



BEng Project Final Report

Dynamic Modelling of a Continuum Robotic Snake-arm and its Performance Evaluation by Analysing Robustness

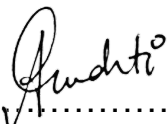
Student Name:	Arundathi Shaji Shanthini
Contact e-mail:	arundathi.shanthini.16@ucl.ac.uk
Student number:	16018351
Project Supervisor:	Prof. Sarah Spurgeon
Contact e-mail:	s.spurgeon@ucl.ac.uk
Department:	Department of Electronic and Electrical Engineering
Submission Date:	21 st of April 2020

DECLARATION

I have read and understood the College and Department's statements and guidelines concerning plagiarism.

I declare that all material described in this report is all my own work except where explicitly and individually indicated in the text. This includes ideas described in the text, figures and computer programs.

Name: Arundathi Shaji Shanthini

Signature: 

Date: 21.04.2020

Contents

1	Introduction	1
1.1	Project Description and Motivations	3
1.2	Literature Review	5
1.3	Aims and Objectives	6
2	Mathematical Modelling	8
2.1	Understanding the system	8
2.2	Model Preliminaries (Structural Design Description)	9
2.3	Kinematic Model	11
2.3.1	Motor Angle of Rotation - Cable Length Kinematics .	11
2.3.2	Cable Length - Joint Angle Kinematics	12
2.3.3	Joint Angle - End-Effector Position Kinematics . . .	16
2.4	Dynamic Model	18
2.4.1	Force Analysis	19
2.4.2	The Inverse Dynamic Model	23
2.4.3	The Forward Dynamic Model	26
3	Implementation and Simulation	27
3.1	Proposed Control Scheme	27
3.2	Implementation on Simulink	29
3.2.1	Dynamic Parameters	29
3.2.2	Reference Mapping (Kinematic Model)	30

3.2.3	Controller Block	31
3.2.4	Feedforward Block	32
3.2.5	Actuator Block (Maxon Motor)	33
3.2.6	Plant Block (Dynamic Model)	34
4	Results and Analysis	36
4.1	Cross-verification of Block Results	36
4.1.1	Verifying the Cable Length and Joint Angle Kinematic Model	37
4.1.2	Verifying the End-effector Position and Joint Angle Kinematic Model	37
4.1.3	Verifying the Inverse Dynamics Block	38
4.2	Tuning the Controller with the Actuator Model	39
4.3	Tuning the controller with the Forward Dynamic Block	41
4.4	Analysis of Robustness	47
5	Conclusions	48
5.1	Future Work	49
	Appendix A Project Gantt Chart (as submitted with proposal)	51
	Appendix B List of Notations	54
	Appendix C Supporting Explanation to Some Derivations	58
C.1	Derivation of Equation (2.5)	58
C.2	Derivation of Equation (3.6) and (3.7)	59
	Appendix D Simulink Implementation	60
D.1	The Proposed Control Scheme	60
D.2	Kinematic Model	61
D.2.1	Cable Length to Motor Angle	61
D.2.2	Cable Length \leftrightarrow Joint Angle	61

D.2.3	End-effector position \leftrightarrow Joint Angle	62
D.3	Dynamic Model (Forward Dynamics)	63
D.4	Feedforward Block (Inward Dynamics)	64
D.5	Actuator Block	64
D.5.1	Inside the Actuator Block	64
D.5.2	Modelling the Maxon Motor	65
D.5.3	Converting Speed to Torque	65
D.5.4	Tuning the Actuator	65
Appendix E	MATLAB Code	66
E.1	Parameter Definition	66
E.2	Kinematic Model: Cable Length - Joint Angles	67
E.2.1	Cable Length to Joint Angles	67
E.2.2	Joint Angles to Cable Length	70
E.3	Kinematic Model: Joint Angles - End-effector position Mapping	72
E.3.1	Joint Angles to End-effector position	72
E.3.2	End-effector position to Joint Angles	73
E.4	Dynamic Model	74
E.4.1	Calculating Contact Forces	74
E.4.2	Newton-Euler Recursive Algorithm	78

Abstract

Cable-driven continuum robotic arms are widely used because of its high accessibility and manoeuvrability properties. It also has the ability to perform dexterous in-situ tasks even in confined and hazardous spaces. However, for applications involving imaging/measurement tasks, precise trajectory tracking of the end-effector is essential. This project involves modelling this robotic manipulator and developing a potentially robust control configuration that ensures very low tracking error. For this, the kinematic and dynamic model of the system was developed, cross-validated and implemented in Simulink. Then, a baseline control system design was proposed. However, due to time constraints, only implementation of the the proposed control scheme could be achieved. Improving the results from the controller for the dynamic model and the analysis of its robustness remains work to be undertaken further.

Chapter 1

Introduction

Industrial robots that perform repetitive dexterous tasks have been around for quite some time. Over the years, these industrial robots have developed from being just command following, space-occupying large devices to systems that have better manoeuvrability, intelligence and accessibility properties, enabling it to perform a myriad of operations. Expanding a robot's taskspace by improving the manoeuvrability of a robot has been of great interest and over the past decade a lot of interest has grown towards bio-mimetic or bio-inspired systems to achieve this.

Bio-inspired robotic systems implement concepts of locomotion inspired from nature onto robotic systems. This involves studying concepts of locomotion from nature and implementing that into robotic systems. One such kind of bio-inspired system is a snakebot. The locomotion achieved by snakes is highly desirable because of its accessibility and manoeuvrability properties which means that snakes can access confined spaces and manoeuvre through different kinds of terrain. The undulatory motion of snakes enables them to traverse through terrains where legged robots usually fail. Additionally, it is known that some snake gaits makes it possible for them to swim, while some snakes can even fly [1]. Such manoeuvrability of snakes have generated a lot of interest in wanting to study the different gaits of a snake and developing a mathematical model so that the motion can be mimicked. Successful prototypes of such robots have been proven to be very useful in a wide range of applications such as disaster management (it can collect data and images from inaccessible areas at search and rescue sites), industrial inspection - like subsea operations, aircraft wing inspection, military reconnaissance etc. Fig. 1.1 shows some of the different kinds of snakebots that have been developed over the years by different

research groups around the world. Some of these are amphibious snakebots which means that they can operate in both on land and under water.



Figure 1.1: This picture shows some of the various kinds of snakebots that has been developed over the years. It shows examples of wheeled, wheel-less and amphibious snakebots that are used for varied operations. The pictures were sourced from a review on snakebots by Liljebäck P. et.al. [2]

While Fig. 1.1 shows different examples of ground-hugging, vertebrae structure type snakebots that were studied and developed for their ability to propagate through challenging terrains, there are also studies around using the body-structure of snakes as an inspiration for developing wire-driven “invertebrate like” continuum robotic arms which have better flexibility and manoeuvrability. *Continuum robotic arms* are continuously curving manipulators that can be used to manoeuvre through confined and inaccessible places and can be made to perform dexterous in-situ tasks. In comparison to a normal non-redundant robotic manipulator higher degree of freedom means that the robot will have a bigger task space for manipulation. Generally, the hyper-redundancy is achieved in the design through springs or cables. In the robotic manipulator studied by this project, the joints of the manipulator are driven by cables (wire ropes).

Cable-driven Hyper-redundant Manipulator (CDHRM) is a continuum robotic arm that is driven by wire-ropes and can be used in various applications involving service operations in confined spaces and hazardous environments, close range measurements, imaging etc. Generally, there are 2 types of CDHRM which differ in their design. Type I or multi-segmented

elastic CDHRM has multiple segmented backbone that are controlled by cables that run through the robot. Type II or fully-constrained rigid CDHRM has multiple rigid links attached to each other through universal joints in between with cables running through designated holes along the periphery of the shell of the link. Fig. 1.2 shows a Type II CDHRM which is similar to the system being studied by this project.

While such a CDHRM comes with extra manoeuvrability and accessibility properties, one of the other main advantages of this particular kind of robotic arm is that, it has no electronics in the robot itself. All of the electronics associated with the robot is located in the base to which the first joint of the arm is fixed. This makes it possible for the robot to access even hazardous envi-



Figure 1.2: *A cable-driven robotic snake arm developed by OC Robotics [3]*

ronments without any risk of damage to the robot (except minor structural damage) or hindrance to its operation. This is desirable as this feature of the structural design makes the robot qualify some of the safety considerations for industrial application and hence makes risk assessments for systems with such a structure easier.

1.1 Project Description and Motivations

A key area for research undertaken by UCL Robotics addresses one of the technology challenges of Robotics and Autonomous Systems (RAS) which is - "Robotic Teleoperation for Multiple Scales". Under this challenge, attempts have been made to use photogrammetry techniques for the structural testing of aircraft wing of an Airbus by measuring the shape and dimensions of aerodynamic surfaces [4], and the six degrees of freedom (6DoF) position and orientation of aerospace models to study its structural dynamics and aerodynamic forces.

Photogrammetry is one of the optical metrology methods that obtains measurements of physical objects and/or their environment with a high accu-

racy through the process of capturing or recording photographic images and patterns of electromagnetic imagery and analysing it to measure and interpret reliable information of the object and environment under consideration. There are many variants of photogrammetry. The photogrammetry technique being studied at UCL involves the measurement of shape of aerodynamic surfaces and the six degree of freedom (6DoF) position and orientation of aerospace models.

Photogrammetry for aerospace models requires a non-contact, non-intrusive, accurate and reliable technique with a high sample rate. Within this study, the measurement of aerospace models in wind tunnel test sections are novel and for such a specialised measurement, close range photogrammetry is well suited. Currently off-the-shelf-digital cameras are used from different fixed positions to simultaneously capture/record images, however this has limitations and makes close-range photogrammetry difficult.

The proposed solution to this involves using a snake-arm that uses optical metrology techniques for aerodynamic surfaces. For this study, there is a physical snake-arm on-site at Here East and this project proposed is aligned to the modelling and control research effort being developed for this snakebot.

The snake-arm robot at Here East has been provided by OC Robotics. It is classified as Series II X125 snake-arm robot. The robot was specifically designed for the robotic deployment of photogrammetry capability for measurement and inspection of cluttered or confined 3D environments. This snake-arm is the unsleeved version of the Series II X125. It can potentially achieve up to 225 degrees of cumulative bend [5] and has an independently controlled tip. Currently, it simply uses a nose following control technique to reach to the desired final position. The head of the snake can have different payloads attached to it. Unlike other versions of this series of snake-arms, the tethered end of this robot rests on a platform that can move forward and backward as required.

This project involves modelling this physical snakebot and understanding the end-effector pose control requirements for performing automated close-range optical metrology operations (specially photogrammetry). Once the control requirements are defined and the control strategies are realised, then the project aims to develop a robust control system that can perform measurements in various scenarios including measurements in wind tunnels. The control system configuration will be simulated, and the results obtained will be analysed and improved over the course of the project. Depending on the metrology technique adopted and in the case of using the

robot to perform dexterous tasks such as maintenance of the structure, the payload attached to the head of the robot is subject to change. Significant external disturbances may be acting on the robot as well, especially in cases where the robot is performing dexterous tasks or is in environment with considerable disturbances like in a wind tunnel. Therefore, the robustness of the system will be tested using change in mass of head node and/or external disturbances acting on the snake-arm.

1.2 Literature Review

On investigating some of the literature around snakebots it can be seen that many different kinds of approaches were adopted towards developing snakebots. Earlier in the decade, wheeled snakebots were being developed a lot more than wheel-less snakebots. [6] discusses some of the very early work done on wheeled snake robots and gives more insight to the motivations for study of wheel-less snake robots. Hirose S. [7] through his work came up with the mathematical equation that represented the shape that the snake assumes to propagate forward. He introduced the concept of serpenoid curve which has been extensively used by other research groups till date to develop a mathematical model for the serpentine locomotion of snakes. Work done by Pettersen, K. Y. et. al. [1] also provides a very detailed insight into the research around snake robots and discusses the mathematical model, design and control system used for productionising amphibious snake robots. This source is an excellent summary of the history and study of snake robots. While most of these papers discuss wheel-less snake robots with rigid links, [8] discusses how wire-driven "invertebrate like" continuum arm can have better flexibility and accessibility properties. A detailed review on the work done around continuum robots has also been published by Singh P. K. et. al. [9].

Since, the system under consideration is specifically that of the OC Robotics snake arm, materials published by the founders Graham A. and Buckingham R. were of immense interest. Even though material directly related to the robot or its technology could not be found, [10–12] by these authors were a good introduction to their work on snake arms and the bigger context of the varied applications of such robots. The slides in [12], also discussed the main features of snake arm robots.

Given that the system was identified as a wire rope driven, hollow core structure with passive revolute joints connecting the links to each other,

attempts were made to identify literature around modelling of a similar system. The work of Jones B. et al. [13] (dated 2006) has been cited in most papers. His work on kinematics for multi-section continuum robots seems to be the inspiration for most of the work around continuum snake arm robots. The works of Li Z. et al. [14–17] discusses how the skeleton of the robot is inspired from the skeleton of a snake and the idea of a the cable driven structure is inspired from the muscle arrangement in the tentacle of an octopus. Even though across various papers their work discusses the kinematic model of the snake arm robot, they do not discuss much about the dynamic model and control of the system. The references [18–21] represents the OC robotics snake arm the most. The system described by these papers uses a cable-driven hyper-redundant manipulator (CDHRM) in which the cables are controlled at the base with maxon motors exactly like the snake arm at Here East. The papers published by Tang L. et al. [18, 19] describes the design, motion planning and path tracking of the snake-arm. Most of the literature related to the CDHRM refers a detailed study and experiments based on its kinematic model. [19, 20] discuss the kinematic model in detail. The simulation and a path planning method for the CDHRM using the kinematic model is given by [18]. The main challenge in developing a kinematic model for this manipulator is around the inverse kinematic relationship. The forward kinematic model can have infinite solutions, is non-homogenous and has a non-linear relationship. [13] proposes a numerical solution to this problem whereas other sources propose more efficient geometrical methods [22] or complex learning methods [23]. However, the work of Xu W. et al [21] dated 2018, remains the most suitable reference found so far as it discusses the dynamic model and control of the cable-driven snake arm in great detail.

1.3 Aims and Objectives

The aim of this project is to contribute to the modelling and control effort around the snake-arm at Here East by developing and implementing a mathematical model and testing a control system configuration suitable for the task.

The main objectives of the project are:

- **Develop and implement a mathematical model that best represents the snake-arm at Here East:** The main aim of the project is to develop and implement a mathematical model that best represents

the system using Simulink. The model will be then tested to ensure that its behaviour is as expected and represents the snake-arm well.

- **Design and test control strategies that will help control the path followed by the arm during the metrology task:** The next objective is to define control requirements for the automation of the optical metrology process and design control strategies that performs the operation with the required accuracy. Through iterative testing, the model will be improved over the course of the project till satisfactory results are obtained.
- **Analyse the robustness of the control system configuration developed by running experiments with varied physical parameters:** The project also aims to investigate the robustness of the system since it is expected to work in environments with varied conditions where external disturbances affecting the arm's propagation might be considerable. It is also expected that depending on the technique adopted for the optical metrology different payloads may be attached to the head of the snakebot. The payload may also change in mass especially in occasions when improvements have been made to the same metrology technique under consideration. Therefore, the robustness of the system under the influence of change of mass of the head and the external disturbances will be investigated through experiments in the simulation software.

The following chapters discuss the theory, proposed solution and the results as obtained for the work undertaken as part of this project. Chapter 2 discusses the kinematic and dynamic model equations and its brief derivation. Chapter 3 discusses the attempt at implementation of these on Simulink and the proposed control scheme. Chapter 4 presents the results obtained and finally, Chapter 5 concludes the results observed, problems faced etc.

Chapter 2

Mathematical Modelling

The first step towards simulation of a control scheme for any plant involves developing/identifying a suitable mathematical model that represents the physical system (the plant). This chapter discusses briefly the derivation of the mathematical model of the snake arm under consideration. For this, first the mechanical design of the system was understood. Given the unique structural design of the CDHRM, it was realised that a kinematic model had to be developed in order to map the reference value (the end-effector position) to the actuator variable (motor angle of rotation) that can be controlled and vice versa. In the later half of the chapter, the derivation of the dynamic model (the plant block) representing the system has been discussed.

2.1 Understanding the system

To find the most suitable mathematical model for the snake-arm, a clear understanding of the system itself was to be established. This also included identifying the actuators that drive the system and understanding what can be controlled. This was essential to determine what parameters and system variables were required to be calculated in order to derive the mathematical model and to also understand exactly what can and cannot be controlled.

The videos that have been made available by OC Robotics [24–27], helped to understand the snake-arm and its capabilities to an extent. The snake-arm developed by OC Robotics is a hollow core structure with a fixed num-

ber of links. Each joint is driven by 3 wire ropes (big stainless steel cable ropes) that runs through the holes in the shell of the link at equidistant points from each other, parallel to the surface of the shell of the link. [28] (as can be seen in Fig. 2.1). The cables are controlled by 27 or more motors in the base [25] using maxon motors as it offers higher power density. All the electronics associated to the robot including the control systems are located at the base of the robot. This means that the robot can easily operate even in hazardous environments. The nose of the snake arm can accommodate different payloads and it also has a hollow core, which allows cables, hoses or any other equipment required by the specific payload to be routed through the centre of the snake arm.

In the video for Maxon Motors [25], Graham A. also talks about the unique nose following property of the snake arm robot and how it is easily controlled using a proprietary software developed by OC Robotics. From these information, it can be safely concluded that the system closely represents a cable-driven hyper-redundant manipulator (CDHRM) and so the mathematical model developed for the robot was derived mainly, with help of the following literature [19–21] around modelling of a CDHRM. Kinematic modelling and Dynamic modelling have both been studied for the CDHRM. While kinematic modelling of the CDHRM has been thoroughly studied, there are only very few papers that discuss dynamic modelling.

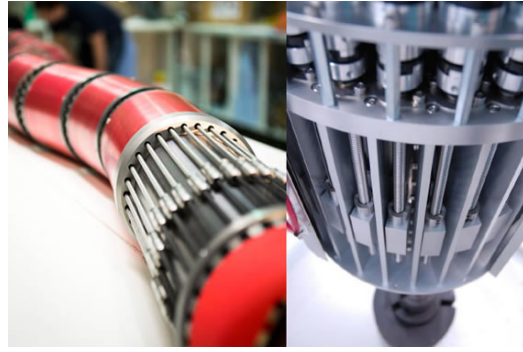


Figure 2.1: *The figure showing the maxon motors at the base to which the wire-ropes are attached (right) and the transmission system (left) that converts the rotational motion of motor to translational motion of the wire-ropes. [3]*

2.2 Model Preliminaries (Structural Design Description)

Consider a CDHRM snake-arm with **N universal joints** and hence has **N+1 links**. The length of each link is l_{link} , the distance between 2 joints is a and the distance between the two links is $2d$. Given the structural design, we can say that $l_{link} = a - 2d$. Each universal joint consists of 2 revolute

Driving subsystem

Motors Cable Transmission System

Base Link

Joint

Manipulator

Nose

End-effector is attached here

Every link has an identical structure but is connected at 90° with respect to each other. A link consists of a hollow cylindrical shell with two identical disks. The disk extensions from 2 links join in the centre onto a common lumen to form a universal joint. The links are arranged such that the joints in between forms an alternate pitch-yaw formation as described below.

For a given joint n ($n \in 1, 2, \dots, N$), the disc attached to link $n - 1$ is called the **distal disc** and the disc attached to link n is called the **proximal disc**. Given this notation, note that as depicted in Fig. 2.2 a given link number i can be denoted such that $i \in \{0, 1, 2, \dots, N\}$ where, base is link 0 and the nose is link N .

The diagram illustrates a cable-driven joint connecting Link $n-1$ and Link n . Link $n-1$ is a cylindrical component on the left, and Link n is a similar component on the right. They are connected by a central 'Universal joint' and a 'Hollow rod'. A 'Cable' is shown passing through the joint, with forces F_1 , F_2 , and F_3 indicated. A 'Disk' is part of the joint assembly. The angle between the links is labeled as 120° . The joint is labeled 'Joint n '.

10

$3N$ holes on each disc through which the cables are routed. Out of the $3N$ cables, each link will have a total $3(N - n + 1)$ cables routed to it such that $3(N - n)$ cables pass through and 3 cables terminate at the proximal disc of that link n . For a given joint n , cables C_{3n-2}, C_{3n-1} and C_{3n} drive joint n and terminate at the proximal disc of the link n as can be seen in Fig. 2.3. In the following sections, a cable will be represented by the notation, C_k where, $k \in \{1, 2, 3, \dots, 3N\}$.

2.3 Kinematic Model

To establish the relationship between the task space and the control space a kinematic model was required. The kinematic model here finds the relationship between the angle of rotation of the motor and the end-effector pose of the snake-arm. The conversion from the control space to task space and vice versa involves a multi-level mapping through relationships between parameters in different spaces. This is because unlike in common robotic manipulators, the actuators do not directly control the joint angles. In the case of a CDHRM, the motors drive the cables, and the cables drive the joint. This multi-level mapping from the control space to the task space is depicted in Fig. 2.4 below.

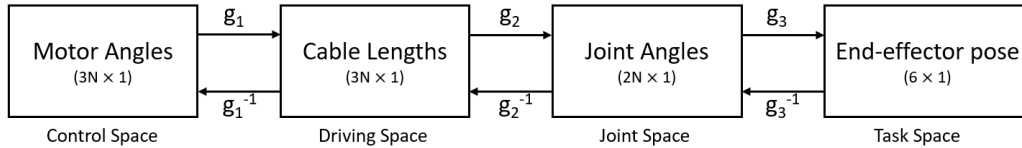


Figure 2.4: This picture shows the various mapping relationships required between different spaces to derive the relationship between the angle of rotation of the motor and the end-effector pose of the snake-arm

2.3.1 Motor Angle of Rotation - Cable Length Kinematics

Motor Angle of Rotation \rightarrow Cable Length (Forward Kinematics)

The rotation of the motors at the base are transmitted as translational motion causing a change in length of the cable driving the robotic arm. For a given change in angle of rotation of the motor the change in cable length

(the translational distance travelled) can be calculated as [29]:

$$\Delta L_k = k_{trans} \sqrt{(r_{drum})^2 + \left(\frac{p}{2\pi}\right)^2} \times \Delta \theta_{motor,k} \quad (2.1)$$

where,

- ΔL_k = Change in length of cable number k
- $\Delta \theta_{motor,k}$ = Angle of rotation of the motor
- k_{trans} = Transmission Ratio
- r_{drum} = Radius of the Drum
- p = Pitch of the Drum

Cable Length \longrightarrow Motor Angle of Rotation (Inverse Kinematics)

Equation (2.2) is the inverse kinematic relationship between the motor angle of rotation and cable length (g_1^{-1} as per Fig. 2.4). From this the forward kinematic relationship (g_1 as per Fig. 2.4) between the motor angle of rotation and cable length can be derived as:

$$\Delta \theta_{motor,k} = \frac{\Delta L_k}{k_{trans} \sqrt{(r_{drum})^2 + \left(\frac{p}{2\pi}\right)^2}} \quad (2.2)$$

2.3.2 Cable Length - Joint Angle Kinematics

Joint Angle \longrightarrow Cable Length (Inverse Kinematics)

To derive the relationship between the cable length and the joint angles, three reference frames $\{P_n\}$, $\{U_n\}$ and $\{D_n\}$ must be defined at each universal joints as seen in Fig. 2.5.

- The reference frame $\{U_n\}$ is defined such that its origin (O_U) corresponds to the centre of the joint. X_U is defined parallel to ζ_P and Y_U is defined parallel to ζ_D . Z_U is then defined using right-hand thumb rule.
- The reference frame $\{P_n\}$ is defined such that its origin (O_P) corresponds to the centre of the proximal disk. Z_P is normal to the proximal disc and X_P is parallel to ζ_P . Y_P is then defined using right-hand thumb rule.
- The reference frame $\{D_n\}$ is defined such that its origin (O_D) corresponds to the centre of the distal disk. Z_D is normal to the distal disc

and Y_D is parallel to ζ_D . X_D is then defined using right-hand thumb rule.

where, ζ_P and ζ_D represents the axis of rotation of the joint attached to the proximal and distal discs respectively. This means that by definition we have the relationship: $O_{P,n}O_{U,n} = O_{U,n}O_{D,n} = d$

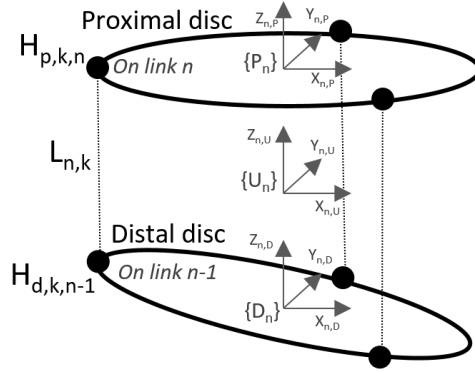


Figure 2.5: This diagram shows the different reference frames that were defined

The transformation matrix from $\{D_n\}$ to $\{P_n\}$ can be defined as:

$${}^D A_P = {}^D A_U \times {}^U A_P$$

$$= \begin{bmatrix} \cos \phi_{p,n} & 0 & \sin \phi_{p,n} & d \sin \phi_{p,n} \\ \sin \phi_{p,n} \sin \phi_{d,n} & \cos \phi_{d,n} & -\sin \phi_{d,n} \cos \phi_{p,n} & -d \sin \phi_{d,n} \cos \phi_{p,n} \\ -\cos \phi_{d,n} \sin \phi_{p,n} & \sin \phi_{d,n} & \cos \phi_{d,n} \cos \phi_{p,n} & d + d \cos \phi_{p,n} \cos \phi_{d,n} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

As can be seen from Fig. 2.5 the length of the cable between 2 discs can be derived as:

$$l_{k,n} = \| {}^D A_P^P H_{p,k,n} - {}^D H_{d,k,n-1} \| \quad (2.4)$$

such that,

$${}^P H_{p,k,n} = \begin{bmatrix} r \cos \delta_{n,k} \\ r \sin \delta_{n,k} \\ 0 \end{bmatrix};$$

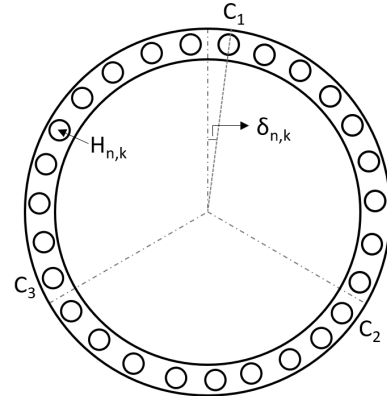


Figure 2.6: This diagram shows the angular positions of the holes on a disc.

$${}^D H_{d,k,n-1} = \begin{bmatrix} r \cos \delta_{k,n-1} \\ r \sin \delta_{k,n-1} \\ 0 \end{bmatrix}$$

where, ${}^P H_{p,k,n}$ and ${}^D H_{d,k,n}$ is the vector defining the position of the hole on the proximal and distal disc respectively w.r.t their local reference frame.

The angular position of these holes as seen in Fig. 2.6 can be calculated as:

$$\delta_{n,k} = \begin{cases} \frac{120^\circ}{(N+1)} \times M + 120^\circ(k-1) - 3^\circ & ; n = \text{odd} \\ \frac{120^\circ}{(N+1)} \times M + 120^\circ(k-1) + 87^\circ & ; n = \text{even} \end{cases} \quad (2.5)$$

where,

– $M = \text{floor}((k-1)/3) + 1$ is the link at which cable k terminates.

The total length of the cable k can therefore be written as:

$$L_k = M l_{link} + \sum_{j=1}^M l_{j,k} \quad (2.6)$$

Therefore, equations (2.4) and (2.6) define the inverse kinematics relationship between the joint angles and the cable lengths (g_2^{-1} as per Fig. 2.4).

Cable Length \rightarrow Joint Angle (Forward Kinematics)

We know that for a given joint n , cables $3n-2, 3n-1$ and $3n$ drive that joint. Therefore, considering equation (2.4) to be a function of ϕ_p and ϕ_d we know there exists a relationship:

$$\begin{bmatrix} l_{n,3n-2} \\ l_{n,3n-1} \\ l_{n,3n} \end{bmatrix} = \begin{bmatrix} f_{n,3n-2}(\phi_{p,n}, \phi_{d,n}) \\ f_{n,3n-1}(\phi_{p,n}, \phi_{d,n}) \\ f_{n,3n}(\phi_{p,n}, \phi_{d,n}) \end{bmatrix} \quad (2.7)$$

From equation (2.7), the differential equation can be derived as:

$$\begin{bmatrix} dl_{n,3n-2} \\ dl_{n,3n-1} \\ dl_{n,3n} \end{bmatrix} = J_d \begin{bmatrix} d\phi_{p,n} \\ d\phi_{d,n} \end{bmatrix}$$

where,

- J_d = Jacobian matrix of $[f_{3n-2}(\phi_{p,n}, \phi_{d,n}), f_{3n-1}(\phi_{p,n}, \phi_{d,n}), f_{3n}(\phi_{p,n}, \phi_{d,n})]^T$ w.r.t $[\phi_{p,n}, \phi_{d,n}]^T$ given by,

$$J_d = \begin{bmatrix} \frac{dl_{3n-2}}{d\phi_p} & \frac{dl_{3n-2}}{d\phi_d} \\ \frac{dl_{3n-1}}{d\phi_p} & \frac{dl_{3n-1}}{d\phi_d} \\ \frac{dl_{3n}}{d\phi_p} & \frac{dl_{3n}}{d\phi_d} \end{bmatrix}$$

Therefore, the forward kinematics relationship between the joint angles and the cable lengths (g_2 as per Fig. 2.4) can be numerically calculated as:

$$\begin{bmatrix} \Delta\phi_{p,n} \\ \Delta\phi_{d,n} \end{bmatrix} = J_d^+ \begin{bmatrix} \Delta l_{3n-2,n} \\ \Delta l_{3n-1,n} \\ \Delta l_{3n,n} \end{bmatrix} \quad (2.8)$$

where,

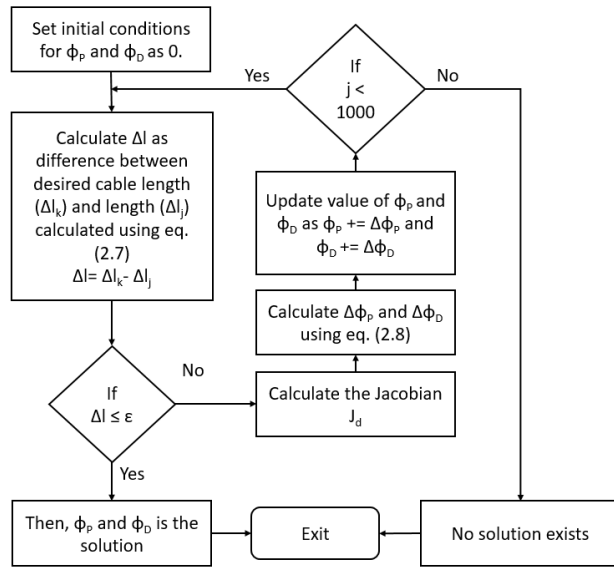
- J_d^+ = Moore-Penrose pseudo-inverse of J

However, the calculation of Jacobian matrix J_d requires us to know ϕ_p and ϕ_d and for this problem they are unknown. Therefore, equation (2.8) has 3 equations with only 2 unknowns and the equations are non-linear about them. This makes the unknown variables difficult to

isolate and therefore the equations cannot be solved analytically. To solve this problem, [21] proposes using a numerical iteration method that determines the least squares solution.

This involves starting from the known initial condition when $\phi_d = 0$ and $\phi_p = 0$ given by:

$$\begin{bmatrix} f_{n,3n-2}(0,0) \\ f_{n,3n-1}(0,0) \\ f_{n,3n}(0,0) \end{bmatrix} = \begin{bmatrix} 2d \\ 2d \\ 2d \end{bmatrix}$$



and the corresponding length l_k is calculated. The difference between the desired

Figure 2.7: Flowchart showing the numerical iteration for Cable Length → Joint Angle Kinematics

and the calculated length is computed and is checked to see if it is within tolerance (ϵ). Through each iteration the algorithm attempts to bring the calculated length value closer to the desired length value by using the error (Δl) to calculate the error in ϕ_p and ϕ_d and then updating the proposed solution with these errors. If no solution is found within tolerance before the maximum number of iterations is exceeded then the algorithm exits giving the message that "No solution could be found". Fig. 2.7 presents the flow chart for this algorithm.

2.3.3 Joint Angle - End-Effector Position Kinematics

The problem of getting the end-effector position from joint angles is a very standard problem in robotics. The most commonly adopted approach involves using the Denavit-Hartenberg (D-H) parameters which provides a systematic approach to deriving the transformation matrix and hence the Jacobian matrix. However, the inverse kinematics approach does not have a standard approach and hence a numerical iteration method has been discussed.

Joint Angle \rightarrow End-effector Position (Forward Kinematics)

The first step towards deriving the forward kinematic relationship between the joint angles and end effector position is to define the D-H frames. Fig. 2.8 shows the kinematic diagram with the D-H frames defined as per rules. Based on this, the D-H parameters can be derived as per Table 2.1

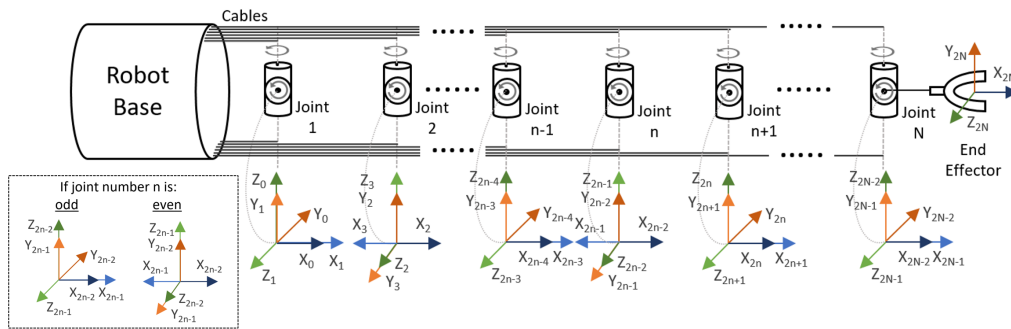


Figure 2.8: Kinematic diagram showing the universal joints as 2 revolute joints and shows the D-H frames.

From the DH parameter table (Table 2.1), a row n can be used to calculate the homogenous transformation matrix for adjacent frames $n-1A_n$ as:

$${}^{n-1}A_n = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n)\cos(\alpha_n) & \sin(\theta_n)\sin(\alpha_n) & r_n\cos(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n)\cos(\alpha_n) & -\cos(\theta_n)\sin(\alpha_n) & r_n\sin(\theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This can then be used to derive the base to end-effector transformation matrix as: ${}^0A_{2N} = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \dots {}^{2N-1}A_{2N}$, which is a function of joint angles θ .

The differential kinematic equation for any robotic manipulator is given by:

n	θ_n (°)	α_n (°)	r_n (mm)	d_n (mm)
1	θ_1	90	0	0
2	θ_2	0	a	0
3	$180+\theta_3$	90	0	0
4	$180+\theta_4$	0	a	0
...
17	θ_{17}	90	0	0
18	θ_{18}	0	a	0

Table 2.1: The DH Parameters

$$\dot{x}_{end} = J\dot{\Theta} \quad (2.9)$$

where,

– J = Jacobian Matrix

The Jacobian can therefore be calculated based on:

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}_{6 \times 2N} = \begin{bmatrix} [J_{v1}]_{3 \times 1} & [J_{v2}]_{3 \times 1} & \dots & [J_{v2N}]_{3 \times 1} \\ [J_{\omega 1}]_{3 \times 1} & [J_{\omega 2}]_{3 \times 1} & \dots & [J_{\omega 2N}]_{3 \times 1} \end{bmatrix}$$

where,

$$\begin{aligned} - J_{vi} &= {}^0R_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times ({}^0P_{2N} - {}^0P_{i-1}) \\ - J_{\omega i} &= {}^0R_{i-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

and, the rotational matrix (${}^{n-1}R_n$) and position vector (${}^{n-1}P_n$) can be taken from the homogeneous transformation matrix ${}^{n-1}A_n$ as:

$${}^{n-1}A_n = \begin{bmatrix} {}^{n-1}R_n & {}^{n-1}P_n \\ 0 & 1 \end{bmatrix}$$

and, ${}^0A_{i-1} = \prod_{j=1}^{i-1} {}^{j-1}A_j$

End-effector Position \rightarrow Joint Angle (Inverse Kinematics)

From equation (2.9), we can derive the inverse relationship as:

$$\dot{\Theta} = J^+ \dot{x}_{end} \quad (2.10)$$

However, similar to the situation in subsection 2.3.2, to calculate the Jacobian matrix you need to know the joint angles. It is also a non-linear equation such that the joint angles cannot be separated and hence the equation cannot be calculated analytically. This means that you need to solve this through numerical solution as well. The algorithm used to perform this iteration is shown in Fig. 2.9.

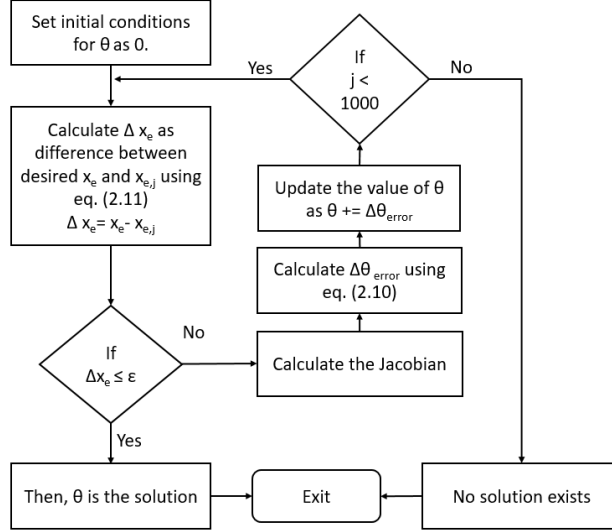


Figure 2.9: *Flowchart showing the numerical iteration for End-effector Position \rightarrow Joint Angle Kinematics*

Note that, in the algorithm the differential equation given by eq. (2.9) was modified to numerical form such that in the first step x_{end} is calculated as:

$$x_{end} = J(\Theta - \Theta_{prev}) + x_{end_prev} \quad (2.11)$$

Generally, the end effector position to joint angle inverse kinematic relationship is considered a challenging one. While several methods based on geometry, learning etc. exist to tackle this problem, they are specific to the robotic manipulator that they are being proposed for and often do not apply to manipulators with a different structural design. Still, better methods like the modal method proposed in [30] exist for CDHRM. However, for simplicity the numerical method was adopted.

2.4 Dynamic Model

To develop a high performing control for the snake arm, a dynamic model considering the various forces acting on the system is important. To derive the dynamic model, we need to first analyse the forces acting on the

system. For any generic robotic manipulator, the dynamic model can be represented as:

$$u = M(\Theta) \ddot{\Theta} + C(\Theta, \dot{\Theta}) \dot{\Theta} + F_v \dot{\Theta} + g(\Theta) + J^T(\Theta) F_e \quad (2.12)$$

where,

- $M(\Theta)$ = Inertia matrix
- $C(\Theta, \dot{\Theta})$ = Centrifugal and Coriolis forces
- F_v = Coefficient of viscous friction matrix
- $g(\Theta)$ = Moment generated by the presence of gravity
- u = Actuation torque¹
- F_e = Force and moment vector exerted by the end-effector on the manipulator.²

A popular method to derive the dynamic model from first principles in this form involves using the Lagrangian method. Another popular method for deriving the dynamic model for robotic manipulators is, Newton-Euler Recursive algorithm. For deriving the dynamic model of CDHRM this method is preferred as it offers higher efficiency [21] and does not involve differentiation.

2.4.1 Force Analysis

One of the steps in the Newton-Euler Recursive algorithm involves using force balance equations acting on a link. Figure 2.10, shows the free body diagram of the forces acting on the link. Apart from these forces acting on the link there are cable tension forces. Some of the expressions describing these forces are as follows:

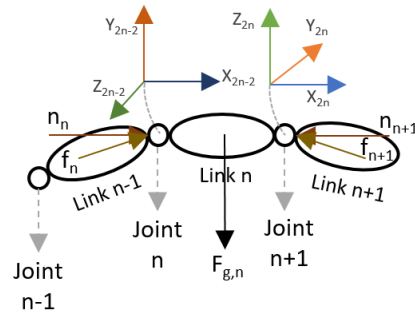


Figure 2.10: *Free Body diagram showing forces acting on a link (excludes the cable contact and forces).*

Link Gravitational Force

Link Gravitational Force is the force acting on the link n due to the presence of gravity.

¹Note that, our system is passive and hence this is 0 in the case of a CDHRM.

²This is the terminal condition that is to be defined in the Newton-Euler Recursive algorithm.

This is given by:

$${}^{2n}F_{g,n} = {}^{2n}R_0(-m g_0) \quad (2.13)$$

where,

- g = acceleration due to gravity given by $g = 9.8\text{m/s}^2$

In the Newton-Euler Recursive algorithm, the above expression is not explicitly used as gravity as it is considered implicitly in the calculation of linear acceleration as $g_0 = [0 \ 0 \ g]^T$ acting at the base.

Link Inertia Force

The link inertia force can be calculated if the linear acceleration of the centre of mass and inertia are known.

$${}^{2n}F_n = m {}^{2n}\dot{v}_{cn} \quad (2.14)$$

Similarly, if the angular acceleration and velocity are known the link inertia moments can be calculated as:

$${}^{2n}T_n = {}^{cn}I_n {}^{2n}\dot{\omega}_n + {}^{2n}\omega_n \times {}^{cn}I_n {}^{2n}\omega_n \quad (2.15)$$

In the above equations:

- m = mass of a link
- ${}^{cn}I_n$ = Inertia Tensor

The Inertia tensor ${}^{cn}I_n$ about the frame of reference $\{cn\}$ fixed to the centre of mass of the link can be defined as a diagonal matrix given by the equations for Inertia for a hollow cylinder. This gives:

$${}^{cn}I_n = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (2.16)$$

where,

- $I_{xx} = I_{yy} = \frac{m}{4}(r_{outer}^2 + r_{inner}^2) + \frac{m}{12}(l_{link})^2$
- $I_{zz} = \frac{m}{2}(r_{outer}^2 + r_{inner}^2)$

Adjacent Link Interaction Forces

The interaction between adjacent links subject the link to interaction forces, f_n and $-f_{n+1}$ by the link $n - 1$ and link $n + 1$ respectively. The link is also subject to a torque τ_n and $-\tau_{n+1}$ by these adjacent links. These interaction forces has been shown in Fig. 2.10. These forces and torques can calculated recursively using Newton-Euler Recursive algorithm and this is discussed later on in section 2.4.2.

Note, that the torque described in frame $\{2n-1\}$ has only a x-axis component along X_{2n-1} as the rotation axes of the universal joint, ζ_P and ζ_D is consistent with Z_{2n-1} and Y_{2n-1} .

Cable Tension Forces

When the motor rotates, the cable experiences a tension across it. This can be denoted in terms of its amplitude and direction vector as:

$$\begin{aligned} \mathbf{F}_{tp,n,k} &= F_{tp,n,k} \mathbf{e}_{tp,n,k} \\ \mathbf{F}_{td,n,k} &= F_{td,n,k} \mathbf{e}_{td,n,k} \end{aligned} \quad (2.17)$$

Given that tension acts along the cable the direction vector $\mathbf{e}_{tp,n,k}/\mathbf{e}_{td,n,k}$ must be same as the direction vector $\mathbf{e}_{L1,n,k}/\mathbf{e}_{Ld,n,k}$ representing the direction unit vector of the position vector of the cable. Therefore,

$$\mathbf{e}_{tp,n,k} = \mathbf{L}_{p,n,k} / \|\mathbf{L}_{p,n,k}\| \quad (2.18)$$

$$\mathbf{e}_{td,n,k} = \mathbf{L}_{d,n,k} / \|\mathbf{L}_{d,n,k}\| \quad (2.19)$$

The global position vector of the cables can be written as:

$$\mathbf{L}_{p,n,k} = \begin{cases} {}^0A_{2n} {}^{2n}\mathbf{r}_{p,n,k} - \begin{bmatrix} -d & r \cos \delta_{n,k} & r \sin \delta_{n,k} & 1 \end{bmatrix}^T & ; n = 1 \\ {}^0A_{2n} {}^{2n}\mathbf{r}_{p,n,k} - {}^0A_{2n-2} {}^{2n-2}\mathbf{r}_{d,n-1,k} & ; n > 1 \end{cases} \quad (2.20)$$

$$\mathbf{L}_{d,n,k} = {}^0A_{2n} ({}^{2n}\mathbf{r}_{d,n,k} - {}^{2n}\mathbf{r}_{p,n,k}) ; n \geq 1 \quad (2.21)$$

where, the homogeneous representation of the hole position can be written as:

$$\begin{aligned} {}^{2n}\mathbf{r}_{p,n,k} &= \\ \begin{cases} \begin{bmatrix} d - l_{link} & r \sin \delta_{n,k} & -r \cos \delta_{n,k} & 1 \end{bmatrix}^T & (n = \text{odd number}) \\ \begin{bmatrix} d - l_{link} & r \cos \delta_{n,k} & r \sin \delta_{n,k} & 1 \end{bmatrix}^T & (n = \text{even number}) \end{cases} \end{aligned} \quad (2.22)$$

$${}^{2n}\mathbf{r}_{d,n,k} = \begin{cases} [-d & r \sin \delta_{n,k} & -r \cos \delta_{n,k} & 1]^T & (n = \text{odd number}) \\ [-d & r \cos \delta_{n,k} & r \sin \delta_{n,k} & 1]^T & (n = \text{even number}) \end{cases} \quad (2.23)$$

Tension in the cable is the force that induces the motion of cables and the change in its length. When the cables move, there is friction between the holes and cables. The frictional force can be related to the force causing the motion which is tension in this case. Therefore, to calculate the value of the magnitude of cable tension forces we recursively use eq. 2.29 and 2.26 that are discussed in the part that follows this section.

Cable Contact Forces

As the cables slide in the holes, frictional forces act on the cable. Therefore, the resultant of both the cable tension and the frictional force can be thought of as the contact force of the cable at one hole. Fig. 2.11, depicts the tension forces and also shows that the frictional force, as one would expect, acts opposite to that of the direction of motion³. Therefore, frictional forces caused by the sliding motion of the cables can be related to the tension through the following equations:

$$\|\mathbf{F}_{fr_p,n,k}\| = (F_{tp,n,k} + F_{td,n,k}) \frac{(e^{\mu\varphi_{p,n,k}} - 1)}{(e^{\mu\varphi_{p,n,k}} + 1)} \quad (2.24)$$

$$\varphi_{p,n,k} = \cos^{-1}(e_{tp,n,k} \cdot e_{td,n,k}) \quad (2.25)$$

$$e^{\mu\varphi_{p,n,k}} = \frac{F_{tp,n,k}}{F_{td,n,k}} \quad (2.26)$$

$$\|\mathbf{F}_{fr_d,n,k}\| = (F_{td,n-1,k} + F_{tp,n,k}) \frac{(e^{\mu\varphi_{d,n-1,k}} - 1)}{(e^{\mu\varphi_{d,n-1,k}} + 1)} \quad (2.27)$$

$$\varphi_{d,n-1,k} = \cos^{-1}(e_{td,n-1,k} \cdot e_{tp,n,k}) \quad (2.28)$$

$$e^{\mu\varphi_{d,n-1,k}} = \frac{F_{td,n-1,k}}{F_{tp,n,k}} \quad (2.29)$$

where, μ is the coefficient of friction and $F_{fr_p,n,k}$ and $F_{fr_d,n,k}$ acting on the hole $H_{n,k}$ in the proximal and distal disc respectively.

³Note that the direction of motion can also be thought of as the direction of velocity of cables. Therefore, [21] discusses computing the sign of the frictional force as the sign of the resultant of velocity of the cable at that point.

It has been established that the frictional forces and the normal forces together produce contact forces at the discs. Given this, from above equations that establish the relationship between tension and frictional forces we can derive the contact forces $F_{cp,n,k}$ and $F_{cd,n,k}$ acting on the hole $H_{n,k}$ in the proximal and distal disc respectively, in terms of the cable tension forces acting at that point as:

$$F_{cp,n,k} = \begin{cases} F_{td,n,k} e_{td,n,k} & 3n+1 \leq k \leq 3N; n \leq M \\ -F_{tp,n,k} e_{tp,n,k} & \\ -F_{tp,n,k} e_{tp,n,k}, & 3n-2 \leq k \leq 3n; n = M \end{cases} \quad (2.30)$$

$$F_{cd,n-1,k} = F_{tp,n,k} e_{tp,n,k} - F_{td,n-1,k} e_{td,n-1,k} \quad (2.31)$$

$$3n+1 \leq k \leq 3N; n \leq M$$

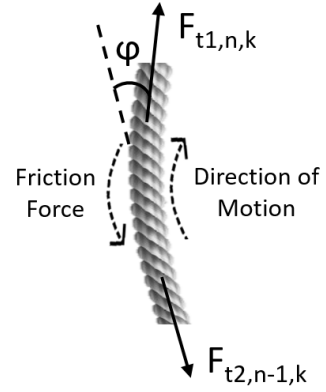


Figure 2.11: This diagram depicts the direction in which the friction forces act the forces causing them.

Given this the equivalent contact forces and moments can be denoted as:

$${}^{2n}F_{ec,n} = \sum_{k=3n+1}^{3N} {}^{2n}F_{cd,n,k} + \sum_{k=3n-2}^{3N} {}^{2n}F_{cp,n,k} \quad (2.32)$$

$$\begin{aligned} {}^{2n}T_{ec,n} = & \sum_{k=3n+1}^{3N} ({}^{2n}r_{d,n,k} - {}^{2n}P_{cn}) \times {}^{2n}F_{cd,n,k} \\ & + \sum_{k=3n-2}^{3N} ({}^{2n}r_{p,n,k} - {}^{2n}P_{cn}) \times {}^{2n}F_{cp,n,k} \end{aligned} \quad (2.33)$$

where, ${}^{2n}F_{cd,n,k} = {}^{2n}R_0 F_{cd,n,k}$ and ${}^{2n}F_{cp,n,k} = {}^{2n}R_0 F_{cp,n,k}$

2.4.2 The Inverse Dynamic Model

To derive the dynamic model for the CDHRM, Newton-Euler Recursive Algorithm is used here as it offers higher efficiency in comparison to the Lagrange method. Newton-Euler Recursive Algorithm involves two iterations - a Forward Recursion that calculates the velocities and accelerations and a Backward Recursion that calculates the forces and torques. The Newton-Euler method that is described here provides the inverse dynamics relationship. However, the same algorithm can be used to derive back the forward dynamics and this will be discussed in the next chapter.

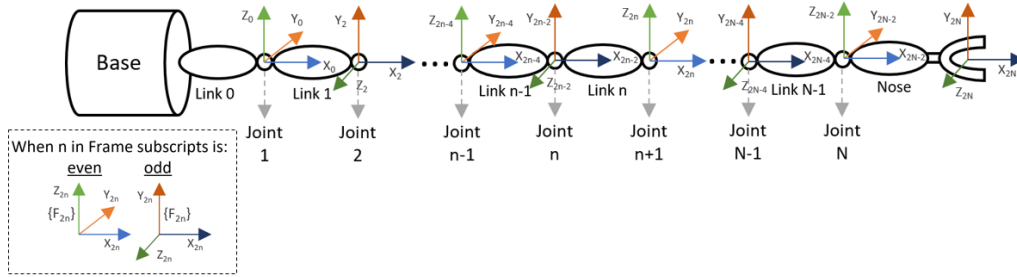


Figure 2.12: This diagram depicts the frames of reference defined for the derivation of the dynamic model.

Before deriving the dynamic model, the reference frames attached to each joint must be established. To keep things simple the reference frame $\{F_{2(n-1)}\}$ is defined at joint n and this corresponds to D-H frames defined in section 2.3.3. The reference frames as defined can be seen in Fig. 2.12.

Corresponding to these frames the following position vectors P_n , $P_{n,n+1}$ and P_{cn} can be defined for the joints. P_n is the position vector of joint n with respect to base frame. $P_{n,n+1}$ is the position vector from joint n to joint $n+1$. P_{cn} is the position vector of the centroid of link n with respect to $\{F_{2n}\}$. Fig. 2.13 shows these vectors defined for a link n .

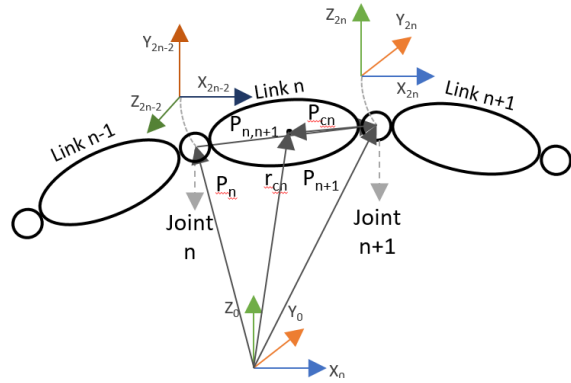


Figure 2.13: This diagram depicts the vectors as per definitions for the Newton-Euler recursive algorithm.

Forward Recursion

The first recursion involves calculation of the angular velocity, angular acceleration and linear acceleration for each link through forward recursion from the base to the end-effector.

With the initial conditions defined as, ${}^0\omega_0 = 0 \text{ rad/s}$; ${}^0\dot{\omega}_0 = 0 \text{ rad/s}^2$ and ${}^0\dot{v}_1 = [0 \ 0 \ g]^T$ the following expressions can be derived for each link:

$${}^{2n}\omega_n = {}^{2n}R_{2n-2} z_0 \dot{\theta}_{2n-1} + {}^{2n}R_{2n-1} z_0 \dot{\theta}_{2n} + {}^{2n}R_{2n-2} {}^{2n-2}\omega_{n-1} \quad (2.34)$$

$$\begin{aligned}
{}^{2n}\dot{\omega}_n = & {}^{2n}R_{2n-2} \left({}^{2n-2}\omega_{n-1} z_0 \dot{\theta}_{2n-1} \right) + {}^{2n}R_{2n-2} {}^{2n-2}\dot{\omega}_{n-1} + \\
& {}^{2n}R_{2n-2} z_0 \ddot{\theta}_{2n-1} + {}^{2n}R_{2n-1} z_0 \ddot{\theta}_{2n} + \\
& {}^{2n}R_{2n-2} \times \left((z_0 \dot{\theta}_{2n-1} + {}^{2n-2}\omega_{n-1}) \times {}^{2n-2}R_{2n-1} z_0 \dot{\theta}_{2n} \right)
\end{aligned} \tag{2.35}$$

$${}^{2n-2}\dot{v}_n = \begin{cases} {}^{2n-2}R_{2n-4} {}^{2n-4}\dot{v}_{n-1} + {}^{2n-2}\dot{\omega}_{n-1} \times {}^{2n-2}P_{n-1,n} + \\ {}^{2n-2}\omega_{n-1} \times ({}^{2n-2}\omega_{n-1} \times {}^{2n-2}P_{n-1,n}) \end{cases}, n > 1 \tag{2.36}$$

$$\begin{aligned}
{}^{2n}\dot{v}_{cn} = & {}^{2n}\dot{\omega}_n \times ({}^{2n}P_{n,n+1} + {}^{2n}R_{2n-2} {}^{2n-2}\dot{v}_n) + {}^{2n}P_{cn} \\
& + {}^{2n}\omega_n \times ({}^{2n}\omega_n \times ({}^{2n}P_{n,n+1} + {}^{2n}P_{cn}))
\end{aligned} \tag{2.37}$$

Backward Recursion

The second recursion involves calculating the interaction force and torque acting on each link recursively from the force and moment balance equations. The force and moment balance equations are:

$$\begin{aligned}
{}^{2n}F_n = & {}^{2n}f_n + {}^{2n}F_{ec,n} - {}^{2n}R_{2n+2} {}^{2n+2}f_{n+1} \\
{}^{2n}T_n = & {}^{2n}\tau_n - {}^{2n}R_{2n+2} {}^{2n+2}\tau_{n+1} - ({}^{2n}P_{n,n+1} + {}^{2n}P_{cn}) \times {}^{2n}f_n \\
& - {}^{2n}P_n \times ({}^{2n}R_{2n+2} {}^{2n+2}f_{n+1}) + {}^{2n}T_{ec,n}
\end{aligned}$$

With the terminal conditions defined as, ${}^{2N+2}f_{N+1} = 0$ N the following expressions can be derived for each link:

$${}^{2n}f_n = {}^{2n}R_{2n+2} {}^{2n+2}f_{n+1} + {}^{2n}F_n - {}^{2n}F_{ec,n} \tag{2.38}$$

$$\begin{aligned}
{}^{2n}\tau_n = & {}^{2n}R_{2n+2} {}^{2n+2}\tau_{n+1} + {}^{2n}T_n + ({}^{2n}P_{n,n+1} + {}^{2n}P_{cn}) \times {}^{2n}f_n \\
& + {}^{2n}P_n \times ({}^{2n}R_{2n+2} {}^{2n+2}f_{n+1}) - {}^{2n}T_{ec,n}
\end{aligned} \tag{2.39}$$

where,

- ${}^{2n}F_n$ = Link Inertia force given by eq. (2.14)
- ${}^{2n}F_{ec,n}$ = Equivalent moment due to contact forces about the centroid given by eq. (2.32)
- ${}^{2n}T_n$ = Link Inertia moment given by eq. (2.15)
- ${}^{2n}T_{ec,n}$ = Equivalent moment due to contact forces about the centroid given by eq. (2.33)

From eq. (2.39), we now get the dynamic relationship between the driving force (cable tension forces) and the joint angles as⁴:

$$\tau_{m,k} = \frac{r_{drum} F_{td,0,k}}{\eta} \quad (2.40)$$

Therefore it can be said that eq. (2.39) and (2.40) together, defines the inverse dynamic relationship for a CDHRM. Using these, we can derive the required torque to be produced by the motor to achieve a required configuration defined by the given joint angles.

2.4.3 The Forward Dynamic Model

In the previous subsection, we used the Newton-Euler Recursive Algorithm to derive the inverse dynamic relationship. Based on eq. (2.12), we can derive the forward dynamic relationship as:

$$\ddot{\Theta} = M^{-1}(\Theta) (u - C(\Theta, \dot{\Theta})\dot{\Theta} - F_v\dot{\Theta} - g(\Theta) - J^T(\Theta) F_e) \quad (2.41)$$

However, this representation is derived using the Lagrangian method and since the inverse dynamics was discussed based on the Newton-Euler Recursive Algorithm, it would be preferred to use the same to derive forward dynamics using the same as well. [31], proposes a method to do this. It suggest that if used iteratively, the relationship defined by Newton-Euler Recursive Algorithm can be used to derive eq. (2.41) by setting some of the inputs of the Newton-Euler Recursive Algorithm block to 0 or 1, eliminating terms and isolating the coefficients. The method to do this has been discussed in section 3.2.6 in the next chapter.

⁴This is based on the relationship between force transferred to the cable for a certain torque produced by a motor.

Chapter 3

Implementation and Simulation

Now that a suitable mathematical model was developed, the second objective of the project was to design a control scheme that would facilitate precise control of the end-effector position. This chapter puts together the different equations discussed in the previous chapter and explains how it was implemented as different blocks on Simulink. These blocks were then put together as per the proposed control scheme to simulate its behaviour.

3.1 Proposed Control Scheme

It is known that the snake-arm robot is indirectly controlled by cables that are attached to the motors at the base of the arm. Based on this knowledge of the inputs and outputs, as a baseline control scheme, the configuration as shown in Fig. 3.1 using the dynamic and kinematic model developed in the previous chapter, was proposed.

The control objective here is to achieve a very low tracking error (<0.1 mm) for the end-effector position. This means our reference value is the end-effector position but the value that can be controlled is the angular position of the actuator. Therefore, in the proposed scheme the kinematic model is used to map the reference value from end-effector position to the angular position of the actuator.

3.2 Implementation on Simulink

3.2.1 Dynamic Parameters

The equations described in the previous chapter included several dynamic parameters whose values are constant and are usually estimated from the datasheets of the components or the design specifications of the system that is made available. However, for the purpose of this project the idea was to use dynamic parameters from a literature source that describes a similar system. However, all the literature that was identified, associated to dynamic modelling of a CDHRM, performs experiments directly on the actual system and does not discuss a simulation. This meant that the equations for the model as described in this report were compiled from different sources and some were derived. This meant that some of the independent dynamic parameters had to be taken from different sources. Table 3.1, lists the value used for the simulation.

So, while the values of these parameters were mainly taken from [21], some of the other independent dynamic parameters were assigned based on what was found from different sources. Therefore, these values are possibly not the best estimates of these parameters and could become a reason why the behaviour of the simulated system might deviate from the actual system. A common approach to reducing such parameter estimation errors involves incorporating a regressor term $Y(\Theta, \dot{\Theta}, \ddot{\Theta})$ in the dynamic model to compensate estimation errors. Another approach involves, modelling the system using Computer-Aided Design (CAD) software to derive some of the physical parameters of the manipulator. To improve the performance further, more sophisticated methods involving learning techniques could also be used. However, all these remain future work and is beyond the scope of this project.

Notation	Definition	Value
N	Number of joints	9
a	Distance between two adjacent universal joints	140 mm
$2d$	Distance between discs of two adjacent links	23 mm
m	Mass of a link	0.2 kg
l_{link}	Length of the link ($= a - 2d$)	117 mm
r	Radius of the link	20.5 mm
d_{drum}	Diameter of the cable drum	84 mm
r_{drum}	Radius of the cable drum ($= \frac{d_{drum}}{2}$)	42 mm
p	Pitch of the cable drum	8 mm
k_{trans}	Transmission ratio	0.36
η	Transmission efficiency	0.39
J_R	Rotor Inertia	$0.864 \times 10^{-7} \text{ kg m}^2$
J_L	Load Inertia	$0.014 \times 10^{-7} \text{ kg m}^2$
k_m	Speed-torque Gradient	$0.014 \times 10^{-7} \text{ rad s}^{-1} \text{N}^{-1} \text{m}^{-1}$
n_0	No load Speed	$1007.4 \text{ rad s}^{-1}$
g	Acceleration due to gravity	9.6 m/s^2
μ	Coefficient of friction	0.17

Table 3.1: Table defining the parameters and its value as used in the simulation.

3.2.2 Reference Mapping (Kinematic Model)

Based on the control objective defined earlier in this chapter, we know that the reference value is the desired end effector position. The unique design of the CDHRM robot implies that joint angles and the end-effector position

are not directly controlled by the actuator. This meant that as described by Fig. 3.2, there are other intermediate spaces between the task space and the control space. So, to map between these spaces a kinematic model had to be developed. From the kinematic relationships defined by eq. (2.10),(2.2) and (3.1), we map the desired end-effector position to the desired amount of angular rotation of the motor.

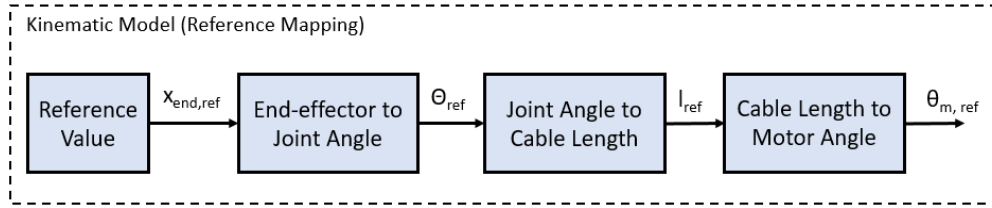


Figure 3.2: The part of the control scheme that maps the reference value from the task space to the control space.

Note that, eq.(2.6) had to be manipulated further to obtain eq. (3.1) such that the angular position w.r.t initial position¹ was calculated. We know, $L_{0,k} = M l_{link} + n \times (2d)$. Then, the required change in length of the cable can be calculated as $\Delta L_k = L_k - L_{0,k}$

$$\Delta L_k = \sum_{j=1}^M l_{j,k} - 2nd \quad (3.1)$$

3.2.3 Controller Block

As mentioned earlier, the proposed control scheme involves a feedback + feedforward strategy.

As can be seen in Fig. 3.3, the feedback signal goes through a PID controller block. So, the output of this block can be described as:

$$u_{fb} = K_P e_{\theta_m}(t) + K_I \int_0^t e_{\theta_m}(t') dt' + K_D \frac{de_{\theta_m}(t)}{dt} \quad (3.2)$$

This signal u_{fb} is then added to the feedforward signal, u_{ff} which is described in the next subsection.

¹Initial position here refers to the configuration of the robot when all joint angles are 0.

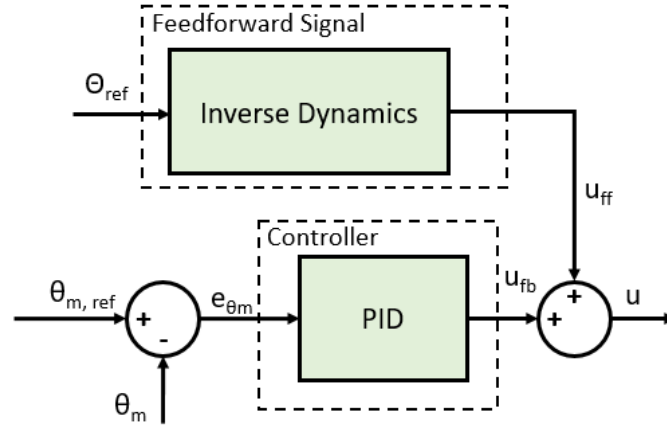


Figure 3.3: The blocks associated to the controller in the proposed design

3.2.4 Feedforward Block

The feedforward block uses the inverse dynamics relationship established by eq. (2.39) to calculate u_{ff} . As seen in Fig. 3.3, the feedforward signal block takes the θ_{ref} as the input. However, the equation requires τ_m as the input as well and this is unknown. This means that in eq. (2.39), there are four unknowns (three of which are values of cable tension that is unknown since τ_m is unknown). To simplify this problem and to allow preliminary analysis, τ_m was just set to zero as can be seen in the figure in Appendix D.4.

[21] describes a better and more accurate method to derive the feedforward signal. In the given problem, since τ_m is unknown, eq. (2.39) will have four unknowns. If three of these are solved by treating the problem as a constrained minimization problem as described below, then the fourth unknown can be calculated (the output) as:

$$\left\{ \begin{array}{l} \text{minimize : } F = \sum_{k=3n-2}^{3n} F_{tp,n,k} \\ \text{subject to : } \left\{ \begin{array}{l} {}^{2n-1}n_{c,n} = {}^{2n-1}R_{2n} \left({}^{2n}R_{2n+1} {}^{2n+1}n'_{n+1} \right. \\ \quad \left. - {}^{2n}R_{2n+2} {}^{2n+2}n_{n+1} + {}^{2n}T'_{ec,n} \right. \\ \quad \left. + {}^{2n}P_{n,n+1} \times \sum_{j=n+1}^N {}^{2n}R_{2j} {}^{2j}F_{ec,j} \right) \\ {}^{2n}R'_{2n-1}n_{c,n} = {}^{2n}R'_{2n-1}n'_n \\ F_{t1,n,k} \geq F_{\min}; (k = 3n - 2, \dots, 3n) \end{array} \right. \end{array} \right. \quad (3.3)$$

where ${}^{2n}F_{c2,n,k} = {}^{2n}R_0 F_{c2,n,k}$, ${}^{2n}F_{c1,n,k} = {}^{2n}R_0 F_{c1,n,k}$ and $F_{\min} = 40N$. F_{\min} here is the pretensioning force required to keep cable tense.

After implementing a working version of this block based on the first approach, an attempt was made at implementing this block based on the second approach, however it was unsuccessful and could not be completed.

3.2.5 Actuator Block (Maxon Motor)

It was identified from online sources that the actuator used here is a maxon motor. Therefore, from its datasheet [32], it was found that the following model can be used to represent the motor:

$$\ddot{\theta}_m = \frac{u}{J_L + J_R} \quad (3.4)$$

where,

- J_L = Load Inertia
- J_R = Rotor Inertia

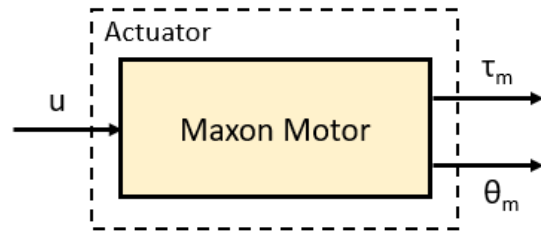


Figure 3.4: Block associated to the actuator in the proposed control scheme

Based on this equation, the model was implemented on Simulink as shown in Fig. 3.5.

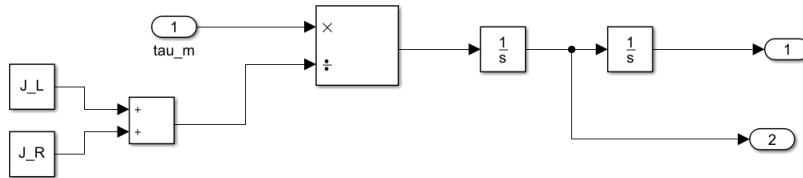


Figure 3.5: Block diagram showing the implementation of the maxon motor model.

Further, from the obtained speed the ideal transmitted torque (without loss) is calculated as:

$$\tau_m = n_0 - k_m M \quad (3.5)$$

where,

- n_0 = No load speed
- k_m = Speed-torque gradient

The value τ_m then goes in as input to the robot's dynamic model. Note that, the values for motor characteristics n_0, k_m, J_L and J_R were derived from [33] for the maxon motor serial number 110063.

3.2.6 Plant Block (Dynamic Model)

In the previous chapter, we described eq.(2.12) as the general inverse dynamic relationship for the CDHRM and then used the Newton-Euler Recursive algorithm to derive eq. (2.39) which is another equation that represents the same system. Therefore, if we think of the block that implements all the equations associated to Newton-Euler Recursive algorithm as a function given by $NE(\Theta, \dot{\Theta}, \ddot{\Theta})$ then, we can say that:

$$u = NE(\Theta, \dot{\Theta}, \ddot{\Theta}) = M(\Theta) \ddot{\Theta} + C(\Theta, \dot{\Theta}) \dot{\Theta} + F_v \dot{\Theta} + g(\Theta) + J^T(\Theta) F_e$$

From this we can derive that,

$$NE(\Theta, \dot{\Theta}, 0) = C(\Theta, \dot{\Theta}) \dot{\Theta} + F_v \dot{\Theta} + g(\Theta) + J^T(\Theta) F_e \quad (3.6)$$

Next, the Mass matrix $M(\Theta)$ can be derived through n iterations. Each iteration i gives one column $M_i(\Theta)$ of $M(\Theta)$ such that,

$$NE(\Theta, 0, e_i) = C(\Theta, \dot{\Theta}) \dot{\Theta} + F_v \dot{\Theta} + g(\Theta) + J^T(\Theta) F_e. \quad (3.7)$$

where, e_i is the i^{th} column of the identity matrix I_{2N} . Note that for eq. (3.7) to hold true, values of g and F_e must be explicitly set to 0. Also, the viscous friction term $F_v \dot{\theta}$ in eq. (2.12), was omitted for the simulation here.

The figure below shows the block diagram of the dynamic model as implemented in Simulink.

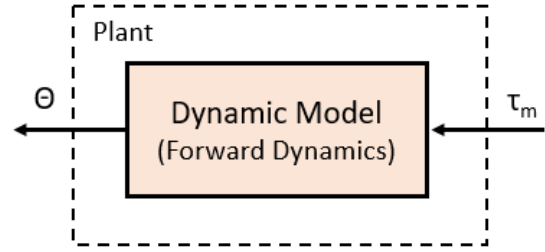


Figure 3.6: The block in the control scheme referring to the dynamic model of the system.

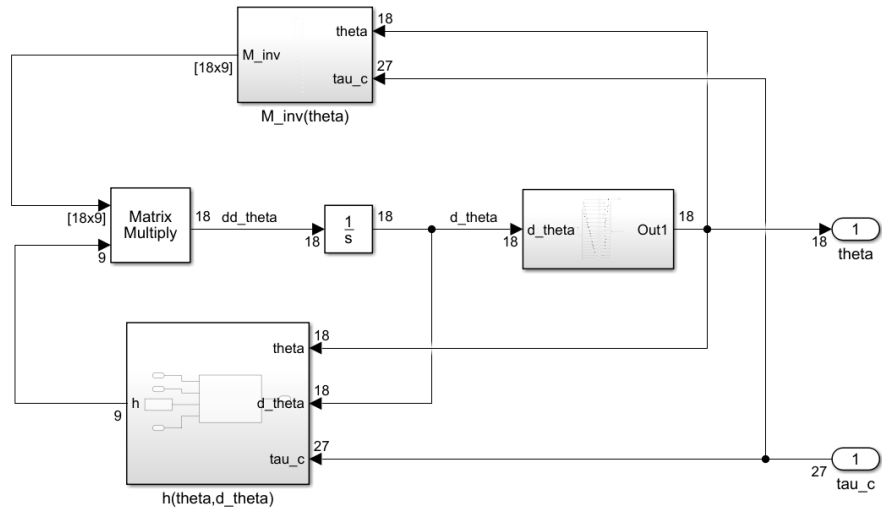


Figure 3.7: Block Diagram showing the implementation of the forward dynamics model.

Chapter 4

Results and Analysis

In the previous chapter, the proposed control scheme and the block diagram based on which the Simulink model was implemented was discussed. The blocks that were implemented on Simulink were first cross-validated and then two models in the control scheme were separately tuned to understand their behaviour under the influence of different gains. This chapter discusses the results obtained from these tasks.

4.1 Cross-verification of Block Results

Once the block was implemented and the model was debugged, the next task was to make sure there are no logical errors and that the block behaves as expected. Ideally, one would want to validate the behaviour of the model with the behaviour of the actual system for the same inputs. However, this was beyond the scope of this project and hence the next easiest way to do something similar was to check that if for a known input-output combination, if the output of the inverse relationship block is connected to the input of the forward relationship block for a particular mapping, then these blocks can be considered to have cross validated. Additionally, wherever possible for the same task known values were used.

4.1.1 Verifying the Cable Length and Joint Angle Kinematic Model

To verify the cable length and joint angle mapping of the kinematic model of this system we use the Simulink model as shown below.

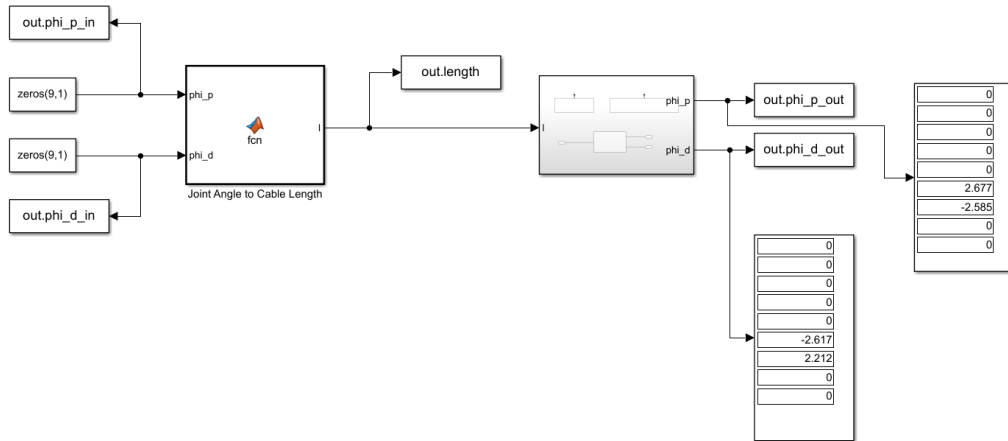


Figure 4.1: The figure shows the Simulink model used to verify the forward and inverse Cable Length and Joint angle kinematic model

We know that when, the joint angles are 0, then the length of cable between 2 links is $2d$. This known condition is used to verify the results obtained for the 2 blocks. It can be seen in the above figure that the for an input with all angles set to 0, the output on the other end mostly gives 0. However, there are two exceptions and this might stem from the fact that the given system is hyper-redundant. This means that even for the same change in cable length the robot may be in a different pose. However, it must be noted that on checking the values saved in `out.length`, this was observed to be $2d$ as expected.

4.1.2 Verifying the End-effector Position and Joint Angle Kinematic Model

To verify the end-effector position and joint angle mapping of the kinematic model of this system we use the Simulink model as shown below.

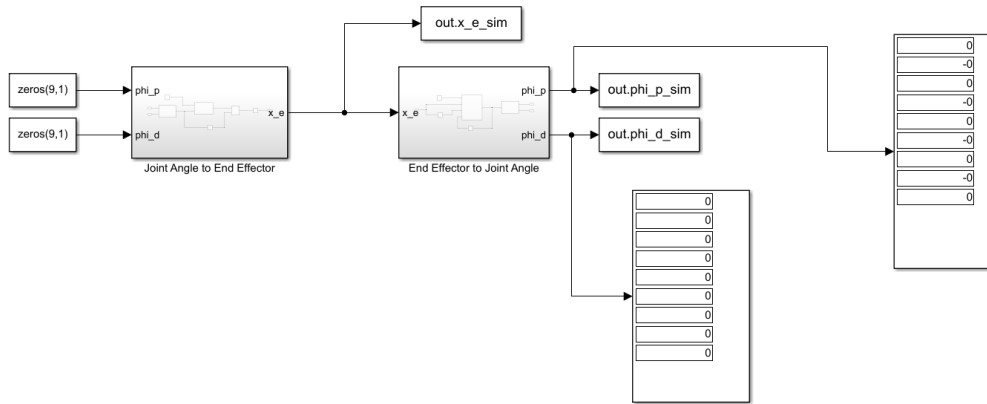


Figure 4.2: The figure shows the Simulink model used to verify the forward and inverse end-effector position and Joint angle kinematic model

Unlike in the previous section, here we do not have a know initial condition. However, it can be seen from the results obtained for this setup that the results at the input and the output are the same as desired. This should assure that the blocks perform as expected.

4.1.3 Verifying the Inverse Dynamics Block

The inverse dynamic relationships obtained from the Newton-Euler Recursive Algorithm was implemented in the block named `Newton-Euler Inverse Dynamics Block`, shown in the Fig. 4.3 below. To debug the block and observe the value at the output end of the Newton-Euler Recursive algorithm the model setup depicted by Fig.4.3 was implemented.

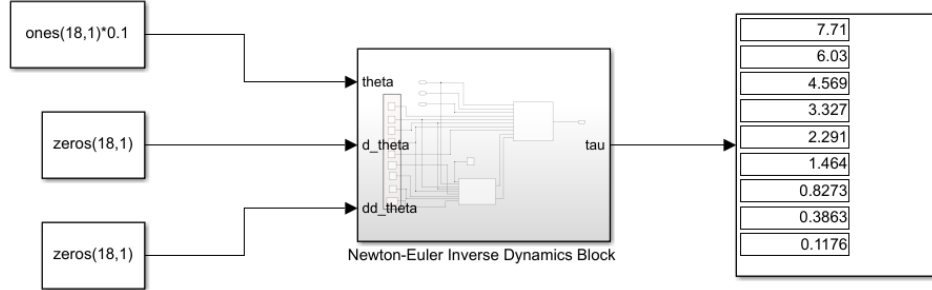


Figure 4.3: The Simulink model used to verifying the results obtained from the Newton-Euler Inverse Dynamics Block.

Additionally, since the same block was attempted to be used as the feed-forward signal. This model was used to test the output received when the input τ_{au_c} is made 0 to ensure the outputs are feasible torque values that can be then used as the feedforward signal. When the τ_{au_c} values are mapped to the respective cable tension forces then it was found that they meet the constraints specified by eq. (3.3) and hence it was concluded that this is could be a suitable choice as a feedforward signal as well.

4.2 Tuning the Controller with the Actuator Model

In the proposed control scheme, the actuator block includes the dynamic model of the maxon motors. In the actual system, the actuator is what can be directly controlled and so in the proposed scheme the signal fed into the PID controller is the feedback of the angular position of the motor. If we isolate just the actuator with the feedback loop and controller from the control scheme, then we get the model as shown in Fig. 4.4.

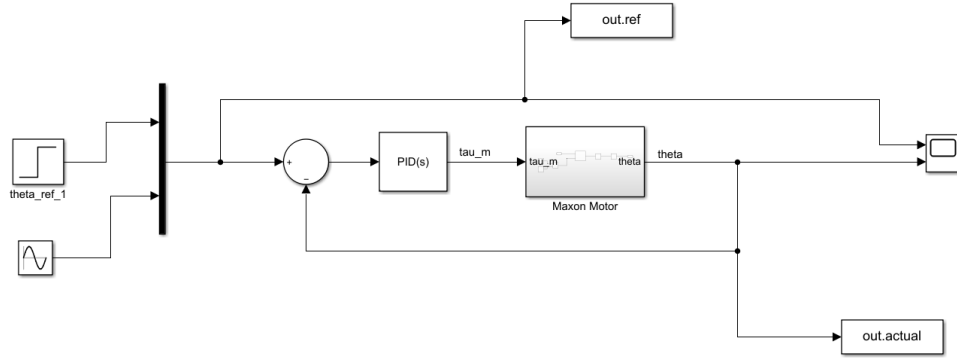


Figure 4.4: The Simulink model used for tuning the dynamic model of the motor.

The main motivation behind attempting to tune the controller for the actuator model separately is the fact that the gains obtained from this can be used as the starting point when the controller for the whole system needs to be manually tuned.

On tuning the PID controller with the dynamic model of the maxon motor, with a step input, a satisfactory response (with a very tiny overshoot at the beginning) as shown in Fig. 4.5a was observed when gain values were set as: $K_P = 0.92$; $K_I = 33$; $K_D = 0.005$;

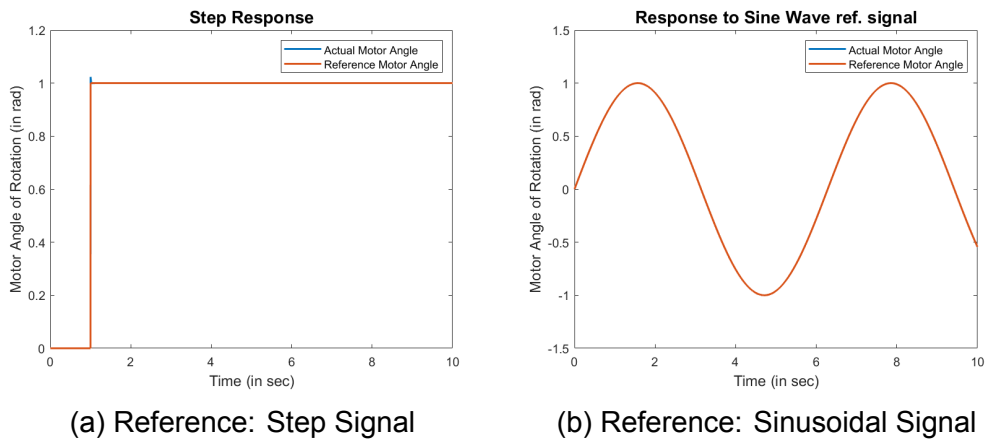


Figure 4.5: Response of the tuned motor block to a reference signal

Now that a satisfactory step response was obtained, to ensure that the controller works well for any input, the input was changed to a sinusoidal

wave and the response was observed. Fig. 4.5b shows the response obtained and confirms that the reference value is being tracked well.

4.3 Tuning the controller with the Forward Dynamic Block

After the forward dynamic block was made, to simply test the functionality of the block the model shown in the figure below was implemented. This model isolates the dynamic model block from the other complicated blocks in the proposed control scheme which means that its response to a simple feedback loop configuration can be studied to further understand the behaviour of the system. Note that, here the input reference is a $3N \times 1$ vector and the number of outputs is $2N \times 1$. Therefore, the feedback is adjusted to match the dimensions of the vector. This is because every 2 joints in between 2 links in the snake-arm are controlled by 3 cables each.

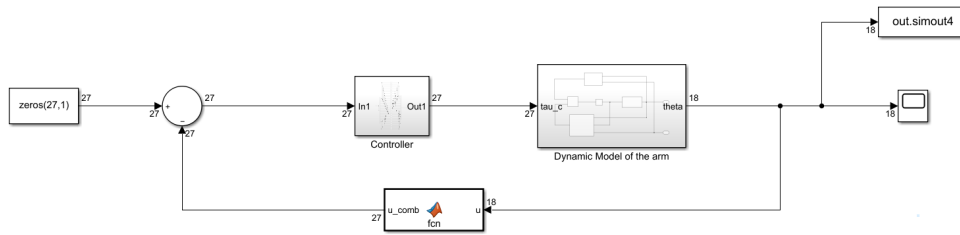
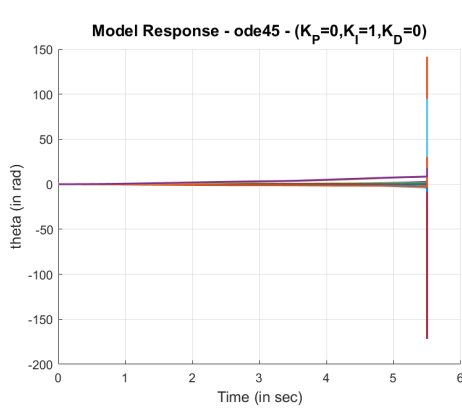
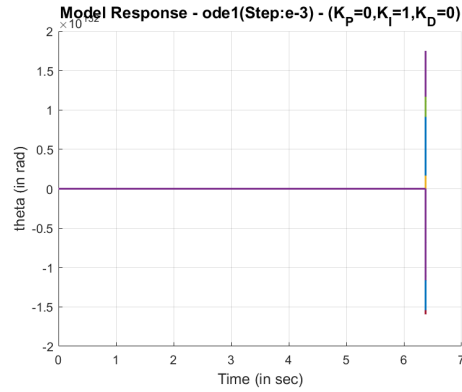


Figure 4.6: The simplified Simulink model used for tuning the controller with the dynamic model.

On trying to run the model, the first couple of attempts failed mid-way through the simulation suggesting unstable behaviour as shown in the Fig. 4.7. This was mainly because the control gains were not tuned yet.



(a) Solver: ode45 ; $K_P=0, K_I=1, K_D=0$



(b) Solver: ode1 (Step size= 10^{-3}) ; $K_P=0, K_I=1, K_D=0$

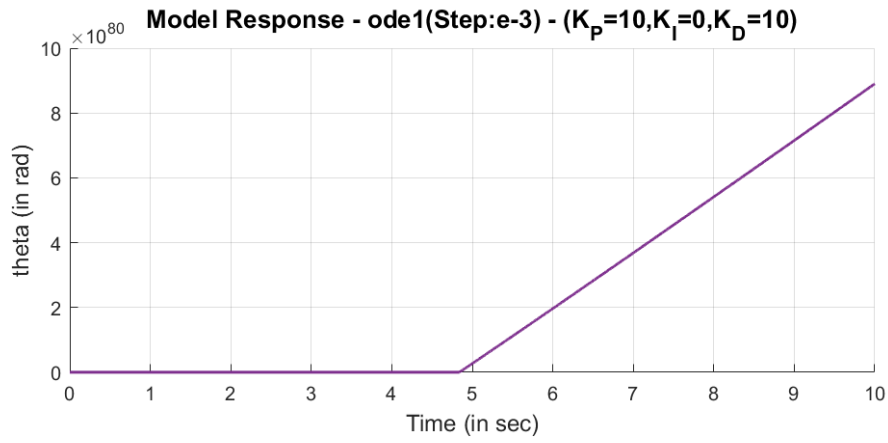
Figure 4.7: Unstable responses from the first attempts

An attempt was then made to tune the PID controller keeping the same solver, ode1 (Step size= 10^{-3}). The simulation time was set to 10s and an attempt was made at manually tuning the gains of the controller. Some of the best results obtained from this attempt, for different gain values, has been shown in Fig. 4.8. It can be seen here that $K_P = 100$; $K_I = 0$; $K_D = 50$ shows a smaller peak from among the three. However, since it was unstable, it could not be considered a suitable combination of gains either.

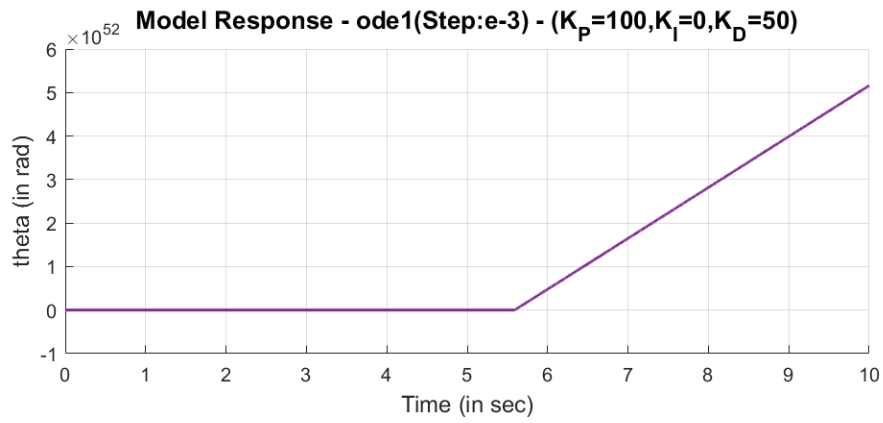
Note that at this stage, only a single PID controller was used for all the 27 inputs. This meant that the attempt at controlling the snake-arm was being made with the assumption that all the joints can be controlled by the same gain value.

So to make tuning more efficient, each input was assigned a separate controller. This meant that each controller had to be specifically tuned for the joint it was controlling. These were then manually tuned. When trying to control link 1, the best result obtained from among the different gain values that were tried was for: $K_P = 10$; $K_I = 0$; $K_D = 10$. Similar, tuning was to be done for the other joints, however due to time constraints tuning of all the links could not be completed on time.

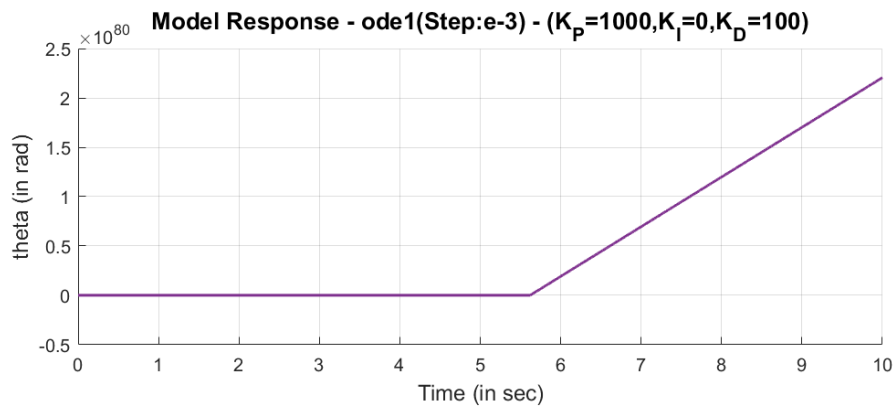
However, from the response with the gain values of the tuned controller associated to link 1, the following experiments were done and the observations made are discussed in the following subsections.



(a) Solver:ode1 (Step size= 10^{-3}) ; $K_P=0, K_I=1, K_D=0$



(b) Solver: ode1 (Step size= 10^{-3}) ; $K_P=0, K_I=1, K_D=0$



(c) Solver: ode1 (Step size= 10^{-3}) ; $K_P=0, K_I=1, K_D=0$

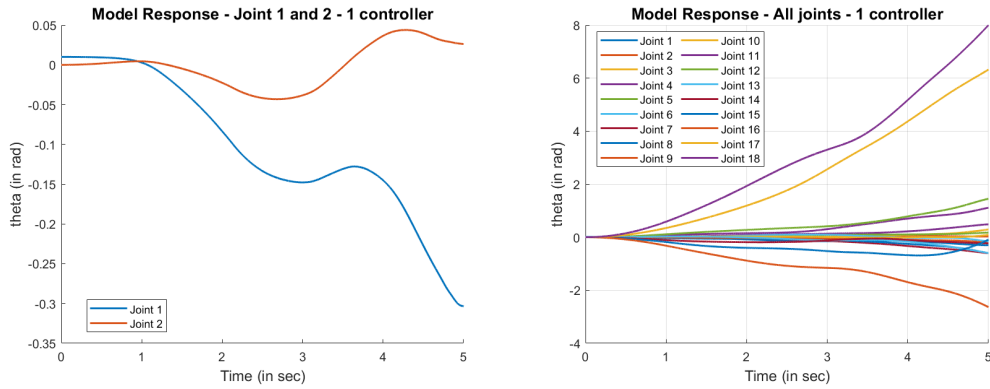
Figure 4.8: The responses obtained from attempting to tune a single PID controller that is assigned to all inputs for the snake-arm.

Comparing the use of one controller against three controllers for controlling a link

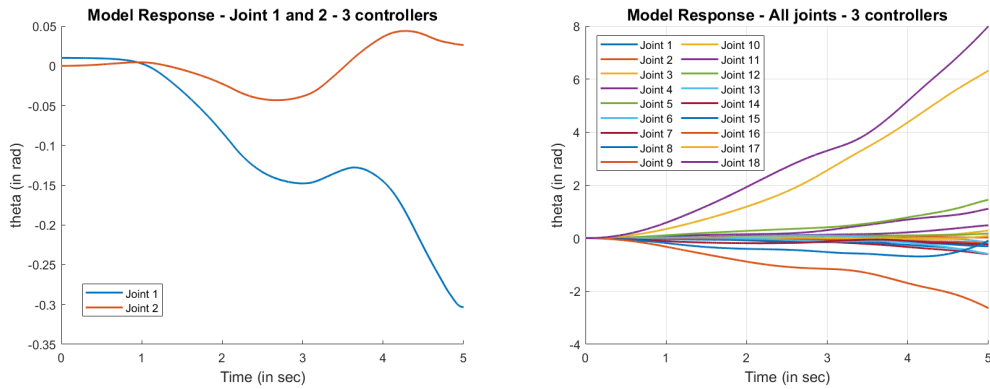
It is known that the 2 joints between adjacent links are controlled by 3 cables and this raised the question of whether tuning one controller would be enough to control a joint. So, a comparison was made by controlling the same joint with 1 controller and then 3 controllers of the same gain at the same time.

Fig. 4.9 below compares the plots obtained on controlling the first 2 joints and the last 2 joints with the same gain values and with the same solver (ode1; Step size: 10^{-3}).

From these plots, it can be observed that not much of a change was observed between the two cases. However, given that the results from the tuned controller are still not very robust it might not be appropriate to make a conclusion based just on this observation.



(a) Plot showing the response of joint 1 and 2 on being controlled by only 1 controller
(b) Plot showing the response of all other joints when joint 1 and 2 are being controlled by only 1 controller



(c) Plot showing the response of joint 1 and 2 on being controlled by 3 controllers
(d) Plot showing the response of all other joints when joint 1 and 2 are being controlled by 3 controllers at the same time.

Figure 4.9: Comparison of the responses of the dynamic block when the number of controllers controller a joint is increased

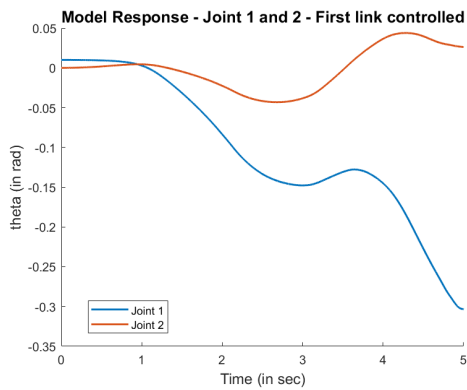
Comparing the plots obtained on controlling the first 2 joints and last 2 joints with the same gain values

An attempt is made here to try and understand if the same gain values will be effective in controlling different links. For this a comparison is made by controlling the last link with its controller taking the gain values that were tuned specifically for the first link.

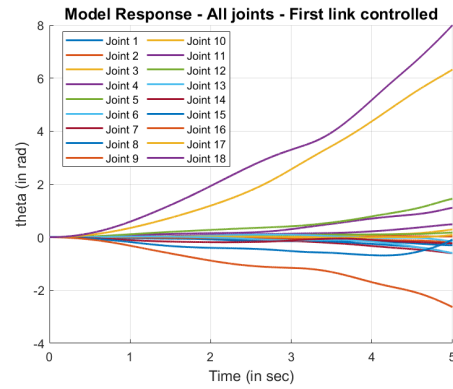
Fig. 4.10 below, compares the plots obtained on controlling the first 2 joints and the last 2 joints with the same gain values and with the same solver

(ode1; Step size: 10^{-3}).

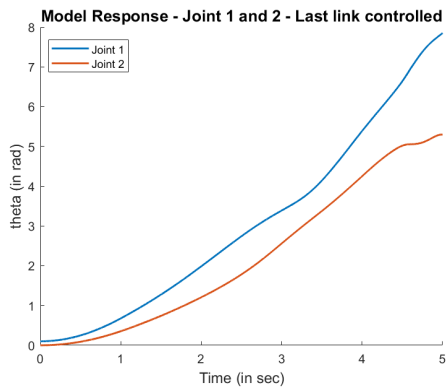
From these plots, it can be observed that the response of the first 2 joints are better for the given gain values in comparison to that of the last 2 links. This is a clear indication of the fact that each controller has to be tuned separately for each joint which means that the initial attempt of using a single PID controller was insufficient. As expected, from the plots showing the responses of all joints, it can be clearly seen that depending on which joint is being controlled the joints adjacent to the links being controlled are the most affected while the others don't experience much of a perturbation from the movement of the other links.



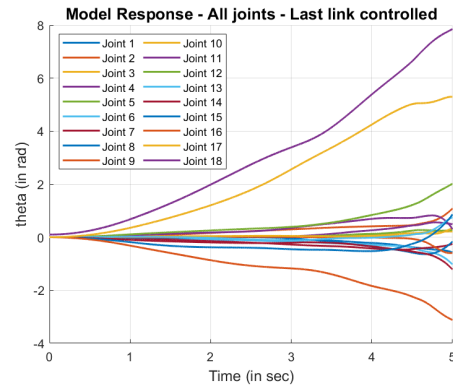
(a) Plot showing the response for first 2 joints when they are directly controlled



(b) Plot showing the response for all joints, when only the first 2 joints are controlled



(c) Plot showing the response for last 2 joints when they are directly controlled



(d) Plot showing the response for all joints, when only the last 2 joints are controlled

Figure 4.10: Comparison of the responses of the dynamic block when only two different links are controlled at at time

4.4 Analysis of Robustness

Since the tuning of the controller could not be completed within the given time constraints, the task associated to evaluating the control scheme's robustness could not be achieved. However, if the controller was tuned the analysis of the robustness of the control scheme would have been done based on two tests.

The first analysis would have been based on the effect of changing load at the nose of the snake-arm. The effect of change in load can be incorporated into the dynamic model by considering the term $J^T F_e$ in eq.(2.12) into the dynamic model equation. If the Newton-Euler recursive algorithm was to be used instead, F_e should be set as the terminal condition which has been otherwise set as 0 in the above analysis. Thus in this manner the robustness of the control scheme to changing load could have been analysed by comparing the response in the presence of different F_e values at the nose of the manipulator.

The second analysis would have been based on the effect of external disturbances. In the Newton-Euler recursive algorithm this would have been incorporated as an additional force term, F_{ext} acting at the centroid of the link. Similar to the previous analysis, if the controller would have been tuned, the robustness of the control scheme to changing environments (presence of external disturbances) could have been analysed by comparing the response in the presence of different F_{ext} values acting on individual links.

Chapter 5

Conclusions

This project aimed at developing a dynamic model for a robotic snake-arm manipulator at Here East as an attempt towards contributing to the modelling and control research effort undertaken for this system. The project also aimed at designing a control scheme that would facilitate accurate end-effector position control so that precise trajectory tracking could be achieved. For this, the control system design and the dynamic model was developed, implemented and evaluated using Simulink.

The previous chapters discussed the derivation of the mathematical model, the proposed baseline control scheme and the preliminary results obtained from the attempt. The individual blocks of the mathematical model were developed, implemented and cross-verified mathematically. However, progress was slow and so only preliminary results for the dynamic model could be achieved as discussed in Chapter 4. This meant that only some of the objectives as stated in Chapter 1 could be achieved.

There were several reasons for this. It can be seen in the Gantt Chart in Appendix A, that was submitted along with the proposal that, not much time was allocated for identifying and developing the mathematical model of the system. This was mainly because it was assumed from the work undertaken during summer that the dynamic model would be similar in nature to that of a wheelless-snakebot that moves on the ground. For such a ground-hugging snakebot the mathematical model was derived from first principles of motion and did not involve a lot of background theory. However, the model for this manipulator required a lot more understanding about the theory of dynamic modelling of robotic manipulators in general and the different algorithms/approaches associated to it in order to recognize the equations. This was realised a month into the project and the learning curve

cost time, much more than anticipated. The lack of suitable resources discussing the dynamic model and its simulation also contributed to the slower progress in the initial phase. Learning the theory and attempting simpler examples to develop a better understanding of the subject, itself took a month or two of term time.

While the other commitments during term time were also taken into consideration while proposing the objectives, the underestimation of the learning curve meant that project demanded more time than what was initially anticipated. However it can be said that, if the dynamic model and the background theoretical knowledge required to recognise the equations been identified prior to the start of the project during summer, it would have given the head start that was required to be able to achieve all the objectives with the constraints of term time.

In order to simplify the situation, midway through the project an attempt had been made to adopt a simpler approach that involved considering the CDHRM system like just any other general robotic manipulator but with alternate links of length 0 to incorporate the fact that there are two revolute joints between each link. This meant that the standard functions and the general dynamic block provided by the Robotics Toolbox in MATLAB could be used to develop the control scheme. However as this was attempted, it was realised that for the unique structural design of the CDHRM using the general dynamic block is not a feasible option as this robotic manipulator is a passive system that is indirectly driven by cable tension forces. This meant that to formulate the dynamic response of the system the cable tension forces had to be intrinsically included and this was not an option provided by the function available via the Robotics Toolbox. Hence, this approach was abandoned.

5.1 Future Work

While this report discusses a suitable mathematical model and proposes a scheme that could be the baseline configuration for the control system of the snake arm, there is more work to be done to be able to achieve the bigger aims of this project.

Even though the equations of the mathematical model were cross-verified mathematically, the behaviour of the dynamic model is yet to be validated against the behaviour of the actual system. If the dynamic model fails to represent the actual system, then may be an alternate modelling approach

could be to use functionalities available within MATLAB like Simscape to use the input and output plant variables recorded from the actual system to develop a dynamic model.

It can be realised that the proposed control scheme also has a lot of room for improvement. For starters, a better control scheme could be something that incorporates feedback from the tracking error of the end-effector position as input to the controller along with the feedback from the angular position of the actuator itself. Another improvement could also be to incorporate force control to regulate the cable tension forces directly, as that is the driving force of the passive robotic arm under consideration that ultimately determines its resulting physical configuration.

Lastly, another aspect about the project that can be explored is the dynamic parameter estimation. Initially, the idea was to use dynamic parameters from one of the literature sources that describes the simulation of a similar system. However, all the literature that was identified, associated to dynamic modelling of a CDHRM, performs experiments directly on the actual system and often did not discuss simulation of the dynamic model. This meant that the equations for the model as described in this report were compiled from different sources. So, while mainly all parameters were defined from [21], some of the dynamic parameters (that were independent) were assigned based on what was found from different sources as a single source covering all parameters could not be identified. Future attempts can possibly rectify this by using the parameters from the actual system. However it is a known issue that even this could result in parameter estimation errors. A common approach to reducing such estimation errors involve incorporating a regressor term like $Y(\Theta, \dot{\Theta}, \ddot{\Theta})$ in the dynamic model to compensate these errors. This might be something that could be considered given that the estimation of parameters like Inertia tensor tend to introduce errors given the structural design of the manipulator. To improve the performance further, even more sophisticated methods involving learning techniques could be considered.

To conclude, while this report proposes a suitable mathematical model that represents the actual system under consideration, the proposed control scheme needs improvement. From the discussion on preliminary results, it is evident that more work needs to be done in order to obtain satisfactory results. More control schemes need to be designed and tested in the future until a suitable one that meets the control objective is developed.

Appendix A

Project Gantt Chart (as submitted with proposal)

This appendix discusses the Project work plan and Gantt Chart as submitted in the project proposal. The project will mainly have four phases - *Preparatory phase*, *Simulation and Modelling*, *Testing and Improving results* and *Analysis and Compilation of Results*

- **Preparatory Phase:** The first month and a half will involve preparatory phase during which through literature review several suitable mathematical models will be identified and the model that best represents the physical snake-arm will be implemented. During this phase the control objectives of the snake arm will also be clearly investigated and will be defined mathematically. Work undertaken over summer proved that the approach of mimicking a biological snakebot to improve reach and span of the robot will not be suitable, as controlling such a snakebot poses a lot of challenges. Even though extensive research over the years have led to improved results, the level of accuracy of the tracking of the controlled variable is lower than what is required for a task like air wing inspection. The risk of a snake-arm making contact with the structures due to incorrect path following can be very unsuitable for this scenario. Also, given that the OC Robotics snake-arm is more like a tethered continuum robot moving in free space, the viscous friction model adopted by most authors to describe the dynamic model of a wheel-less ground-hugging robot snakebot won't be suitable. Therefore, the initial approach when modelling the snake-arm the concept of continuum robots may be used the most.
- **Simulation and Modelling:** The next phase of the project will then

involve implementing the mathematical model, simulating the control system on Simulink and obtaining the initial results. The tracking error of the parameters (specially speed, position and orientation) will be analysed and efforts will be made to improve these results. It is expected that at this stage, an iterative testing method would be adopted to improve the model until desired tracking accuracy is obtained.

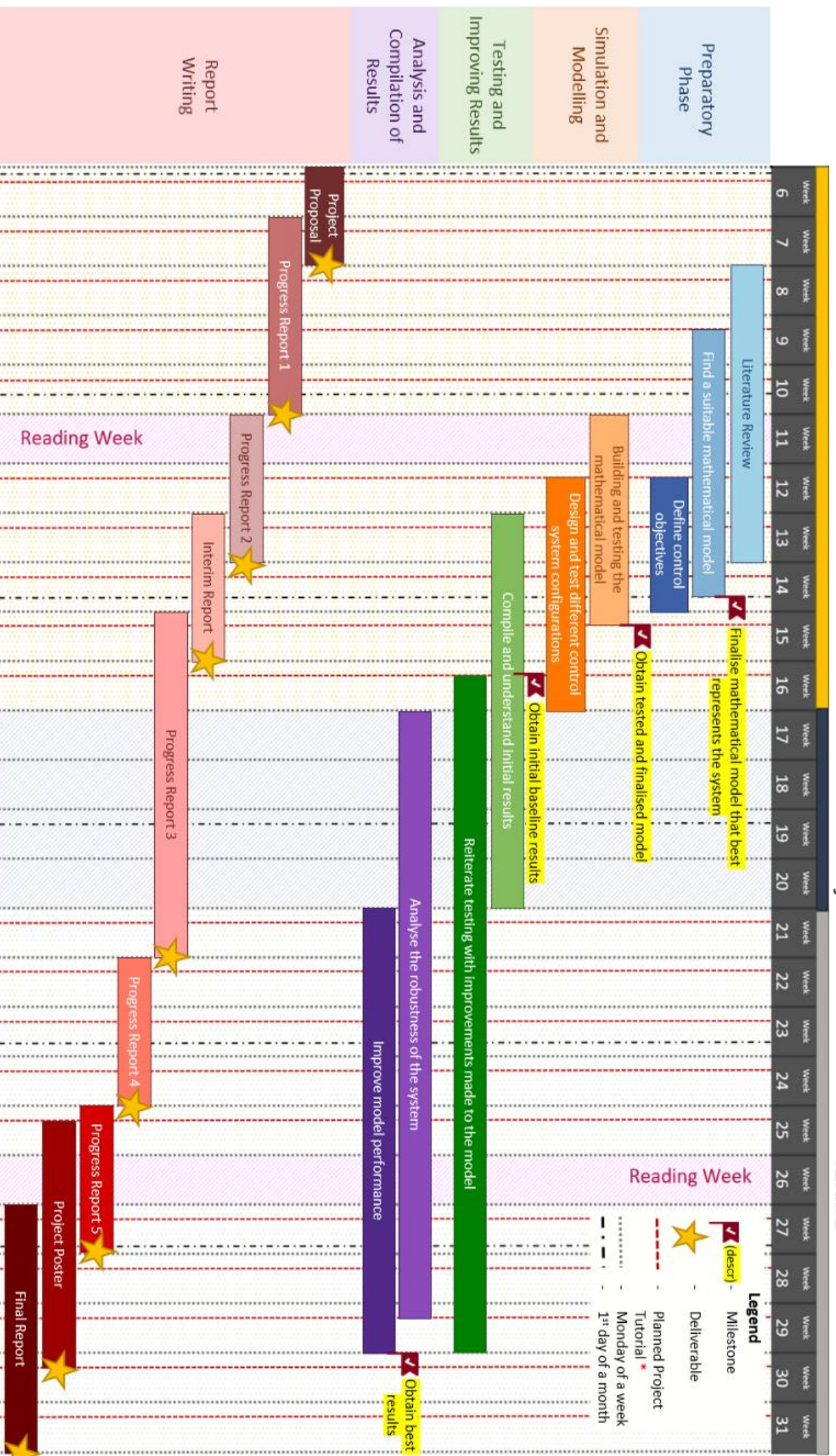
- **Testing and Improving Results:** From the start of the second phase to the end of the project, results will be improved and tested iteratively until desired results are achieved. Although this has been specified as a separate phase it represents the steps that are part of the second and the last phase.
- **Analysis and Compilation of Results:** Once satisfactory results are obtained; the final phase of the project will focus on testing the robustness of the system in varied conditions. It is expected that the payload attached to the head will change depending on the optical metrology technique for which it is being used. Further, it might be performing measurements in environments where external disturbances are no longer negligible. So, for these scenarios, tests will be performed to analyse the robustness of the system to these disturbances. Results obtained will be reported systematically and periodically through progress reports that are roughly due once in every three weeks since the start of the project.

Project Gantt Chart

TERM 1

Christmas Holidays

TERM 2



* - Planned tutorials are subject to change based on supervisors availability

Appendix B

List of Notations

Notation	Definition
Subscripts	
N	Number of joints (N=9)
n	Subscript that can take values from the set $n = \{1, 2, 3, \dots, N\}$ (used for joints)
k	Subscript that can take values from the set $k = \{1, 2, 3, \dots, 3N\}$ (used for motors, cables)
i	Subscript that can take values from the set $i = \{0, 1, 2, \dots, N\}$ (used for links)
Parameters	
d_{drum}	Diameter of the cable drum
p	Pitch of the cable drum
m	Mass of a link
l_{link}	Length of the link
Kinematic Model	
$\theta_{motor,k}$	Angle of rotation
L_k	Length of the cable k
$\{P_n\}$	Frame of reference fixed at the centre of the proximal disc
$\{U_n\}$	Frame of reference fixed at the centre of the universal joint
$\{D_n\}$	Frame of reference fixed at the centre of the distal disc

$O_{P,n}$	Origin of frame of reference $\{P_n\}$
$O_{U,n}$	Origin of frame of reference $\{U_n\}$
$O_{D,n}$	Origin of frame of reference $\{D_n\}$
ζ_P	Axis of rotation of joint attached to proximal disc
ζ_D	Axis of rotation of joint attached to distal disc
X_U	X-axis of reference frame fixed at center of the universal joint
Y_U	Y-axis of reference frame fixed at center of the universal joint
Z_U	Z-axis of reference frame fixed at center of the universal joint
X_P	X-axis of reference frame fixed at center of the proximal disc
Y_P	Y-axis of reference frame fixed at center of the proximal disc
Z_P	Z-axis of reference frame fixed at center of the proximal disc
X_D	X-axis of reference frame fixed at center of the distal disc
Y_D	Y-axis of reference frame fixed at center of the distal disc
Z_D	Z-axis of reference frame fixed at center of the distal disc
${}^D A_P$	Transformation matrix from $\{D_n\}$ to $\{P_n\}$
${}^D A_U$	Transformation matrix from $\{D_n\}$ to $\{U_n\}$
${}^U A_P$	Transformation matrix from $\{U_n\}$ to $\{P_n\}$
$\phi_{p,n}$	Joint angle defined w.r.t frame $\{U_n\}$ about axis of rotation ζ_P
$\phi_{d,n}$	Joint angle defined w.r.t frame $\{U_n\}$ about axis of rotation ζ_D
$l_{k,n}$	Length of cable between 2 discs
$H_{p,k,n}$	Vector defining the position of the hole on proximal disc w.r.t the local frame
$H_{d,k,n}$	Vector defining the position of the hole on distal disc w.r.t the local frame
$\delta_{n,k}$	Angular position of the hole
M	Link number at which cable k terminates
J_d	Jacobian matrix
J_d^+	Moore-Penrose pseudo-inverse of J

θ_n	Angle of rotation along Z_{n-1} to align X_{n-1} and X_n
α_n	Angle of rotation along X_n to align Z_{n-1} and Z_n
r_n	Distance between origins in the X_n direction
d_n	Distance between origins in the Z_n direction
${}^{n-1}A_n$	Homogeneous transformation matrix for adjacent frames
\dot{x}_{end}	Velocity of the end effector position
$\dot{\Theta}$	Angular velocity of joint angles
${}^{n-1}R_n$	Rotational matrix
${}^{n-1}P_n$	Position vector
Dynamic Model	
$M(\Theta)$	Inertia matrix
$C(\Theta, \dot{\Theta})$	Centrifugal and Coriolis forces
F_v	Coefficient of viscous friction matrix
$g(\Theta)$	Moment generated by the presence of gravity
u	Actuation torque
F_e	Force and moment vector exerted by the end-effector on the manipulator
${}^{2n}F_{g,n}$	Link gravitational force
g	Acceleration due to gravity
${}^{2n}F_n$	Link inertia force
${}^{2n}\dot{v}_n$	Linear acceleration of center of mass
${}^{2n}T_n$	Link inertia moments
${}^{cn}I_n$	Inertia tensor
${}^{2n}\omega_n$	Angular velocity of the link w.r.t $\{F_{2n}\}$
${}^{2n}\dot{\omega}_n$	Angular acceleration of the link w.r.t $\{F_{2n}\}$
$\mathbf{F}_{tp,n,k}$	Cable tension force vector acting beyond proximal disc hole
$\mathbf{F}_{td,n,k}$	Cable tension force vector acting beyond distal disc hole

$\mathbf{e}_{tp,n,k}$	Direction of cable tension force acting beyond proximal disc hole
$\mathbf{e}_{td,n,k}$	Direction of cable tension force acting beyond distal disc hole
$\mathbf{L}_{p,n,k}$	Cable position vector
$\mathbf{L}_{d,n,k}$	Cable position vector
${}^{2n}\mathbf{r}_{d,n,k}$	Coordinates of the holes $H_{d,n,k}$ defined w.r.t $\{F_{2n}\}$
${}^{2n}\mathbf{r}_{p,n,k}$	Coordinates of the holes $H_{p,n,k}$ defined w.r.t $\{F_{2n}\}$
$\mathbf{F}_{cp,n,k}$	Cable contact force (proximal disc)
$\mathbf{F}_{cd,n,k}$	Cable contact force (distal disc)
$\mathbf{F}_{ce,n,k}$	Equivalent cable contact force
$\{F_{2(n-1)}\}$	Reference frame defined at joint n
P_n	Position vector of joint n with respect to base frame
$P_{n,n+1}$	Position vector from joint n to joint $n + 1$
P_{cn}	Position vector of the centroid of link n with respect to $\{F_{2n}\}$

Appendix C

Supporting Explanation to Some Derivations

C.1 Derivation of Equation (2.5)

The derivation of eq. (2.5) is based on the design of the discs at either end of each link which indicates the angular position of the hole w.r.t the center of the disc. The equation has three terms and was defined as:

$$\delta_{n,k} = \begin{cases} \frac{120^\circ(k-1) + 120^\circ}{(N+1)} \times M - 3^\circ & ; n = \text{odd} \\ \frac{120^\circ(k-1) + 120^\circ}{(N+1)} \times M + 87^\circ & ; n = \text{even} \end{cases} \quad (\text{C.1})$$

From Fig. 2.6, we know there are $3N$ holes that are placed equidistant from each other. We also know that the three cables that terminate at a given link n , are equidistant from each other. This means dividing 360° by 3 we get that, these three cables have to be 120° apart. So for a given cable k , this can be indicated as $120^\circ(k-1)$. This gives us the first term in the equation.

So now that from term 1 we know which of the three 120° regions (divided by the dotted line in Fig. 2.6) the hole is in, next we need to find the angular position of the hole within the region. We know that, cable k terminates at joint number M , given by $M = \text{floor}((k-1)/3) + 1$. In other words, we can say that the M^{th} hole in one of these regions associate to the cables that terminate at link M . To place N holes at equidistant positions, the 120° region needs to be divided into $N+1$ regions. Therefore, the position of

cable k that terminates at link M in that region can be given by $\frac{120^\circ}{N+1} \times M$. This gives us the second term.

Note that here, the holes associated to the same cable have to be aligned in both discs despite the fact that there is an offset of 90° between the two discs. This is possible if the position of the holes were multiples of 9. However, this isn't the case. Our first hole position is at 12° according to the above terms. So to ensure that the holes are aligned, the offset is split between the 2 such that the hole position of the first hole on the first disc becomes a multiple of 9. The easiest way to achieve this is to make the position of the hole at the nearest multiple of 9. This can be done taking away 3° from the first disc and therefore to balance it out the same is done to the adjacent disc i.e. the offset of 90° is now split between the two discs as 87° and 3° . This way the first hole position on the first disc becomes 9° and hence it assures that the 2 are aligned. This explains the need for the third constant term in the equation comes from.

Therefore, summing up the three terms we get eq. (2.5).

C.2 Derivation of Equation (3.6) and (3.7)

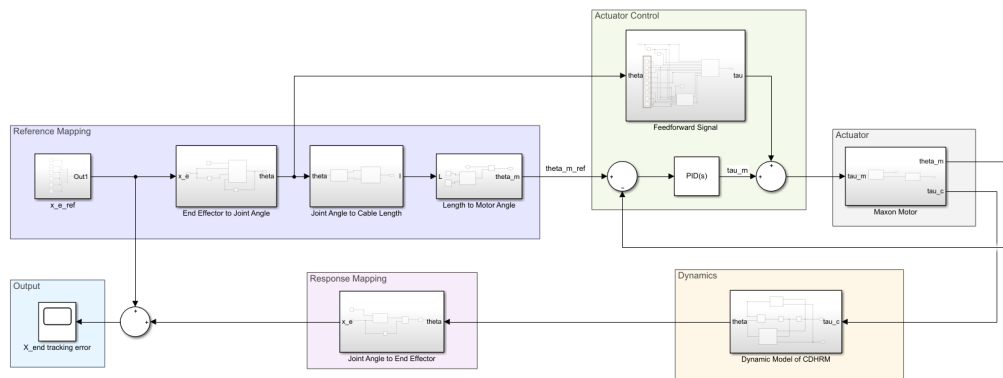
The derivation of these equations are backed by the fact that by setting some of the inputs explicitly as 0 or 1, we can eliminate some terms, isolate coefficients and the resulting output would give you the desired term based on the fact that the 2 forms of the dynamic model are equivalent since they represent the same model. A clear explanation of this approach can be found in [34] and [31].

Appendix D

Simulink Implementation

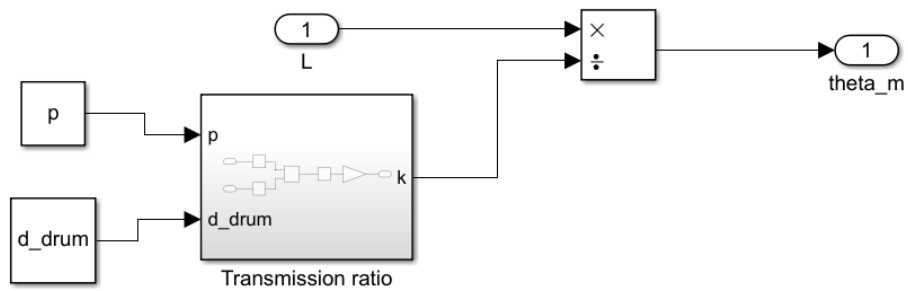
In this chapter of the appendix, block diagrams (screenshots of implementation in Simulink) of the different subsystems within blocks have been presented for reference.

D.1 The Proposed Control Scheme



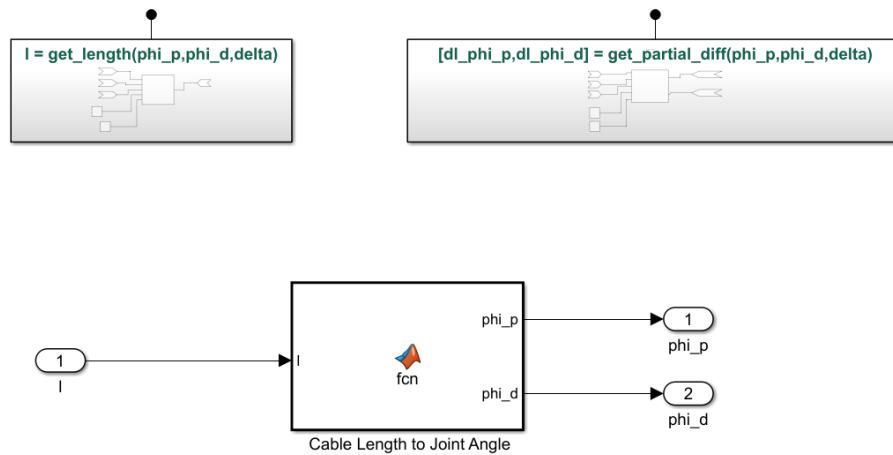
D.2 Kinematic Model

D.2.1 Cable Length to Motor Angle

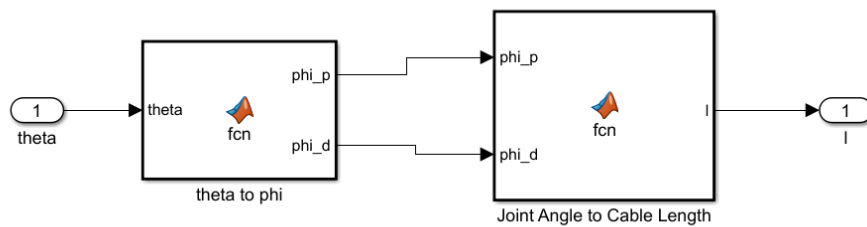


D.2.2 Cable Length \leftrightarrow Joint Angle

Cable Length to Joint Angle

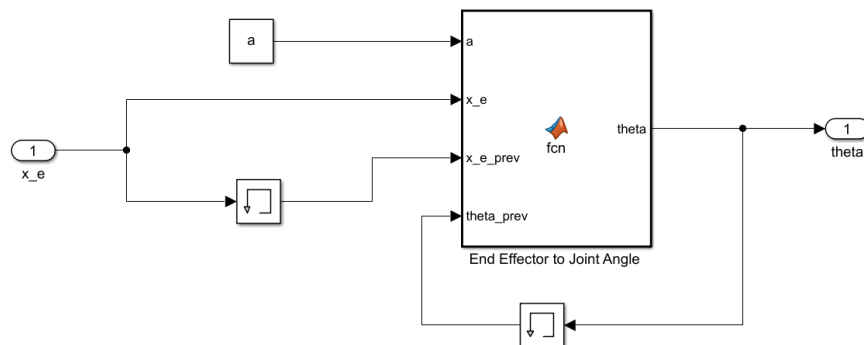


Joint Angle to Cable Length

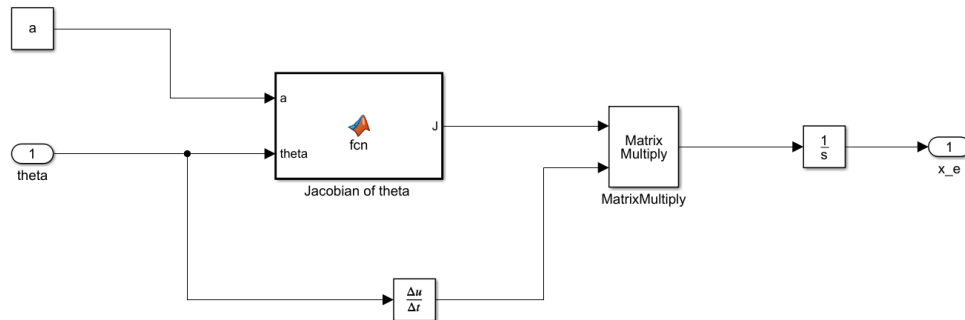


D.2.3 End-effector position \leftrightarrow Joint Angle

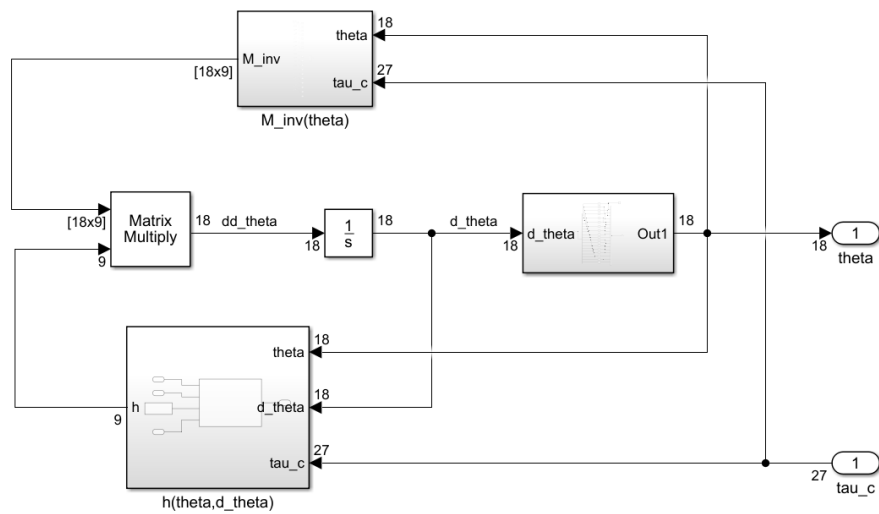
End-effector position to Joint Angle



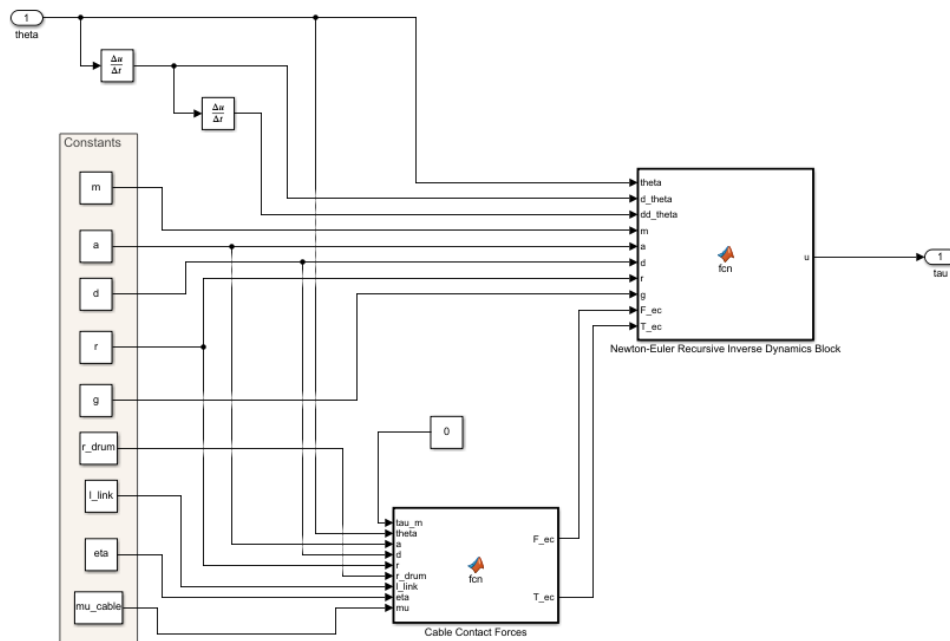
Joint Angle to End-effector position



D.3 Dynamic Model (Forward Dynamics)

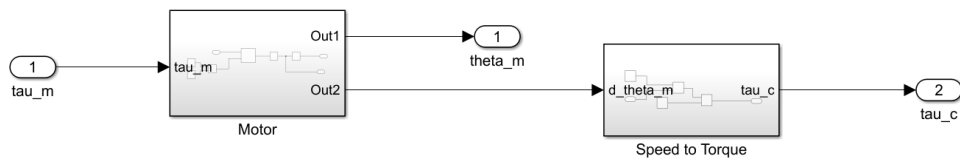


D.4 Feedforward Block (Inward Dynamics)

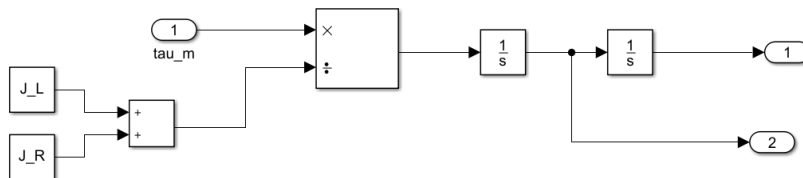


D.5 Actuator Block

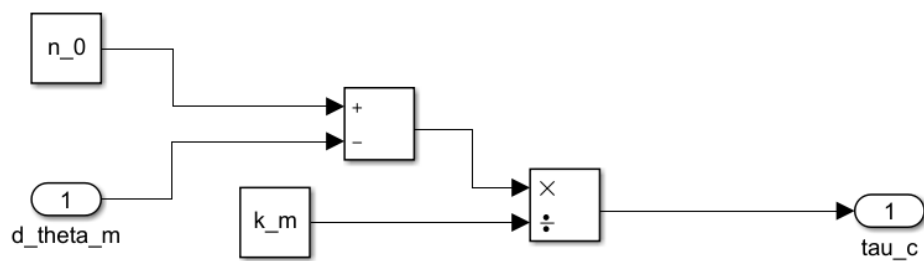
D.5.1 Inside the Actuator Block



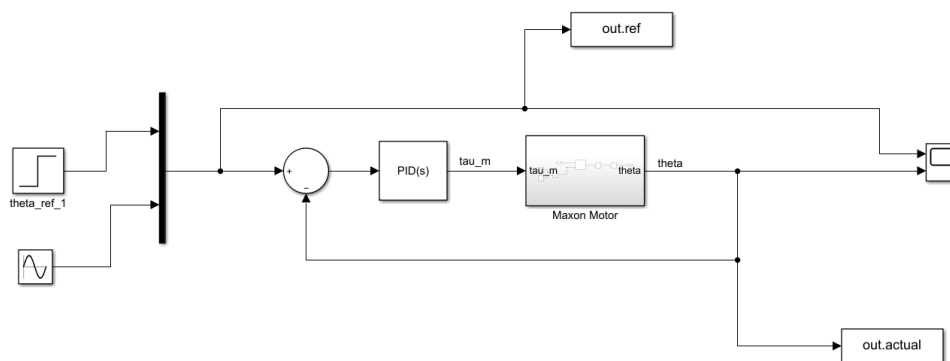
D.5.2 Modelling the Maxon Motor



D.5.3 Converting Speed to Torque



D.5.4 Tuning the Actuator



Appendix E

MATLAB Code

This chapter of the appendix lists the code associated to the some of main blocks of the proposed control scheme that used MATLAB functions. All the code and Simulink models associated to the work undertaken as part of this project can be found in:

<https://github.com/Aru2612/BEng-Project-Control-of-CDHM>

E.1 Parameter Definition

This script defines the dynamic parameters and constants for the model. It is specified in the `InitFcn()`, and therefore is called automatically in the background every time the model is run.

```
1 % This script is run as InitFcn by the Simulink model.
2 % clear all;
3
4 %% Physical parameters of CDHRM
5 % Number of links
6 N_links=10;
7 % Number of joints
8 N=N_links-1;
9 % Distance between 2 links
10 d=23/2000;
11 % Distance between 2 joints
12 a=140/1000;
13 % Radius of disc
14 r=20.5/1000;
15 % Mass of the link
16 m=0.2;
```



```

17 % Length of the link
18 l_link=a-2*d;
19 %% Motor Characteristics : Maxon Motor 110063
20 % Transmission Efficiency
21 zeta = 0.8;
22 % Rotor Moment of Inertia
23 J_R = 0.864 * 10e-7;
24 % Load Moment of Inertia
25 J_L = 0.014 * 10e-7;
26 % Transmission efficiency
27 eta=0.39;
28 % Winch Transmission ratio
29 k_trans=0.36;
30 % Radius of the drum
31 r_drum=0.042;
32 % Pitch
33 p=8/1000;
34 % Speed-torque Gradient
35 k_m=2560*100*pi/3;
36 % No load speed
37 n_0=9620*pi/30;
38 %% Physical constants
39 % acceleration due to gravity
40 g = 9.81;
41 % Coefficient of friction
42 mu_cable=0.17;
43 % Steel to steel friction
44 zeta_fr = 0.09;

```

E.2 Kinematic Model: Cable Length - Joint Angles

E.2.1 Cable Length to Joint Angles

```

1 function [phi_p,phi_d] = fcn(l)
2 N=9;
3 d=23/2/1000;
4 r=20.5/1000;
5 error_tol=1/1000;
6 phi_d=zeros(N,1);
7 phi_p=zeros(N,1);
8 % Numerical iteration to find solution
9 for n=1:N
10     % Get delta

```

```

11     if rem(n,2)==0
12         delta=[(12*(round(((3*n)-1)/3)+1)+120*((3*n)-1)+90)*pi
13         /180;...      (12*(round(((3*n-1)-1)/3)+1)+120*((3*n-1)-1)+90)*
14         pi/180;...      (12*(round(((3*n-2)-1)/3)+1)+120*((3*n-2)-1)+90)*
15         pi/180];
16     else
17         delta = [(12*(round(((3*n)-1)/3)+1)+120*((3*n)-1))*pi
18         /180; ...      (12*(round(((3*n-1)-1)/3)+1)+120*((3*n-1)-1))*pi
19         /180; ...      (12*(round(((3*n-2)-1)/3)+1)+120*((3*n-2)-1))*pi
20         /180];
21     end
22     l_d=l(3*n-2:3*n);
23     l_d=l_d/N;
24     phi_j=[0;0];
25
26     for j=0:500
27         % Calculate l_j
28         l_3n_0=get_length(phi_j(1),phi_j(2),delta(1));
29         l_3n_1=get_length(phi_j(1),phi_j(2),delta(2));
30         l_3n_2=get_length(phi_j(1),phi_j(2),delta(3));
31         l_j=[l_3n_0;...
32             l_3n_1;...
33             l_3n_2];
34         % Calculate error
35         delta_l=l_d-l_j;
36         % Check if the error is within tolerance
37         if (delta_l(1)<=error_tol) && (delta_l(2)<=error_tol) &&
(delta_l(3)<=error_tol)
38             phi_d(n)=phi_j(1);
39             phi_p(n)=phi_j(2);
40             break;
41         end
42         % Calculate the Jacobian
43         [J_1_1 J_1_2] = get_partial_diff(phi_j(1),phi_j(2),delta
44         (1));
45         [J_2_1 J_2_2] = get_partial_diff(phi_j(1),phi_j(2),delta
46         (2));
47         [J_3_1 J_3_2] = get_partial_diff(phi_j(1),phi_j(2),delta
48         (3));
49         J=[[J_1_1 J_1_2];...
50           [J_2_1 J_2_2];...
51           [J_3_1 J_3_2]];
52         J_inv=pinv(J);

```

```

50     %Calculate error
51     delta_phi= J_inv*delta_l;
52     % Use error to estimate delta_theta
53     phi_j=phi_j+delta_phi;
54
55     if j==10000
56         phi_d(n)=NaN;
57         phi_p(n)=NaN;
58     end
59 end
60 end

```

Functions in this block

```

1 function l = fcn(phi_d,phi_p,delta,d,r)
2 l = sqrt((d*sin(phi_d)+r*cos(phi_d)*cos(delta)-r*cos(delta))^2 + ...
3         (r*cos(delta)*sin(phi_p)*sin(phi_d)+r*cos(phi_p)*sin(
4         delta)-d*sin(phi_p)*cos(phi_d)-r*sin(delta))^2 + ...
5         (r*sin(phi_p)*sin(delta)-r*cos(delta)*cos(phi_p)*sin(
6         phi_d)+d*cos(phi_p)*cos(phi_d)+d)^2);
7
8 function [dl_phi_p,dl_phi_d] = fcn(phi_p,phi_d,delta,d,r)
9 dl_phi_p =(2*(r*cos(phi_p)*sin(delta) - d*cos(phi_d)*sin(phi_p)
10 + ...
11         r*cos(delta)*sin(phi_d)*sin(phi_p))*(d + ...
12         d*cos(phi_d)*cos(phi_p) + r*sin(delta)*sin(phi_p) -
13         ...
14         r*cos(delta)*cos(phi_p)*sin(phi_d)) + ...
15         2*(d*cos(phi_d)*cos(phi_p) + r*sin(delta)*sin(phi_p)
16         - ...
17         r*cos(delta)*cos(phi_p)*sin(phi_d))*(r*sin(delta) +
18         ...
19         d*cos(phi_d)*sin(phi_p) - r*cos(phi_p)*sin(delta) -
20         ...
21         r*cos(delta)*sin(phi_d)*sin(phi_p)))/(2*((d + ...
22         d*cos(phi_d)*cos(phi_p) + r*sin(delta)*sin(phi_p) -
23         ...
24         r*cos(delta)*cos(phi_p)*sin(phi_d))^2 + (d*sin(phi_d)
25         - ...
26         r*cos(delta) + r*cos(delta)*cos(phi_d))^2 + (r*sin(
27         delta) + ...
28         d*cos(phi_d)*sin(phi_p) - r*cos(phi_p)*sin(delta) -
29         ...
30         r*cos(delta)*sin(phi_d)*sin(phi_p))^2)^(1/2));
31
32 dl_phi_d = -(2*(d*cos(phi_p)*sin(phi_d) + ...
33         r*cos(delta)*cos(phi_d)*cos(phi_p))*...
34         (d + d*cos(phi_d)*cos(phi_p) + ...

```

```

24         r*sin(delta)*sin(phi_p) - ...
25         r*cos(delta)*cos(phi_p)*sin(phi_d)) - 2*(d*cos(
phi_d) - ...
26         r*cos(delta)*sin(phi_d))*(d*sin(phi_d) - r*cos(
delta) + ...
27         r*cos(delta)*cos(phi_d)) + 2*(d*sin(phi_d)*sin(
phi_p) + ...
28         r*cos(delta)*cos(phi_d)*sin(phi_p))*(r*sin(delta) +
...
29         d*cos(phi_d)*sin(phi_p) - r*cos(phi_p)*sin(delta) -
...
30         r*cos(delta)*sin(phi_d)*sin(phi_p)))/(2*((d + d*cos
(phi_d)*cos(phi_p) + ...
31         r*sin(delta)*sin(phi_p) - ...
32         r*cos(delta)*cos(phi_p)*sin(phi_d))^2 + (d*sin(
phi_d) - ...
33         r*cos(delta) + r*cos(delta)*cos(phi_d))^2 + (r*sin(
delta) + ...
34         d*cos(phi_d)*sin(phi_p) - r*cos(phi_p)*sin(delta) -
...
35         r*cos(delta)*sin(phi_d)*sin(phi_p))^2)^(1/2));

```

Φ to Θ Conversion

```

1 function theta = fcn(phi_p,phi_d)
2 N=9;
3 % Create theta
4 theta=zeros(2*N,1);
5 for n=1:N
6     theta(2*n-1:2*n)=[(-1)^(n+1)*phi_p(n);phi_d(n)];
7 end

```

Θ to Φ Conversion

```

1 function [phi_p,phi_d] = fcn(theta)
2 N=9;
3 % Create theta
4 phi_p=theta(1:2:2*N-1);
5 for n=1:N
6     phi_p(n)=(-1)^(n+1)*phi_p(n);
7 end
8 phi_d=theta(2:2:2*N);

```

E.2.2 Joint Angles to Cable Length

```

1 function l = fcn(phi_p,phi_d)
2 N=9;
3 d=23/2/1000;

```

```

4 r=20.5/1000;
5 l=zeros(3*N,1);
6 % Calculate transformation matrices
7 A=cell(N,1);
8 for n=1:N
9     s_p=sin(phi_p(n));
10    s_d=sin(phi_d(n));
11    c_p=cos(phi_p(n));
12    c_d=cos(phi_d(n));
13    A{n}=[ c_p      0      s_p      s_p*d;...
14           s_d*s_p  c_d     -s_d*c_p  -d*s_d*c_p ;...
15           -c_d*s_p  s_d     c_p*c_d   d*(c_p*c_d+1);...
16           0        0        0        1];
17 end
18 % Calculate the length of cable between joints
19 for k=1:3*N
20     l_k = 0;
21     % The link at which cable k terminates
22     M = floor((k-1)/3)+1;
23     for n=1:M
24         % Calculate the angular position of the hole on distal
25         and proximal
26         % discs
27         if rem(n,2)==0
28             delta_p=((120/(N+1))*(M)+120*(k-1)+87);
29             delta_d=((120/(N+1))*(M)+120*(k-1)-3);
30         else
31             delta_p=((120/(N+1))*(M)+120*(k-1)-3);
32             delta_d=((120/(N+1))*(M)+120*(k-1)+87);
33         end
34         % Calculate hole position vector on distal and proximal
35         discs
36         c_delta_p=cosd(delta_p);
37         s_delta_p=sind(delta_p);
38         c_delta_d=cosd(delta_d);
39         s_delta_d=sind(delta_d);
40         H_p=[r*c_delta_p;r*s_delta_p;0;1];
41         H_d=[r*c_delta_d;r*s_delta_d;0;1];
42         % Get the length of cable between the proximal and
43         distal disc at
44         % joint n
45         trans_H_p=A{n}*H_p;
46         l_k_n=trans_H_p(1:3)-H_d(1:3);
47         l_k = l_k + norm(l_k_n);
48     end
49 end
50 l(k)=l_k;
51 end

```

E.3 Kinematic Model: Joint Angles - End-effector position Mapping

E.3.1 Joint Angles to End-effector position

```

1 function J = fcn(a,theta)
2 N=9;
3 % Calculate D-H parameter table
4 DH_params=zeros(2*N,4);
5 for i=1:N
6     if rem(i,2)==0
7         DH_params(2*i-1:2*i,:)=[180+(theta(2*i-1)*180/pi) 90 0
0; 180+(theta(2*i)*180/pi) 0 a 0];
8     else
9         DH_params(2*i-1:2*i,:)=[(theta(2*i-1)*180/pi) 90 0 0; (
theta(2*i)*180/pi) 0 a 0];
10    end
11 end
12
13 % Calculate transformation matrix n_A_n+1 (A_wrt_prev) and store
    in cell
14 % array
15 A = cell(2*N,1);
16 for n=1:2*N
17     theta_i=DH_params(n,1);
18     alpha_i=DH_params(n,2);
19     r_i=DH_params(n,3);
20     d_i=DH_params(n,4);
21     A{n}=[cosd(theta_i) -sind(theta_i)*cosd(alpha_i) sind(
theta_i)*sind(alpha_i) r_i*cosd(theta_i);...
22         sind(theta_i) cosd(theta_i)*cosd(alpha_i) -cosd(theta_i)
*sind(alpha_i) r_i*sind(theta_i);...
23         0 sind(alpha_i) cosd(alpha_i)
d_i;...
24         0 0 0
1];
25 end
26
27 % Create a list of transformation matrices wrt base for all
    joints
28 A_till_prev_wrt_base=cell(2*N,1);
29 A_till_prev_wrt_base{1}=eye(4);
30 for n=1:2*N-1
31     A_till_prev_wrt_base{n+1}=A_till_prev_wrt_base{n}*A{n};
32 end
33 A_2N=A_till_prev_wrt_base{2*N}*A{2*N};
34 % Create Jacobian

```

```

35 J=zeros(6,2*N);
36 for n=1:2*N
37     J_v=cross((A_till_prev_wrt_base{n}(1:3,1:3)*[0;0;1]),(A_2N
(1:3,4)-A_till_prev_wrt_base{n}(1:3,4)));
38     J_w=A_till_prev_wrt_base{n}(1:3,1:3)*[0;0;1];
39     J(:,n)=[J_v;J_w];
40 end

```

E.3.2 End-effector position to Joint Angles

```

1 function theta = fcn(a,x_e,x_e_prev,theta_prev)
2 N=9;
3 error_tol=1/1000;
4
5 theta=zeros(2*N,1);
6
7 theta_j=zeros(2*N,1);
8
9 %Numerical iteration to find solution
10 for j=0:10000
11     % Calculate x_e_j
12     % Calculate D-H parameter table
13     DH_params=zeros(2*N,4);
14     for i=1:N
15         if rem(i,2)==0
16             DH_params(2*i-1:2*i,:)=[180+(theta_j(2*i-1)*180/pi) 90 0
0; 180+(theta_j(2*i)*180/pi) 0 a 0];
17         else
18             DH_params(2*i-1:2*i,:)=[(theta_j(2*i-1)*180/pi) 90 0 0;
(theta_j(2*i)*180/pi) 0 a 0];
19         end
20     end
21
22 % Calculate transformation matrix n_A_n+1 (A_wrt_prev) and store
in cell
23 % array
24 A = cell(2*N,1);
25 for n=1:2*N
26     theta_i=DH_params(n,1);
27     alpha_i=DH_params(n,2);
28     r_i=DH_params(n,3);
29     d_i=DH_params(n,4);
30     A{n}=[cosd(theta_i) -sind(theta_i)*cosd(alpha_i) sind(
theta_i)*sind(alpha_i) r_i*cosd(theta_i);...
31         sind(theta_i) cosd(theta_i)*cosd(alpha_i) -cosd(theta_i)
*sind(alpha_i) r_i*sind(theta_i);...
32         0 sind(alpha_i) cosd(alpha_i)
d_i;...

```

```

33         0           0           0
           1];
34 end
35
36 % Create a list of transformation matrices wrt base for all
    joints
37 A_till_prev_wrt_base=cell(2*N,1);
38 A_till_prev_wrt_base{1}=eye(4);
39 for n=1:2*N-1
40     A_till_prev_wrt_base{n+1}=A_till_prev_wrt_base{n}*A{n};
41 end
42 A_2N=A_till_prev_wrt_base{2*N}*A{2*N};
43 % Create Jacobian
44 J=zeros(6,2*N);
45 for n=1:2*N
46     J_v=cross((A_till_prev_wrt_base{n}(1:3,1:3)*[0;0;1]),(A_2N
        (1:3,4)-A_till_prev_wrt_base{n}(1:3,4)));
47     J_w=A_till_prev_wrt_base{n}(1:3,1:3)*[0;0;1];
48     J(:,n)=[J_v;J_w];
49 end
50
51 x_e_j = J*(theta_j-theta_prev) + x_e_prev;
52 % Get error
53 x_e_j_error = x_e - x_e_j;
54 if norm(x_e_j_error) <= (error_tol*N)
55     theta = theta_j;
56     break
57 end
58 % Use error to estimate delta_theta
59 J_inv=pinv(J);
60 delta_theta_j= J_inv*x_e_j_error;
61 % Update theta_j
62 theta_j = theta_j + delta_theta_j;
63
64 if j==10000
65     theta=ones(2*N,1)*NaN;
66 end
67 end
68 if all(isnan(theta(:)))
69     disp('Could not find solution!')
70 end
71 end

```

E.4 Dynamic Model

E.4.1 Calculating Contact Forces


```

1 function [F_ec,T_ec] = fcn(tau_m,theta,a,d,r,r_drum, l_link ,eta ,
    mu)
2 N=9;
3 F_ec=zeros(3,N);
4 T_ec=zeros(3,N);
5 F_cd=cell(N,3*N);
6 F_cp=cell(N,3*N);
7 for i=1:N
8     for j=1:3*N
9         F_cd{i,j}=[0;0;0];
10        F_cp{i,j}=[0;0;0];
11    end
12 end
13 r_d=cell(N,3*N);
14 r_p=cell(N,3*N);
15 T_p=cell(N,3*N);
16 T_d=cell(N,3*N);
17 P_link_cm=[-(a/2+d);0;0];
18 % Initialise prev values
19 Td_prev=tau_m*r_drum/eta;
20 % Calculate the Transformation matrices
21 DH_params=zeros(2*N,4);
22 for i=1:N
23     if rem(i,2)==0
24         DH_params(2*i-1:2*i,:)=[180+(theta(2*i-1)*180/pi) 90 0
0; 180+(theta(2*i)*180/pi) 0 a 0];
25     else
26         DH_params(2*i-1:2*i,:)=[(theta(2*i-1)*180/pi) 90 0 0; (
theta(2*i)*180/pi) 0 a 0];
27     end
28 end
29
30 % Calculate transformation matrix n_A_n+1 (A_wrt_prev) and store
    in cell
31 % array
32 A = cell(2*N,1);
33 for n=1:2*N
34     theta_i=DH_params(n,1);
35     alpha_i=DH_params(n,2);
36     r_i=DH_params(n,3);
37     d_i=DH_params(n,4);
38     A{n}=[cosd(theta_i) -sind(theta_i)*cosd(alpha_i) sind(
theta_i)*sind(alpha_i) r_i*cosd(theta_i);...
39         sind(theta_i) cosd(theta_i)*cosd(alpha_i) -cosd(theta_i)
*sind(alpha_i) r_i*sind(theta_i);...
40         0 sind(alpha_i) cosd(alpha_i)
d_i;...
41         0 0 0
1];

```

```

42 end
43
44 % Create a list of transformation matrices wrt base for all
    joints
45 A_wrt_base=cell(2*N,1);
46 A_wrt_base{1}=A{1};
47 for n=2:2*N
48     A_wrt_base{n}=A_wrt_base{n-1}*A{n};
49 end
50
51 for i=1:N
52     for j=1:3*N
53         M=floor((j-1)/3)+1;
54         % Calculate delta and r
55         if rem(i,2)==0
56             delta_p=((120/(N+1))*(M)+120*(j-1)+87);
57             delta_d=((120/(N+1))*(M)+120*(j-1)-3);
58             r_p_j=[d-l_link; r*cosd(delta_p); r*sind(delta_p);
1];
59             r_d_j=[-d; r*cosd(delta_d); r*sind(delta_d); 1];
60         else
61             delta_p=((120/(N+1))*(M)+120*(j-1)-3);
62             delta_d=((120/(N+1))*(M)+120*(j-1)+87);
63             if i==1
64                 L_d_prev=[-d; r*cosd(delta_d); r*sind(delta_d);
1];
65                 e_d_prev=L_d_prev(1:3)/norm(L_d_prev(1:3));
66                 e_d_0=e_d_prev;
67                 r_d_prev=[-d; r*cosd(delta_d); r*sind(delta_d);
1];
68             end
69             r_p_j=[d-l_link; r*sind(delta_p); -r*cosd(delta_p);
1];
70             r_d_j=[-d; r*sind(delta_d); -r*cosd(delta_d); 1];
71         end
72         r_p{i,j}=r_p_j(1:3);
73         r_d{i,j}=r_d_j(1:3);
74         % Calculate L
75         if i==1
76             L_p=(A{2*i}*r_p_j)-(r_d_prev);
77         else
78             L_p=(A{2*i}*r_p_j)-(A{2*i-2}*r_d_prev);
79         end
80         L_d=A{2*i}*(r_d_j-r_p_j);
81         % Calculate e
82         e_p=L_p(1:3)/norm(L_p(1:3));
83         e_d=L_d(1:3)/norm(L_d(1:3));
84         % Calculate phi
85         phi_p=acos(dot(e_p,e_d));

```

```

86     phi_d=acos(dot(e_d_prev,e_p));
87     % Get Tp
88     T_p_j=Td_prev/(exp(mu*phi_d));
89     T_p{i,j}=T_p_j*e_p;
90     % Get Td
91     T_d_j=T_p_j/(exp(mu*phi_p));
92     T_d{i,j}=T_d_j*e_d;
93     % Update prev values
94     e_d_prev=e_d;
95     r_d_prev=r_d_j;
96     Td_prev=T_d_j;
97 end
98 end
99
100 % Calculate F_cp and F_cd
101 for n=1:N
102     for k=1:3*N
103         M=floor((k-1)/3)+1;
104         if (3*n+1 <= k) && (k<=3*N) && n<=M
105             F_cp{n,k}=T_d{n,k}-T_p{n,k};
106             if n==1
107                 F_cd{n,k}=T_p{n,k}-((tau_m*r_drum/eta)*e_d_0);
108             else
109                 F_cd{n,k}=T_p{n,k}-T_d{n-1,k};
110             end
111         end
112         if (3*n-2 <= k) && (k <=3*n) && n==M
113             F_cp{n,k}=-T_p{n,k};
114         end
115     end
116 end
117
118 % Calculate F_ec
119 for n=1:N
120     F_ec_n=[0;0;0];
121     A_2n=inv(A{2*n});
122     R_2n=A_2n(1:3,1:3);
123     for k=3*n+1:3*N
124         F_ec_n=F_ec_n+((R_2n)*F_cd{n,k});
125     end
126     for k=3*n-2:3*N
127         F_ec_n=F_ec_n+((R_2n)*F_cp{n,k});
128     end
129     F_ec(:,n)=F_ec_n;
130 end
131
132 % Calculate T_ec
133 for n=1:N
134     T_ec_n=[0;0;0];

```

```

135     A_2n=inv(A{2*n});
136     R_2n=A_2n(1:3,1:3);
137     for k=3*n+1:3*N
138         T_ec_n=T_ec_n+cross((r_d{n,k}-P_link_cm),((R_2n)*F_cd{n,
k}));
139     end
140     for k=3*n-2:3*N
141         T_ec_n=T_ec_n+cross((r_p{n,k}-P_link_cm),((R_2n)*F_cp{n,
k}));
142     end
143     T_ec(:,n)=T_ec_n;
144 end

```

E.4.2 Newton-Euler Recursive Algorithm

```

1 %% Neton-Euler Recursive Algorithm for Inverse Dynamics
2 function u = fcn(theta, d_theta, dd_theta, m, a, d, r, g, F_ec, T_ec)
3 %% Parameters
4 N=9;
5 P_joint_to_joint=[a+2*d;0;0];
6 P_link_cm=[-(a/2+d);0;0];
7 z_0=[0;0;1];
8 x_0=[1;0;0];
9 % Create the D-H parameter table
10 DH_params=zeros(2*N,4);
11 for i=1:N
12     if rem(i,2)==0
13         DH_params(2*i-1:2*i,:)=[180+(theta(2*i-1)*180/pi) 90 0
0;...
14                                     180+(theta(2*i)*180/pi) 0 a 0];
15     else
16         DH_params(2*i-1:2*i,:)=[theta(2*i-1)*180/pi 90 0 0;...
17                                     theta(2*i)*180/pi 0 a 0];
18     end
19 end
20 % Calculate transformation matrix n_A_n+1 (A_wrt_prev) and store
in cell
21 % array
22 R = cell(2*N,1);
23 for n=1:2*N
24     theta_i=DH_params(n,1);
25     alpha_i=DH_params(n,2);
26     R{n}=[cosd(theta_i) -sind(theta_i)*cosd(alpha_i) sind(
theta_i)*sind(alpha_i);...
27         sind(theta_i) cosd(theta_i)*cosd(alpha_i) -cosd(theta_i)
*sind(alpha_i);...
28         0 sind(alpha_i) cosd(alpha_i)
];

```

```

29 end
30 % Calculate Inertia tensor
31 I = cell(N,1);
32 for i=1:N
33     I{i}=[1/12*(m*(a)^2)+1/4*(m*r^2) 0 0;
34           0 1/12*(m*(a)^2)+1/4*(m*r^2) 0;
35           0 0 1/4*(m*r^2)];
36 end
37 %% Forward recursion
38 % Initialising the matrices
39 omega=zeros(3,N);
40 d_omega=zeros(3,N);
41 a_joint=zeros(3,N);
42 a_c=zeros(3,N);
43 % Initialising prev values
44 omega_prev=zeros(3,1);
45 d_omega_prev=zeros(3,1);
46 a_joint_prev=zeros(3,1);
47 for n=1:N
48     omega(:,n)=(R{2*n-1}*R{2*n})'*(omega_prev + z_0*d_theta(2*n
49     -1)) + ...
50     (R{2*n})' * z_0*d_theta(2*n);
51     d_omega(:,n)=(R{2*n-1}*R{2*n})'*(d_omega_prev + z_0*dd_theta
52     (2*n-1)+...
53     cross(omega(:,n),z_0*
54     d_theta(2*n-1))) + ...
55     (R{2*n-1}*R{2*n})'*(cross((z_0*d_theta(2*n-1) +
56     omega_prev),R{2*n-1}*z_0*d_theta(2*n))) + ...
57     (R{2*n})'*(z_0*dd_theta(2*n)));
58     if n==1
59         a_joint(:,n)=[0;0;g];
60     else
61         a_joint(:,n)=(R{2*n-3}*R{2*n-2})'*a_joint_prev+ ...
62         cross(d_omega_prev,P_joint_to_joint)+...
63         cross(omega_prev,cross(omega_prev,
64         P_joint_to_joint));
65     end
66     a_c(:,n)=(R{2*n-1}*R{2*n})'*a_joint(:,n)+...
67     cross(d_omega(:,n),(P_joint_to_joint+P_link_cm))
68     +...
69     cross(omega(:,n),cross(omega(:,n),(P_joint_to_joint
70     +P_link_cm)));
71     % Update prev values for next iteration
72     omega_prev=omega(:,n);
73     d_omega_prev=d_omega(:,n);
74     a_joint_prev=a_joint(:,n);
75 end
76 %% Backward recursion

```

```

71 % Initialising the matrices
72 F=zeros(3,N);
73 tau=zeros(3,N);
74 % Initialise next values
75 F_next=zeros(3,1);
76 tau_next=zeros(3,1);
77 for n=N:-1:1
78     F(:,n) = F_next + m*a_c(:,n)-F_ec(:,n);
79     tau(:,n) = tau_next - cross(F(:,n),(P_joint_to_joint+
80         P_link_cm)) + ...
81         cross(F_next,P_link_cm) + ...
82         l{n}*d_omega(:,n)+ cross(omega(:,n),l{n}*omega(:,n
83         ))-T_ec(:,n);
84     % Update next values for next iteration
85     F_next=F(:,n);
86     tau_next=tau(:,n);
87 end
88 %% Computing the output
89 u=vecnorm(tau)';

```

References

- [1] K. Y. Pettersen, “Snake robots,” *Annual Reviews in Control*, vol. 44, pp. 19–44, 2017.
- [2] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, “A review on modelling, implementation, and control of snake robots,” *Robotics and Autonomous systems*, vol. 60, no. 1, pp. 29–40, 2012.
- [3] Matthew Knight, CNN, “Snake robot offers new twist on nuclear cleanup,” <https://edition.cnn.com/2011/11/18/tech/innovation/snake-arm-robot-nuclear/index.html>, 2011, [Online; accessed 11-December-2019].
- [4] M. R. Shortis, S. Robson, T. W. Jones, W. K. Goad, and C. B. Lunsford, “Photogrammetric tracking of aerodynamic surfaces and aerospace models at nasa langley research center,” *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3, pp. 27–34, 2016.
- [5] OC Robotics, “Series ii, x125 snake-arm robot,” <http://www.ocrobotics.com/technology-/series-ii-x125-system/>, 2015, [Online; accessed 14-October-2019].
- [6] M. Saito, M. Fukaya, and T. Iwasaki, “Modeling, analysis, and synthesis of serpentine locomotion with a multilink robotic snake,” *IEEE control systems magazine*, vol. 22, no. 1, pp. 64–81, 2002.
- [7] S. Hirose and M. Mori, “Biologically inspired snake-like robots,” in *2004 IEEE International Conference on Robotics and Biomimetics*, Aug 2004, pp. 1–7.
- [8] X. Zhou, F. Wang, L. Dong, and Z. Dai, “Analysis of under-actuated snake arm robot,” 2018.

- [9] P. K. Singh and C. M. Krishna, "Continuum arm robotic manipulator: A review," *Universal Journal of Mechanical Engineering*, vol. 2, no. 6, pp. 193–198, 2014.
- [10] R. Buckingham and A. Graham, "Nuclear snake-arm robots," vol. 39, no. 1, pp. 6–11, 2012. [Online]. Available: <https://dx.doi.org/10.1108/01439911211192448>
- [11] R. Buckingham, "Snake arm robots," vol. 29, no. 3, pp. 242–245, 2002. [Online]. Available: <https://dx.doi.org/10.1108/01439910210425531>
- [12] NASA.
- [13] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 43–55, 2006. [Online]. Available: <https://dx.doi.org/10.1109/TRO.2005.861458>
- [14] Z. Li and R. Du, "Design and implementation of a biomimetic wire-driven underactuated serpentine manipulator," *Transaction on Control and Mechanical Systems*, vol. 1, pp. 250–258, 2012.
- [15] Z. Li and R. Du, "Design and analysis of a bio-inspired wire-driven multi-section flexible robot," *International Journal of Advanced Robotic Systems*, vol. 10, no. 4, p. 209, 2013. [Online]. Available: <https://dx.doi.org/10.5772/56025>
- [16] Z. Li, R. Du, M. C. Lei, and S. M. Yuan, "Design and analysis of a biomimetic wire-driven robot arm," in *ASME 2011 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, Conference Proceedings, pp. 191–198.
- [17] Z. Li, R. Du, H. Yu, and H. Ren, "Statics modeling of an underactuated wire-driven flexible robotic arm." IEEE, Conference Proceedings. [Online]. Available: <https://dx.doi.org/10.1109/BIOROB.2014.6913797>
- [18] L. Tang, J. Huang, L.-M. Zhu, X. Zhu, and G. Gu, "Path tracking of a cable-driven snake robot with a two-level motion planning method," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 935–946, 2019. [Online]. Available: <https://dx.doi.org/10.1109/TMECH.2019.2909758>

- [19] L. Tang, J. Wang, Y. Zheng, G. Gu, L. Zhu, and X. Zhu, "Design of a cable-driven hyper-redundant robot with experimental validation," *International Journal of Advanced Robotic Systems*, vol. 14, no. 5, p. 172988141773445, 2017. [Online]. Available: <https://dx.doi.org/10.1177/1729881417734458>
- [20] J. Tang, Y. Zhang, F. Huang, J. Li, Z. Chen, W. Song, S. Zhu, and J. Gu, "Design and kinematic control of the cable-driven hyper-redundant manipulator for potential underwater applications," *Applied Sciences*, vol. 9, no. 6, p. 1142, 2019.
- [21] W. Xu, T. Liu, and Y. Li, "Kinematics, dynamics, and control of a cable-driven hyper-redundant manipulator," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 4, pp. 1693–1704, 2018. [Online]. Available: <https://dx.doi.org/10.1109/TMECH.2018.2842141>
- [22] G. S. Chirikjian and J. W. Burdick, "A modal approach to hyper-redundant manipulator kinematics," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 343–354, 1994.
- [23] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, "A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5033–5039.
- [24] OCRobotics, "Oc robotics - mobile snake-arm robot," <https://www.youtube.com/watch?v=cV7aT8nvUL4>, 2015, [Online; accessed 14-October-2019].
- [25] OCRobotics, "Snake-arm robots using maxon motors," <https://www.youtube.com/watch?v=MaG0p2PWuRI>, 2015, [Online; accessed 14-October-2019].
- [26] maxon UK&I, "Oc robotics - snake arm 101," https://www.youtube.com/watch?v=Ij8VX9YUT_Y, 2016, [Online; accessed 14-October-2019].
- [27] OCRobotics, "Oc robotics - series ii, x125 snake-arm robot for ucl," <https://www.youtube.com/watch?v=I4M6fYeyAIY>, 2015, [Online; accessed 14-October-2019].
- [28] maxon motor, "Snake-arm robots are highly flexible robots ideal for working in confined and hazardous spaces."

- <http://www.eurekamagazine.co.uk/design-engineering-products/snake-arm-robots/146523/>, 2015, [Online; accessed 03-November-2019].
- [29] N. Derbel, J. Ghommam, and Q. Zhu, *New Developments and Advances in Robot Control*. Springer, 2019, vol. 175.
 - [30] W. Xu, Z. Mu, T. Liu, and B. Liang, "A modified modal method for solving the mission-oriented inverse kinematics of hyper-redundant space manipulators for on-orbit servicing," *Acta Astronautica*, vol. 139, pp. 54–66, 2017. [Online]. Available: <https://dx.doi.org/10.1016/j.actaastro.2017.06.015>
 - [31] Prof. Alessandro De Luca, *Dynamic model of robots:Newton-Euler approach*, Sapienza Universita di Roma. [Online]. Available: http://www.diag.uniroma1.it/deluca/rob2_en/06_NewtonEulerDynamics.pdf
 - [32] *maxon DC motor and maxon EC motor*, maxon motor, 2014. [Online]. Available: https://www.maxongroup.com/medias/sys_master/root/8815460712478/DC-EC-Key-Information-14-EN-42-50.pdf
 - [33] *maxon Motor data and Operating Ranges*, maxon motor, 2010. [Online]. Available: <http://storkdrives.com/wp-content/uploads/2013/10/2-maxon-Motor-Data2.pdf>
 - [34] Northwestern Robotics, "Modern robotics, chapter 8.5: Forward dynamics of open chains," <https://www.youtube.com/watch?v=L8zpJOxDbh4>, 2017, [Online; accessed 14-October-2019].