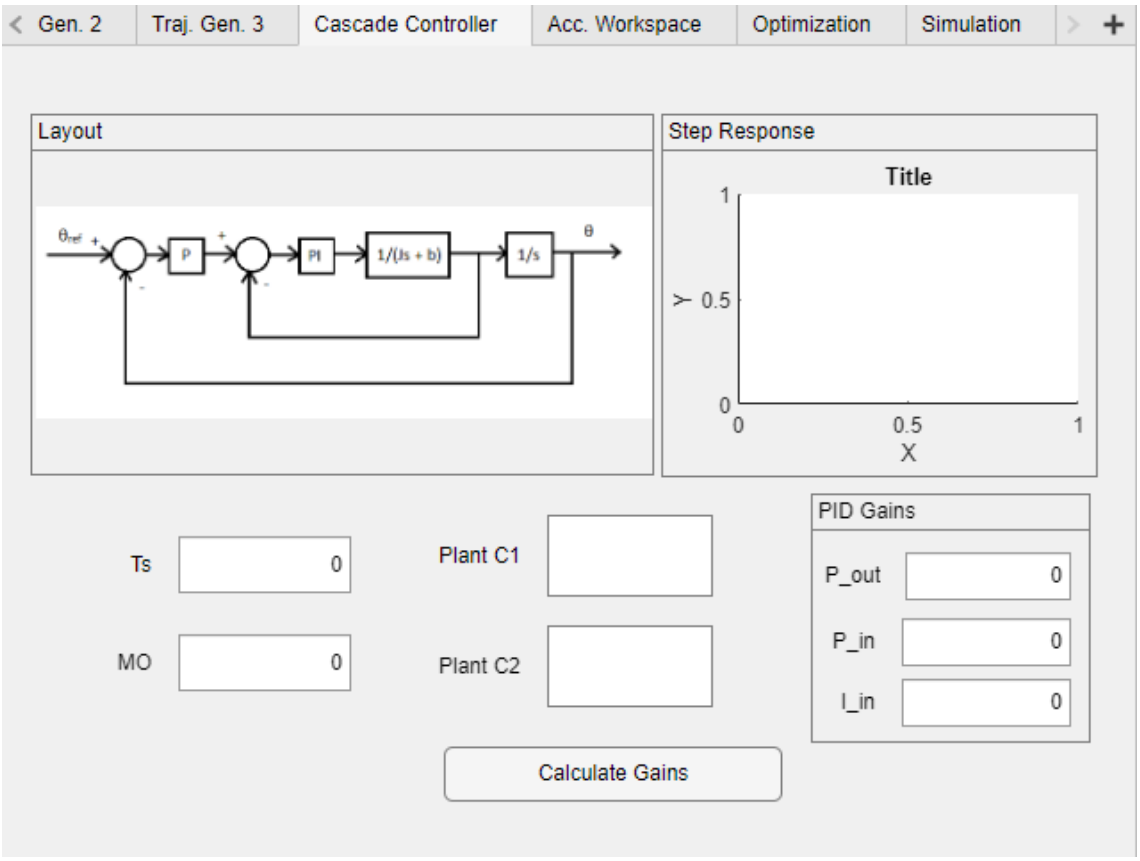

Figure Appendix 1.6 GUI Tab 6

The seventh tab shows the feasible workspace based on the end-effector acceleration and the minimum tensions. It also calculates the static torques.

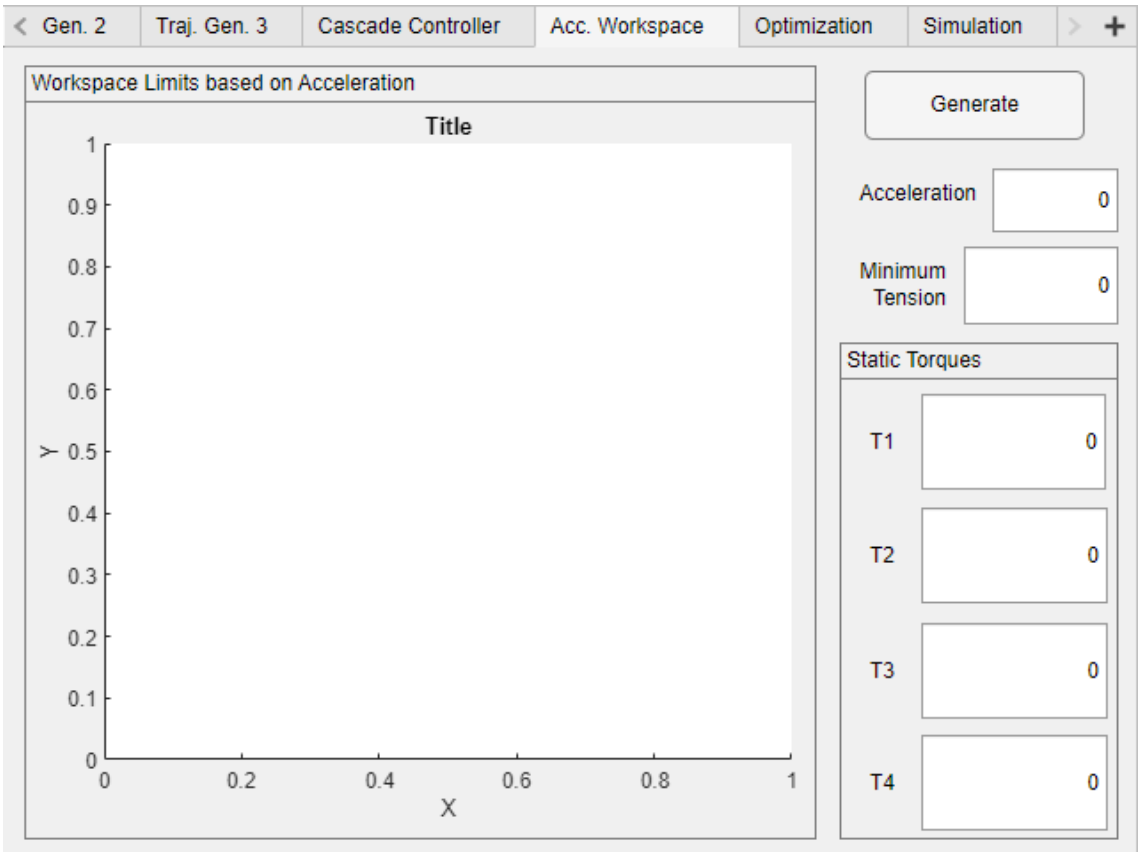

Figure Appendix 1.7 GUI Tab 7

The eighth tab contains the optimization functionality. Individual or combined optimization can be performed. The resultant motor and cable connection coordinates are displayed as well as the value of each cost function after any optimization when the ‘Generate’ button is pressed.

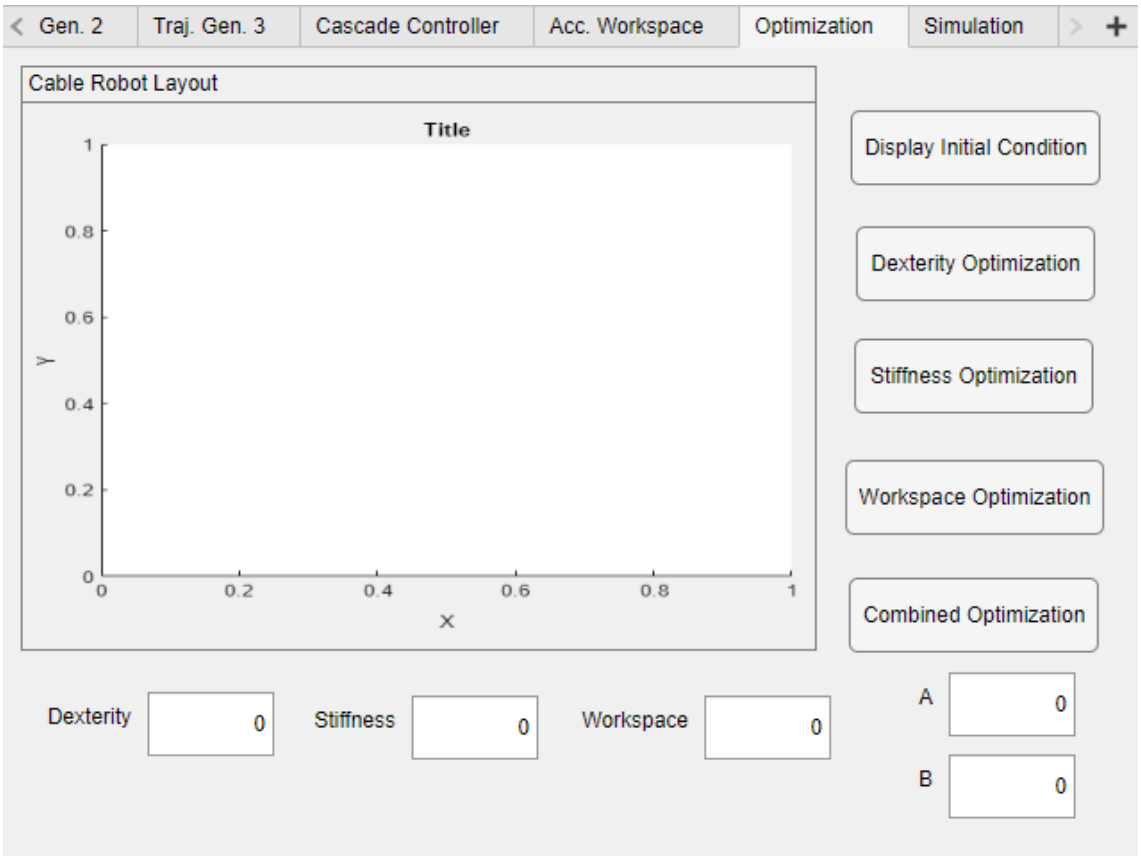

Figure Appendix 1.8 GUI Tab 8

The last and 9th tab is used to run the simulations when all the data has been entered, controller has been designed and necessary optimization is done. Additional buttons can be used to save the data file, open the Simulink model or generate detailed plots when the ‘Generate Plots’ button is pressed (after running the simulation at least once).

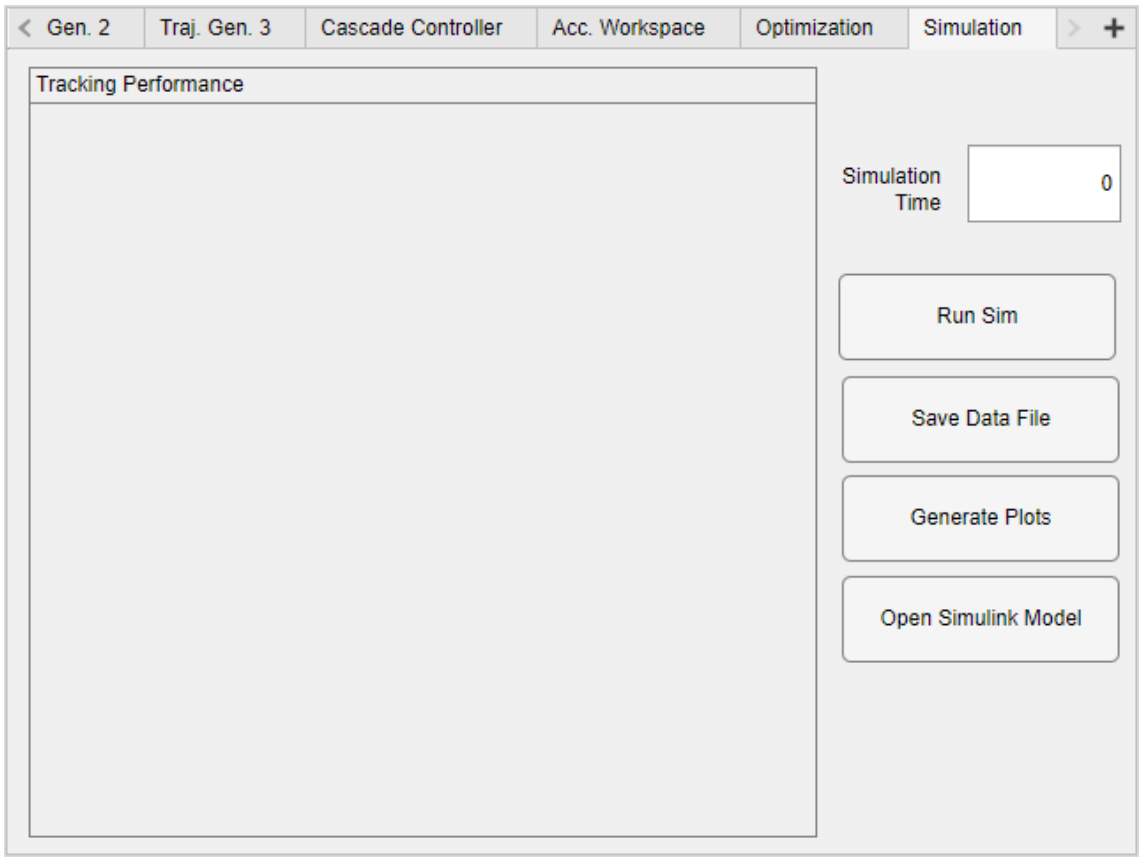

Figure Appendix 1.11 GUI Tab 11