# Trajectory Optimisation Inspired Controller Design of a Low-Cost Bipedal Robot

by

Devlon Erasmus




*Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University.*

Supervisor: Dr C. Fisher

March 2023

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

# Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   *Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.*

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   *I agree that plagiarism is a punishable offence because it constitutes theft.*

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   *I also understand that direct translations are plagiarism.*

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   *Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism*

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   *I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.*

| Studentenommer / *Student number* | Handtekening / *Signature* |
|---|---|
| Devlon Erasmus | 1 December 2022 |
| Voorletters en van / *Initials and surname* | Datum / *Date* |

# Abstract

For robotic systems to become a greater part of everyday life, low-cost platforms are thus needed with the same capabilities as their expensive counterparts. Legged robotics has become the focal point of recent years, partially aided by the developments in trajectory optimisation. However, a major problem in legged robots is that the precise determinants of locomotion are still under debate, making it difficult to design controllers for these systems.

The work done in this research investigated if trajectory optimisation can be used to inspire controller design for a low-cost bipedal robot. A low-cost robotic platform was designed and built to test and validate the trajectories. The robot consisted of servo motors at the hips and pneumatic cylinders as legs and required a support rig to constrain it to the sagittal plane. ROS was used to communicate between the systems running on the robot and the main control loop running on a PC.

Trajectory optimisation was used to generate acceleration, steady-state and deceleration trajectories for the robot. The trajectories were based on the mathematical model of the robot, with the contacts modelled using through-contact methods. This method allowed the optimiser to choose when the contacts occurred, which allowed for an optimal gait to emerge from the trajectories.

Once the robot was finalised and built, the trajectories generated by the trajectory optimisation were used to inspire the design of controllers for the system by analysing common trends within the trajectories. It was found that during the steady-state phase of the robot, the Raibert controller emerged, resulting in the controller taking a state-machine form similar to the one proposed by Raibert.

The controllers were first tested in a ROS Gazebo simulation as a validation step to investigate if the controllers could successfully control the system before implementing it on the physical robot. The controllers were implemented on a fixed and free body model and were tested on a rigid surface as well as a rough surface, in order to test their robustness. The results of the testing indicated that trajectory optimisation could lead to controller design for a low-cost biped robot.

# Uittreksel

Om ten einde robotstelsels 'n groter deel van die alledaagse lewe te maak, is laekosteplatforms nodig met dieselfde vermoëns as hul duur eweknieë. Been-robotika het die fokuspunt van onlangse jare geword, aangehelp deur die ontwikkelings in trajek-optimering. 'n Groot probleem in robote met bene is egter dat die presiese determinante van voortbeweging steeds onder debat is, wat dit moeilik maak om beheerders vir hierdie stelsels te ontwerp.

Die werk wat in hierdie navorsing gedoen was, ondersoek of trajek-optimering 'n beheerstelser ontwerp vir 'n laekoste tweevoetige robot kan inspireer. Vir daai rede was 'n laekoste robotplatform ontwerp en gebou om die trajekte op te toets en te valideer. Die robot bestaan uit 'n servomotors by die heupe en pneumatiese silinders as bene en het 'n ondersteuningstuig nodig om dit tot die sagittale vlak te beperk. ROS was gebruik om te kommunikeer tussen die stelsels wat op die robot en die hoofbeheerlus wat op 'n rekenaar loop.

Trajek-optimering was gebruik om versnelling, bestendige-toestand en vertraging trajekte vir die robot te genereer. Die bane was gebaseer op die wiskundige model van die robot, met die kontakte wat deur "through-contact" metodes gemodelleer was. Hierdie metode het die optimaliseerder toegelaat om te kies wanneer die kontakte plaasvind, wait toegelaat het dat 'n optimale loop gang uit die trajekte na vore kom.

Sodra die robot gefinaliseer en gebou was, was die trajekte wat deur die optimalisering gegenereer was, gebruik om beheerders vir die stelsel te ontwerp deur algemene neigings wat in die trajek na vore kom te ontleed. Daar was gevind dat tydens die bestendige fase van die robot die Raibert-beheerder na vore gekom het. Die beheerder het dus 'n staatsmasjien-vorm aangeneem soortgelyk aan die een wat deur Raibert voorgestel is.

Die beheerders was eers in 'n ROS Gazebo-simulasie getoets as 'n valideringsstap om te ondersoek of die beheerders 'n stelsel suksesvol kan beheer voordat dit op die fisiese robot geïmplementeer word. Die beheerders was op 'n vaste en vrye liggaamsmodel geïmplementeer en was op 'n rigiede oppervlak sowel as 'n growwe oppervlak getoets om die robuustheid van die beheerders te ondersoek. Die resultate van die toetsing het aangedui dat baanoptimering kan lei tot beheerderontwerp vir 'n laekoste tweevoetige robot.

# Acknowledgements

I would like to express my thanks to my supervisor, Dr Callen Fisher. Thank you for all the support and advice you have given me throughout this project, without your guidance I would most likely still be staring at the trajectory optimisation screen.

I would also like to thank the NRF (National Research Fund) for sponsoring this research under grant number 129830, allowing the robot to physically become a reality.

To my lovely parents, I want to say thank you for making my studying a possibility. You have supported me through this entire 6 year endeavour, not only financially but mentally as well. I appreciate all the support and guidance you have given me throughout my life.

To my twin brother, Sheldon, thank you for going with me on this journey and for the advice you have given me. I wish you the best of luck with future work, and I am excited to see what the future has in-store for you. Lastly, to my partner, Cara-Lee, thank you. Without all the coffee breaks, I would not have been able to function. I appreciate you lending me your ear to rant about stuff not working like it is supposed to. Your support and companionship mean the world to me. I am sorry for all the nerdy engineering talk, though.

# Contents

**Declaration** i

**Abstract** ii

**Uittreksel** iii

**Acknowledgements** iv

**Contents** v

**List of Figures** viii

**List of Tables** xi

**Nomenclature** xii

**1 Introduction** 1

1.1 Motivation and Background to the Study . . . . . 2
1.2 Research Aims and Objectives . . . . . 3
1.3 Scope and Limitations . . . . . 4
1.4 Plan of Development . . . . . 4

**2 Literature Review** 6

2.1 Legged Robotics . . . . . 6
2.2 Legged Actuation . . . . . 8
2.2.1 Torque Actuation . . . . . 8
2.2.2 Linear Actuation . . . . . 11
2.2.3 Elastic Actuation . . . . . 12
2.3 Trajectory Optimisation . . . . . 14
2.3.1 Direct Methods . . . . . 14
2.3.2 Through-Contact Methods . . . . . 15
2.4 Controlling Hopping Robots . . . . . 17
2.5 ROS and Gazebo . . . . . 18

**3 Methodology** **22**
3.1 Modelling and Simulating the System . . . 22
3.1.1 Equations of Motion . . . 23
3.1.2 Trajectory Optimisation . . . 24

**4 Robotic Platform** **37**
4.1 Platform Design . . . 37
4.1.1 Torque Actuator . . . 37
4.1.2 Linear Actuator . . . 38
4.1.3 Robotic Platform . . . 39
4.2 Electrical Design . . . 43
4.2.1 Communication . . . 43
4.2.2 Electronics . . . 45

**5 Controller Design** **48**
5.1 Trajectory Analysis of Fixed Body Model . . . 48
5.1.1 Optimiser Setup . . . 48
5.2 Trajectory Analysis of Free Body . . . 55
5.2.1 Optimiser Setup . . . 55
5.2.2 Optimisation Results . . . 55
5.3 Pneumatic Control . . . 58
5.4 Servo Control . . . 60
5.4.1 Fixed Body . . . 60
5.4.2 Free Body . . . 61

**6 Gazebo Simulation** **62**
6.1 Simulation Setup . . . 62
6.2 Gazebo Results . . . 64
6.2.1 Fixed Body . . . 64
6.2.2 Free Body . . . 66

**7 Testing and Results** **69**
7.1 Fixed Body . . . 69
7.1.1 Rigid Surface . . . 69
7.1.2 Rough Surface . . . 71
7.2 Free Body . . . 74
7.2.1 Rigid Surface . . . 74
7.2.2 Rough Surface . . . 75

**8 Conclusion** **79**
8.1 Future Work . . . 81

**A Equations of Motion** **82**

**B Electrical Schematic** **85**

**Bibliography** **87**

# List of Figures

1.1 Two examples of well-known walking platforms that have been produced by past literature. . . . 1
1.2 A graphical representation of an overlook of the thesis. . . . 5

2.1 A figure showing some of the early robots created by Marc Raibert. . . . 7
2.2 A figure showing the Festo Kangaroo. . . . 7
2.3 A figure showing examples of direct-drive motored robots. . . . 9
2.4 A figure showing examples of geared motored robots. . . . 10
2.5 Figure showing the principle on which hydraulic and pneumatic systems work. 11
2.6 Figure showing SEA and PEA. . . . 13
2.7 An Image showing the impact of foot placement of a hopping system. . . . 19
2.8 A figure showing a broad overview of a ROS master and nodes, as well as a typical interaction between a ROS subscriber and publisher. . . . 19
2.9 A Figure showing the friction cone used in ROS Gazebo. . . . 20

3.1 Model of the bipedal pneumatic-electrical robot. . . . 23
3.2 Figure showing a graphical representation of a stitched long-time-horizon trajectory. . . . 26
3.3 A figure showing a graphical representation of how the trajectory was broken up into node points and collocation points. . . . 27
3.4 Figure showing a complementarity constraint for foot height and GRF. . . . 30
3.5 The graph shows a graphical representation of the nodes that are grouped for solenoid switching. . . . 34

4.1 Figure comparing the designed model to the physical robot. . . . 40
4.2 Figure showing an orthographic view of the robot. . . . 41
4.3 Inventor model compared to fully built platform. . . . 42
4.4 A figure showing the pivot point of the free body robot. . . . 42
4.5 Figure showing the communication between the processors, sensors, and actuators. . . . 43
4.6 The figure shows the data passed between the ROS nodes running on the Raspberry Pi and the PC. . . . 45

4.7 Figure showing the PCB used for the electrical connections. . . . . 45
4.8 Figure showing the Zener protection circuit. . . . . 47

5.1 Figure showing the results of the jump test and servo velocity test. . . . . 50
5.2 Figure showing the results obtained from the optimiser for the fixed body while enforcing three different average velocities. . . . . 52
5.3 Figure showing the two control heights for the state machine. . . . . 53
5.4 A figure showing the fixed body robot at the start of the acceleration phase. . 54
5.5 A figure showing the 0.15 m/s trajectory stitched with an acceleration and deceleration phase. . . . . 55
5.6 Figure showing the results obtained from the optimiser for the free body while enforcing three different average velocities. . . . . 56
5.7 A figure showing the free body robot at the start of the acceleration phase. . . 58
5.8 A figure showing the 0.20 m/s trajectory stitched with an acceleration and deceleration phase. . . . . 58
5.9 Figure showing the structure of the state-machine controller. . . . . 59

6.1 A Figure showing the robot within the Gazebo environment. . . . . 63
6.2 A diagram showing the interaction between the Python script running the control loop and the ROS topics and services required to control the robot. . . 64
6.3 A figure comparing the results obtained from the trajectory optimisation to the result from the Gazebo simulation for the fixed body. . . . . 65
6.4 A figure comparing the results obtained from the trajectory optimisation to the result from the Gazebo simulation for the free body. . . . . 67
6.5 A figure showing the body angle obtained from the free body testing compared to the simulation. . . . . 68

7.1 Figure showing data obtained from testing of the fixed body robot on a rigid surface. . . . . 70
7.2 Figure showing the wooden blocks placed in the path of the fixed body robot. 71
7.3 Figure showing data obtained from testing of the fixed body robot on a rigid surface compared to the rough surface. . . . . 72
7.4 Graph showing the result of the limit cycle of the fixed body robot along the limit cycle obtained from the model used in the trajectory optimisation. . . . . 73
7.5 Figure showing the data obtained from testing of the free body robot on a rigid surface. . . . . 75
7.6 Figure showing the data obtained from testing of the fixed body robot on a rigid and rough surface. . . . . 76
7.7 Figure showing the body angle of the free body robot for both the rigid and rough surfaces. . . . . 77

7.8 Graph showing the result of the limit cycle of the free body robot along the limit cycle obtained from the model used in the trajectory optimisation. . . . . 78

B.1 The electrical schematic of the system. . . . . 86

# List of Tables

4.1 Comparison of Servo Motors. . . . . 38
4.2 Comparison of Pneumatic Cylinders. . . . . 39
4.3 Supply voltages required for the robotic platform. . . . . 46

5.1 Time Bounds and Travel Distance for Fixed Body Robot. . . . . 49
5.2 Robot Parameters. . . . . 50
5.3 Analysis of Touchdown and Lift-Off Angles for Fixed Body. . . . . 51
5.4 Neutral Point Analysis of the Fixed Body Robot. . . . . 54
5.5 Analysis of Touchdown and Lift-Off Angles for Free Body. . . . . 56
5.6 Neutral Point Analysis for the Free Body Robot. . . . . 57

# Nomenclature

### Abbreviations

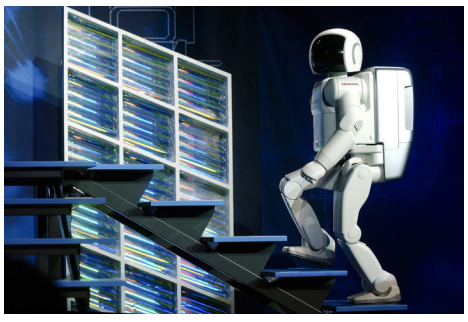
| | |
|---|---|
| *AC* | Alternating Current |
| *DC* | Direct Current |
| *DoF* | Degrees of Freedom |
| *EoM* | Equations of Motion |
| *GRF* | Ground Reaction Forces |
| *MCU* | Micro-Controller Unit |
| *PEA* | Parallel Elastic Actuator |
| *PC* | Personal Computer |
| *PCB* | Printed Circuit Board |
| *QDD* | Quasi Direct-Drive |
| *ROS* | Robot Operating System |
| *SEA* | series Elastic Actuator |
| *UART* | Universal Asynchronous Receive-Transmit |
| *URDF* | Universal Robot Description Format |
| *USB* | Universal Serial Bus |

# Chapter 1

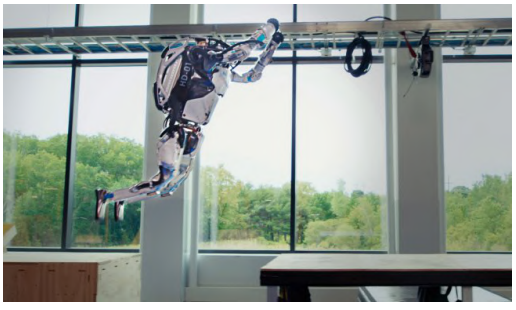
# Introduction

The design of the modern world has been shaped by the urge of man to make daily tasks more automated, reducing the man power and effort required. With this increase in automation comes the desire to create robotic platforms which will be capable of robust and complex agile movements over varying types of terrain, to aid man in daily tasks or to assist in dangerous situations (such as firefighters rushing into a burning building). Legged locomotion allows for the optimal method of movement over land, as it allows for navigation over rugged terrain. Legged robots possess the ability to choose where to place their feet during intermittent contacts allowing for rapid agile manoeuvres, which wheeled robotics simply are not capable of achieving [1][2][3].

However, these systems tend to be expensive, requiring powerful computers to generate the motions of the platform while using expensive actuators to move the limbs. Past literature has produced several walking platforms, these platforms were developed by large organisations that have large industry investments to dedicate to these systems. Organisations such as Honda with Asimo [4], Boston Dynamics with Atlas [5] and MIT with the MIT Cheetah [1], have all developed state-of-the-art platforms, with Atlas even capable of Parkour. Figure 1.1 shows two of the well-known platforms, the slower-moving, Asimo, and the Parkouring robot, Atlas.



**(a)** Asimo [4].



**(b)** Atlas [5].

**Figure 1.1:** Two examples of well-known walking platforms that have been produced by past literature.

Even with these advancements, a way is needed to produce and control low-cost robotic platforms capable of explosive agile manoeuvres to bridge the gap and make robotics more accessible. This research will aim to solve this problem by producing a low-cost platform, with explosive movement, without the use of expensive computers to control the motion of the robot while using low-cost actuation schemes, such as a pneumatic cylinder attached to a torque actuator at the hip, to provide a robust, agile robot [6]. Trajectory optimisation will be used to generate the motions of the robot, which will in turn be used to inspire controllers capable of controlling the robotic platform. Trajectory optimisation has in the literature been proven as a way to produce robust and agile trajectories for a robotic platform [2][7][8] and it can be implemented on any modern-day computer.

## 1.1 Motivation and Background to the Study

This research project was motivated by the idea of introducing legged robotics into the daily lives of men and women around the globe, but in order to achieve this, reliable and low-cost robotics systems capable of robust, agile explosive movements have to be developed and controlled. In the past 40 years, legged robotics have come a long way from the simple one-legged hopper that Marc Raibert built in 1984 [9], to today where a quadruped robot, Spot [10], is commercially sold by Boston Dynamics for $74,500. Before robotics can be a greater part of everyday life, lower-cost systems need to be developed that reach the same functionality as their high-cost counterparts.

Pneumatic actuation is considered simpler and cheaper than brushless direct current (DC) motors used in most robotic systems' leg designs, while providing sufficient power. Pneumatic actuation has been successfully used in legged robotics in the past [6][11]. A pneumatic cylinder paired with a torque actuator at the hip is capable of rapid acceleration and deceleration. This is a low-cost way to deliver ample explosive power to a robotic platform [9].

Any state-of-the-art robotic platform is vastly inferior in legged locomotion compared to those performed in nature. Legged locomotion in nature comes effortlessly while allowing for high agile movements that not even the current state-of-the-art platforms like the Atlas [5] (which is capable of running, rolling, crouching and even parkour) and the MIT Cheetah [1] can compare to. With all these advances in robotics, walking is still considered a complex and unsolved task, which is why researchers apply assumptions such as massless legs [12][13], infinite friction [12][14] as well as simplified and reduced order models [15]. This makes the task easier to solve, while also reducing the cost and complexity of the computers required to generate motions for a robotic system.

In this research, trajectory optimisation was used to generate a path for the robot and the data provided by the optimisation was then used to inspire a controller framework for

the robotic system. From past trajectory optimisation literature [2][7][8], it was clear that trajectory-inspired control has led to robust and agile movements. A technique used in the optimisation, through-contact optimisation methods [16][17] (detailed more in later chapters), further increases the robustness of the optimiser by allowing it to choose its own contact order, allowing for an optimal gait to emerge from the robot's movement. The robustness of the optimisation techniques used led to a robust controller design for the robotic platform.

## 1.2 Research Aims and Objectives

The primary aim of this research project was to design controllers for a low-cost bipedal robot, inspired by trajectory optimisation, which could enable the robot to traverse over terrain to reach its end goal. The aim was then split into the following objectives which should be reached to complete the project.

1. Design and build a low-cost bipedal robotic platform. Two torque actuators are attached at the hip, with the legs consisting of pneumatic actuators. The robot was constrained to the sagittal plane with the use of a support rig. The support rig was equipped with the necessary sensors to control the robot as well as deliver the required power to the actuators.

2. Implement trajectory optimisation to generate trajectories for the bipedal robot for a fixed body as well as a free body model. For the trajectory optimisation, the mathematical model of the robot was developed before the trajectories were generated. The mathematical model of the robot included the desired actuators and ensured that the characteristics thereof were modelled accordingly.

3. Controllers had to be designed which enabled the robot to complete the long-time-horizon movement (start and end at rest after travelling a fixed distance). These controllers were derived from the trajectories generated by the optimiser.

4. Verify the controllers in simulation based on the robot model and compare these results to the data obtained from the trajectory optimisation.

5. Implement the controllers on the physical robotic platform and test them on smooth and rough terrain. Additionally, the results obtained from testing were compared to the generated trajectories to confirm whether trajectory optimisation can inspire robust controller design.

## 1.3 Scope and Limitations

The robot being built was used as validation to prove that trajectory optimisation could inspire controller design for a low-cost bipedal robot, therefore the robot designed should be inexpensive while still providing sufficient explosive power for rapid movement. The robot must be made from readily available parts, to avoid shipping costs, and must also allow for consecutive testing to be done on the platform without breaking. The pneumatic actuation that was used to provide the explosive power was highly non-linear, and the on/off (bang-bang) nature thereof had to be accounted for when initialising the model for trajectory optimisation.

Physical limitations were placed on the design of the robot to ensure that the experiments were successful in the available time for the project. The robot was constrained to the sagittal plane in the form of a boom arm attached to a support platform. The robot did not have to carry its own power sources in the form of electric or pneumatic. The robot did not have to work outside the lab environment, since its purpose was to determine if trajectory optimisation will lead to feasible controllers. Due to space limitations, the robot had only a 1 m rail to travel along.

To account for the complex manoeuvres of a bang-bang robot, limitations were enforced on the trajectory optimisation, making the problems being solved simpler. The number of nodes was kept small, while ensuring that the trajectories being generated were still accurate and feasible. Trough-contact optimisation methods were used to handle contact events, with a friction coefficient of 0.75 ($\mu = 0.75$) to allow for some slipping in the robot model. The *epsilon* value used to determine the accuracy of the contact solutions was deemed acceptable when *epsilon* reached a value of 0.01 ($\epsilon = 0.01$). A smaller epsilon value increased the accuracy of the contact solutions, but due to the complexity of the problems being solved, a more accurate system would not reach a solution in a feasible amount of time.

## 1.4 Plan of Development

The dissertation started with a thorough literature review in Chapter 2, which examined legged robotics developed by different research groups, as well as different actuation methods for legged robotics. Trajectory optimisation for legged robotics and controller design was also detailed.

The methodology, Chapter 3, explained the details of the trajectory optimisation used throughout the research to generate feasible trajectories for the bipedal robot. Which in a later chapter was used to inspire controller design for the system.

The robot and platform design was detailed in Chapter 4. It explained each component

of the robotic platform as a whole, as well as the design choices made in the design of the system. Figures showed the designed as well the physical system. It also detailed the design of the electrical system, as well as the connections and communication between them.

Chapter 5 explained the controller design for the fixed body and free body models of the bipedal robot. It detailed how the optimiser was set up to achieve feasible results. The results generated were analysed to inspire a controller design for the robot.

Chapter 6 and Chapter 7 showed the testing of the controllers, first in simulation as a validation step and then on the physical system respectively. The results for the relative tests were shown and compared to the data generated by the trajectory optimisation to evaluate how the designed controller compared to the simulated and physical results.

In Chapter 8, the dissertation ends with a conclusion and plans for future work. The layout of the dissertation is presented in Figure 1.2.

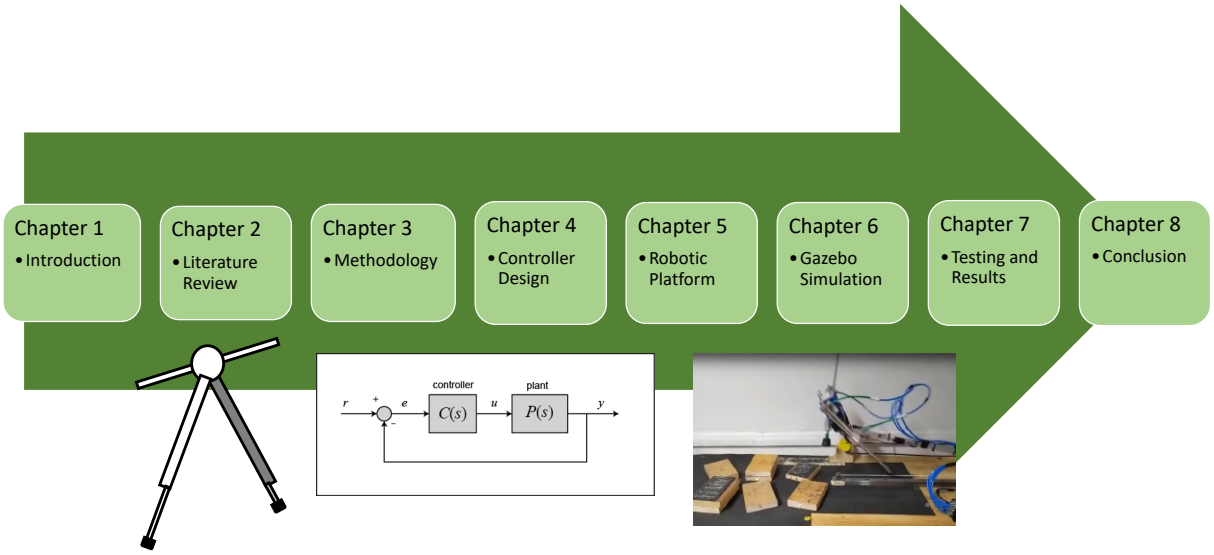



**Figure 1.2:** A graphical representation of an overlook of the thesis.

# Chapter 2

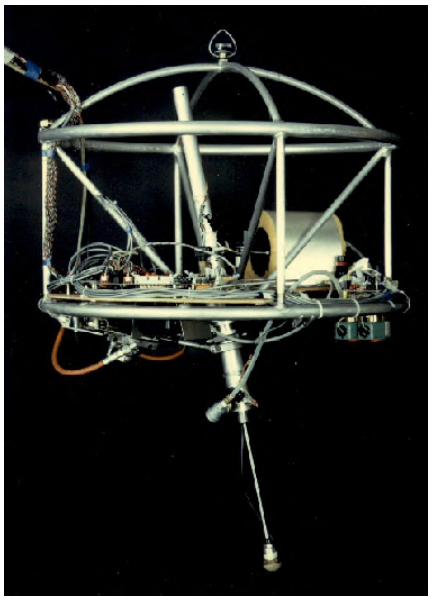
# Literature Review

In nature, legged locomotion looks like an easy executable task, especially when observing the rapid agile movements of animals. Only when studying these movements does it become evident how intricate they actually are, predominantly since the precise determinants of locomotion are still under debate [18].

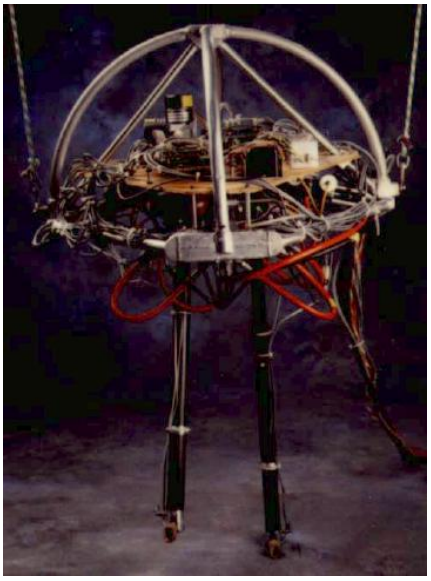
## 2.1 Legged Robotics

Animals in nature exhibit agile locomotion with low effort, obtained through thousands of years of evolution. Humans, with their ever-growing desire to create legged robots that are capable of robust movement, thus look at the years of evolution acquired through nature to aid them in the design of robotic platforms. On the scale of time, legged robotics is a new field and this can evidently be seen by the compilation of videos found throughout the internet of robots tumbling over, seemingly at random. For every video of a robot falling over, there emerges a video showing the amazing progress that has been made in the legged robotics field. Examples of these incredible platforms are the aforementioned Atlas by Boston Dynamics [5] as well as the mini-cheetah by MIT [1], but these state-of-the-art platforms are still few and far in between.

In the past, research on robots were mainly focused on stable walking [19][20] however, robotics have come a long way since their early conception, mainly due to the contributions of Marc Raibert who pioneered the ability of robotic running in the 1980s [9]. Figure 2.1 shows some of the first robotic platforms created by Raibert. The advances in computing techniques and the mathematical models of legged robots further increased the ability of robots to perform more complex movements. With the advances in the capabilities of robots, came the need for better actuators to match the needs and so emerged the series elastic actuator as well as the quasi direct-drive motor (both are discussed in more detail later in the chapter).



**(a)** A 3D Hopper [21].



**(b)** A 3D Biped [22].

**Figure 2.1:** A figure showing some of the early robots created by Marc Raibert.

Due to the research done by Raibert, early robots used prismatic actuators as legs [9] but despite the age of this leg topology, it still appears in current research [23]. These platforms frequently make use of springs or systems with equivalent properties, such as pneumatics cylinders (which were used in this research), as legs. These actuators act similarly to a series elastic system (discussed later in the chapter) and reduce the cost of transport [24]. The pneumatic version of this leg topology provides high power and speed at low weight, for these reasons it has been used in legged robotics since its inception [25] and is still used in platforms today such as the Festo Kangaroo [26] (shown in Figure 2.2).

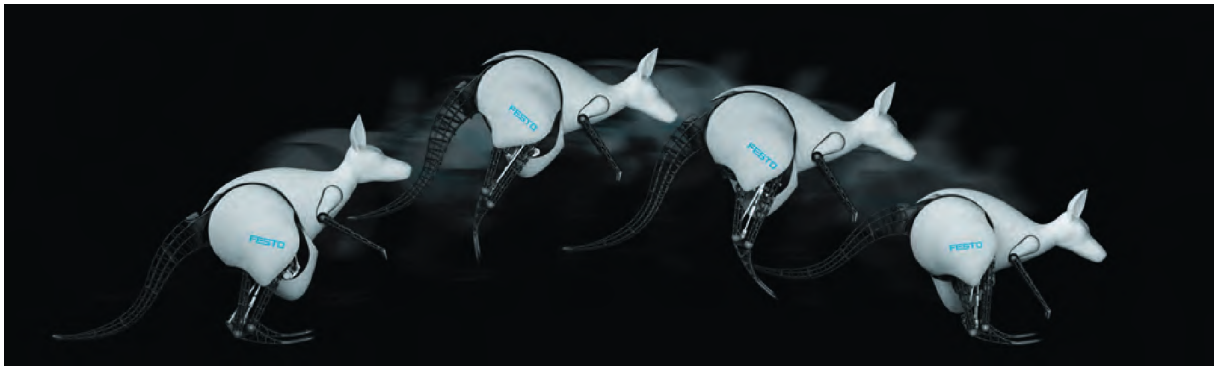



**Figure 2.2:** A figure showing the Festo Kangaroo [26].

Recently, research in the field of robotics has been led by bioinspired robotics, which seeks to match the physical output of a system to nature [27]. The animals used as inspiration for this research are very diverse and range from flamingos [28], to cockroaches [27], and cheetahs [1]. The information available in nature through years of evolution can assist

with the information required concerning gait cycles as well as limb configurations and can aid in both the design and control of these robotic platforms [27].

In the past years, some state-of-the-art robotic platforms have been devolved by publicly funded research laboratories, as well as in private research and development laboratories. Some of these robots include:

- Boston Dynamics Atlas and Spot: A biped and quadruped, respectively, manufactured for commercial use [5][10].
- ETH Zurich's ANYmal: A quadruped capable of executing several gaits, such as walking or trotting [29].
- MIT Cheetah: Made specifically for fast and agile movements and is capable of reaching speeds up to 2.45 m/s [1].

These astonishing advances in robots come from different institutions in different shapes and forms, however a robot's capabilities are still heavily impacted by actuators chosen during the robot's conceptualisation. In the following section, different actuation schemes used in legged robotics will be investigated and discussed.

## 2.2 Legged Actuation

In this section, common actuators in robotics were investigated, with the focus mainly on actuation schemes used in legged robotics. Their advantages and disadvantages were discussed, alongside past literature where these actuation techniques have proven successful.

### 2.2.1 Torque Actuation

Torque actuation investigated actuation schemes that provide rotary movement to a limb of a robot through the means of torque. Even though alternating current (AC) counterparts exit of the actuation schemes that were looked at, it was not discussed due to it being bulkier than the direct current (DC) equivalent while also requiring a specialised power source. The different torque actuators used in legged robotics were looked at, such as:

- Direct-drive motors.
- Geared motors.
- Quasi direct-drive motors.

#### 2.2.1.1 Direct-Drive Motors

Direct-drive motors refer to a motor, DC or AC, where torque is transmitted to a driven component directly connected to the shaft of the motor without the use of gears, pulleys, or belts. Direct-drive systems have smooth torque transmission and close to zero backlash [27]. Backlash refers to the angle an output shaft of a gear head can rotate without the input shaft moving [30]. The lack of backlash means that any external disturbance experienced is directly observed by the motor. Due to the lack of a mechanical drivetrain, direct-drive motors have increased efficiency compared to geared motors (maximum efficiency between 60–90%) [31].

These motors are back drivable which means they can absorb high impacts (such as a leg hitting the ground), whereas gears tend to wear down, which leads to more maintenance being required on the motor. Backdrivability refers to the ability of a system to operate in the reverse direction by applying an external force on the output [32][33]. The advantages of direct-drive motors satisfy some of the requirements of legged robots such as high efficiency, low backlash, backdrivability and high torque at relatively low speeds which are ideal for legged robotics systems.

Direct drive motors are made to perform at high speed, low torque scenarios, which is the main drawback of these motors. For legged robotics, high torque is required at relatively low speeds. For direct-drive motors to deliver the required amount of torque for a legged robotic system, leads to it becoming too large and heavy for the applications required of it. At high torque, low speed, direct-drive motors operate in regimes where Joule heating is experienced, this results in the motor mostly being operated away from its peak power and efficiency (which is closer to no-load conditions). Even though direct-drive motors are not well suited for legged robotics, they have been used in systems in the past such as Jerboa [34], Minitaur [35] and Delta Hopper [31], as can be seen in Figure 2.3.

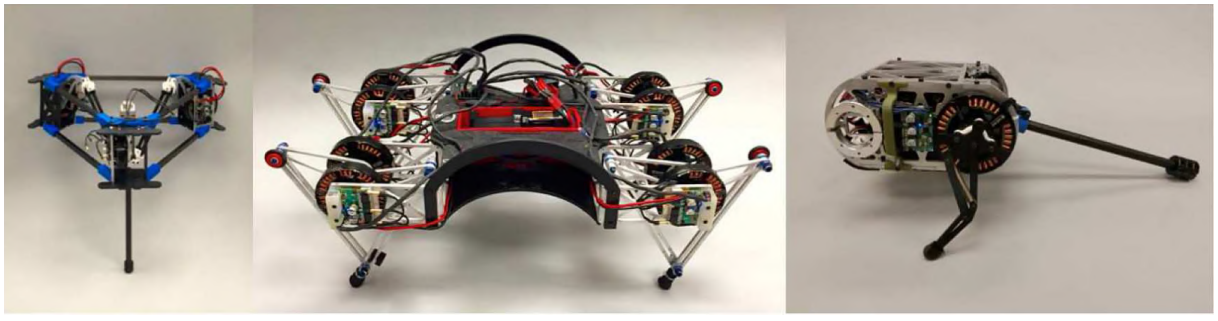


**Figure 2.3:** A figure showing the direct-drive motored robots Delta Hopper (left), Minitaur (center), and Jerboa (right) [31].

#### 2.2.1.2 Geared Motors

Geared motors are motors where gear ratios are used to increase the amount of torque that can be delivered through the use of mechanical advantage. This allows motors to operate in their ideal low-power, high-speed situation while also being able to deliver sufficient power to the driven part at a lower speed.

With the added advantage of being able to deliver the required torque to a component through gears, come a few disadvantages. Firstly, due to the use of gears, the motors have backlash which introduces a small amount of play in the angle precision of the motor. Geared motors also lack good backdrivability caused by the high gear count that limits movement in the opposite direction due to an external force. The use of an $N$:1 gear ratio causes an $N^2$ increase in reflected inertia, resulting in shock loads having much higher forces on the teeth [36], leading them to wear down under continuous high impacts. Geared motors have led to numerous robotic systems in the past, such as Tekken [37], MIT Cheetah 1 [38] and 2 [39] and Unitree Laikago [40], with some of these robots shown in Figure 2.4.

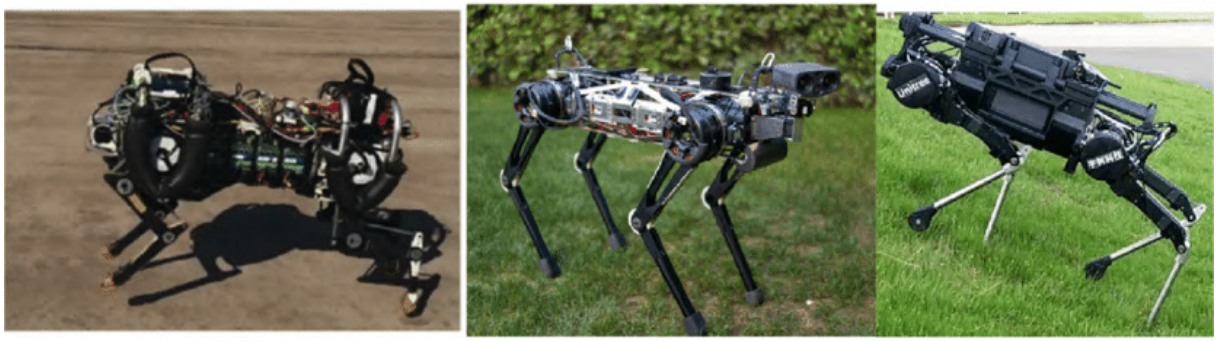


**Figure 2.4:** A figure showing the robots MIT Cheetah 1 (left), MIT Cheetah 2 (center), and Unitree Laikago (right) [41].

#### Quasi Direct-Drive

As a compromise between a geared motor's low acceleration and a direct-drive motor's struggle to achieve high torques, a quasi direct-drive (QDD) motor can be used. A quasi direct-drive motor refers to a geared motor with a gear ratio of less than ten.

This type of motor has the desired properties of low friction, high backdrivability, robust force control and selectable impedance with the main drawback being lower torque density compared to other geared motors [42]. Quasi direct-drive motors allow external forces to be sensed by studying the motor's current, this allows for more precise force control of the motor [43]. Quasi direct-drive actuators have recently been used in robotic platforms such as the Stanford Doggo [43], an unnamed monopod [44] and Blue (a human scale, 7 degrees of freedom, arm) [42].

### 2.2.2 Linear Actuation

Linear actuation refers to actuation techniques that provide linear movement to a robot. Even though there exist numerous amounts of commonly used linear actuation schemes in robotics (such as the electromechanical actuator also known as lead screw actuator) they will not be looked at. Most electromechanical actuators (the ones that are powerful enough are expensive) are not suitable for legged robotics since it does not provide the powerful explosive force a legged robot requires to leap but rather precise, slow and powerful linear movement [27].

#### 2.2.2.1 Hydraulics and Pneumatics

Hydraulics and pneumatics work on the principle of a liquid (in the case of pneumatics, a compressed gas) applying force to a rod to make it extend or retract. Figure 2.5 shows when the liquid is applied at the front end of the piston the cylinder retracts and when it is applied at the rear end the piston extends, while also exhausting liquid out of the opposite end [45].

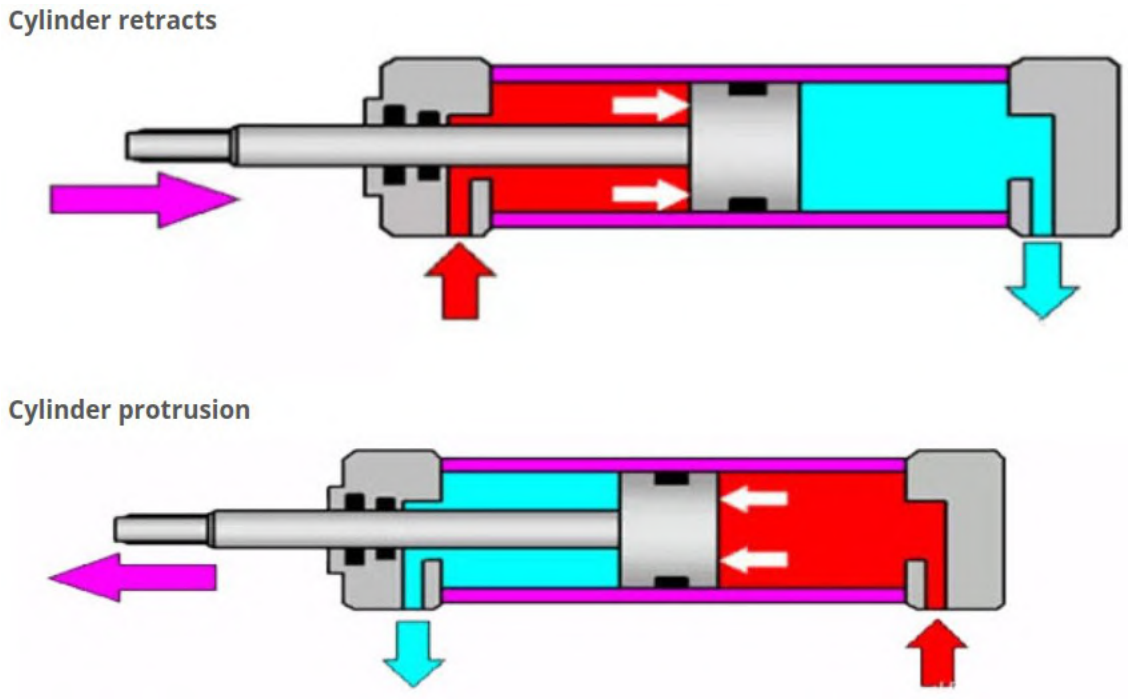



**Figure 2.5:** Figure showing the principle on which hydraulic and pneumatic systems work [45].

Even though these two systems work on closely related principles, each comes with its own strengths and drawbacks. Hydraulics can apply more force than its pneumatic counterpart, this is due to the air in pneumatics having low mass density and being easily compressible. This lower density allows pneumatic systems to be more lightweight than hydraulics, while the compressibility is ideal for legged robotics because it acts as a damper, absorbing some of the force of the legs impacting the ground. Hydraulic systems

tend to be more complex than pneumatics due to having a pump that precisely controls the flow of the liquid. They also require a return path to a reservoir instead of venting it out into the atmosphere, as hydraulic fluid is flammable, whereas pneumatics can cleanly and safely exhaust the fluid into the environment. Pneumatic systems are faster than hydraulics due to the air operating them having a higher flow rate than the oil used in hydraulic systems [46].

Hydraulics have produced robots in the past, such as the Boston Dynamics BigDog [47] but hydraulics-legged robots are used less than their pneumatic counterparts since hydraulics tends to be used in systems that require more power while pneumatics are used in small agile robots such as the monopod produced by the University of Cape Town [6] and the Festo Kangaroo [26].

While having a whole heap of advantages, pneumatics comes with some disadvantages, one of the biggest being the control bandwidth. Due to the switching in the solenoid valves (around 20-40 ms) [48] used to control the flow of the air as well as the dynamics of gas, precise position and force control are nearly impossible and this is only worsened by the on/off (bang-bang) dynamics of the actuator (the force the actuator applies is either on or off with no in-between).

### 2.2.3 Elastic Actuation

Usually in legged robotics, a high overall stiffness of an actuator produces a fast response along with high positional accuracy and simplified control. However, the high contact and interaction forces experienced by the environment can cause damage to the robot. A way of mitigating this problem is by adding measured and controlled compliance to the actuator, this is usually done in the form of some sort of elasticity [27]. Elasticity can be introduced to an actuator in the form of a spring placed between the motor and the load, this is known as a series elastic actuator (SEA), or a spring can also be placed in parallel with the motor and load, this is known as a parallel elastic actuator (PEA). Figure 2.6 illustrates the difference between the two elastic actuators.

The elastics used in legged robotics act as an energy-storing device that improves the overall efficiency of the robot. The elasticity provided by the spring can be used for force sensing, replacing the need for position control [36]. The force can be sensed by observing the displacement of the spring by using Hooke's law as shown below:

$$F = -k\Delta x \tag{2.1}$$

where $F$ is the force sensed due to a displacement in the spring, $k$ is the spring constant and $\Delta x$ is the displacement of the spring. When precise position control is needed, elastic actuators will not be a suitable choice. With a low spring coefficient comes the

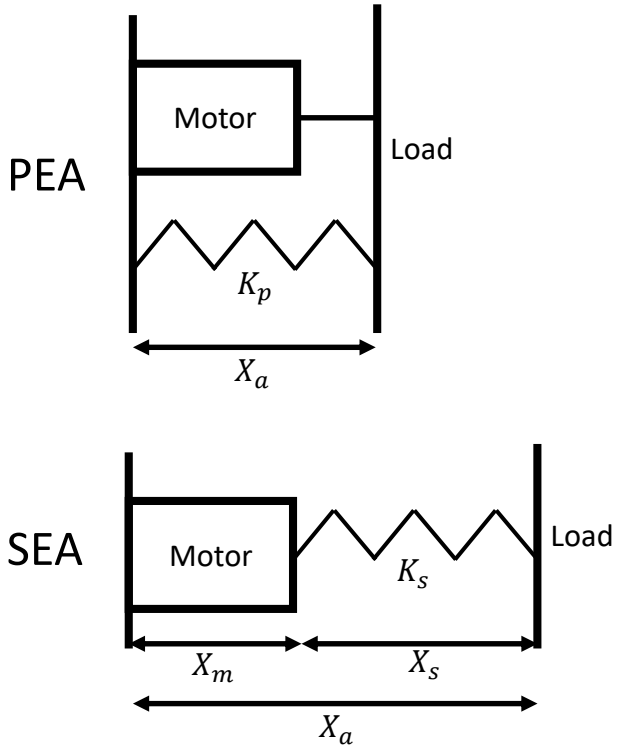



**Figure 2.6:** Figure showing how the spring is placed for the case of a SEA and a PEA [49].

benefit of eliminating linear friction and impedance and with a high spring coefficient comes the added benefit of increased force bandwidth. Unfortunately, a way does not exist to change spring coefficient dynamically, so a compromise will have to be made. Even with less precise position control, several legged robotic platforms in recent years have incorporated elastic actuators, such as the Boston Dynamics BigDog [47] and MIT Cheetah [1].

There are ways to perceive elasticity without physically using a spring. The previously mentioned, quasi direct-drive motor can be used as an elastic actuator by implementing a virtual spring through the control of the motor, this technique uses impedance control to set the stiffness of the actuator [50]. Another way to achieve a form of elastic actuation is through the use of pneumatic actuator. A pneumatic cylinder paired with a gear motor will work as a series elastic actuator, this is due to the compressible nature of gas creating an air cushion within the cylinder. This is an easy way to add elastic actuation to a geared motor while also providing powerful linear actuation at a low cost.

As a pneumatic cylinder was used in the design of the robot, past literature was investigated to determine how pneumatics were used as an elastic actuator [51][52]. The literature mentioned that due to the compressibility of air, it is an ideal fit for a legged robotic system. The compressible air in the pneumatic cylinder can be modelled as a spring with a nonlinear spring coefficient, where the stiffness is increased when the position of the rod increases [51]. The springiness of a pneumatic cylinder can be explained by examining the ideal gas law [52]:

$$PV = mR_{air}T \tag{2.2}$$

where $m$ is the mass of air in the actuator, $T$ is the ambient temperature and $R_{air}$ is the gas constant for air. The pressure, $P$, and the volume, $V$, are inversely related, which causes the cylinder to act as a spring. An increase in the mass of air in the actuator increases its apparent stiffness. When the piston's rod extends, air is pumped into the cylinder, which causes the mass of the air in the actuator as well as the apparent stiffness to increase (this correlates with the findings in [51]).

## 2.3 Trajectory Optimisation

Trajectory optimisation is a form of optimal control. Optimal control in its own right is a powerful framework specifying complex behaviours with simple objective functions, allowing the dynamics and constraints on the system to shape the resulting motion of the robot [53]. Trajectory optimisation uses a set of mathematical techniques to generate an optimal path (trajectory) subjected to several constraints and variable bounds in the process. A solution is deemed feasible once all the constraints and bounds have been satisfied, and an optimal solution is found when a feasible solution has minimised a predefined measure of performance (a cost function). To achieve a solution, a set of decision variables are varied between the variable bounds until it converges to a locally optimal solution that satisfies all the bounds and constraints set in the model [54].

To ensure that the solver achieves a realisable solution, constraints are applied to the model. The constraints include the dynamics of the system (equation of motion of the robot), boundary conditions (such as initial and terminal conditions) and world constraints (how the solver should handle contacts with the ground), more details on the constraint and optimisation can be found in Chapter 3.

Optimisation techniques are classified into two broad solution methods: Direct and Indirect. This research used direct methods, which discretises the solution, converting it to a nonlinear program [55].

### 2.3.1 Direct Methods

The solver used in this research was IPOPT using the default mumps solver [56]. The majority of solvers need a seed point (initial guess) to start the solver. A bad seed point can result in the solver getting stuck or finding an infeasible solution to the problem. Due to the solver using gradient descent, seed points are important depending on the type of direct methods being applied [54]:

- **Direct Single Shooting:** This method solves the problem by converting it into a nonlinear program. The single shooting method approximates the trajectory using a simulation and adjusts the initial conditions and the function that approximates the inputs. This method is only applicable to small applications where the problem is mostly linear, and the control is simpler [54].

- **Direct Multiple Shooting:** This is an extension of the single shooting method, that divides the trajectory into segments and each segment is represented by a simulation. This creates several smaller nonlinear problems, which have fewer decision variables that are coupled. This method is more robust than single shooting and can be applied to more challenging applications. The problem with this method is that the relationship between decision variables and constraints are highly nonlinear, and this leads to poor convergence [54].

- **Orthogonal Collocation:** In this method, the system dynamics are enforced through nonlinear constraints. Thus, the solver needs to find a feasible solution where after it attempts to find a locally optimal solution, whereas the above methods always return a dynamically feasible solution since the dynamics are simulated [16]. The initial guess can be generated from the state trajectory, $x(t)$, since the trajectories are parametrized using this method, this initial guess is easier to determine than the initial values used in the shooting methods. The path constraints can easily be applied to the trajectories, as the solver has access to all the variables used throughout. This method is used to generate large optimisation problems and can handle highly nonlinear systems, while the downfall of this method is that it creates large matrices within the optimisation problem.

In this research, a 3-point orthogonal collocation method was used, detailed more in Chapter 3. The methods mentioned above rely on continuous time trajectories (which include the system dynamics) and therefore struggle to handle contacts that change the dynamics of the system. As a result, through-contact techniques were used.

### 2.3.2 Through-Contact Methods

Direct methods were used to plan locally optimal trajectories for the robotic system and require impacting with the ground for the system to move. Most techniques used in legged robotics require *a priori* knowledge of the contacts in order to define the contact order. This method limits the search space and restricts the potential of the solver to acquire non-intuitive optimal solutions [17] and makes use of a hybrid dynamic approach to connect the various locomotion modes (aerial, contact with ground, etc.) via a discontinuous mapping.

These methods describe the trajectory using smooth dynamics up until a boundary condition where the foot makes contact with the ground. The impact is modelled as a discontinuous jump in the state space. After the initial contact event, the new set of dynamics are smooth for the remainder of the contact until the lift-off condition is reached. Additional constraints need to be added to ensure that the foot is in the correct position at the contact point with the ground and at lift-off, ensuring each segment is connected through the hybrid events [16]. This method assumes infinite friction and does not allow the foot to slip.

Alternative methods of handling contact events model the floor as a spring and damper system and use continuous dynamics to solve the contacts (this is known as soft contact models) and it requires the foot to penetrate the ground in order to produce ground reaction forces. Soft contact models produce stiff dynamics and demand short sample times, increasing the complexity of the optimisation [16]. This necessitates hand-tuning the floor dynamics (the spring and damper), which is advantageous when running on soft ground that can be easily modelled, but for rigid contacts this drastically varies the results.

In trajectory optimisation, the impulsive collisions complicate the model being solved, as these collisions are highly nonlinear and result in large forces with a high change in velocities. Recent success has been achieved using through-contact optimisation methods where the gait is solved by the optimiser, allowing for more robust solutions [16]. It further increases the robustness of the solution because this method allows the foot to slip during contact events.

The contacts, with the use of through-contact methods, are formulated as a set of complementarity constraints (a constraint in the form of $\alpha \times \beta = 0$, detailed more in Section 3.1.2.3) which models contacts as inelastic collisions with a coulomb friction model that allows the foot to slip if the friction cone is breached [16]. With the aid of slack variables and through-contact methods, the optimiser can determine the optimal gait order by making and breaking contacts [17].

Since orthogonal collocation methods were used in the research, there was no need for simulation as the dynamics were implemented using constraints and the trajectories were approximated using high-order splines which were evaluated at collocation points [16]. In past studies, this method in general resulted in poor results, due to using first-order Backwards-Euler integration methods. These methods require numerous amounts of node points (time steps) for a sufficient accuracy (accuracy of $O(1/N)$) [17]. Advances in recent literature [17], have shown an improvement in the accuracy by implementing high-order orthogonal collocation methods, such as three-point collocation. The dynamics are thus solved with third-order polynomials, leading to a smoother depiction of the state trajectory at each node and collocation point, resulting in an acceptable accuracy of

$O((1/N)^{2K-1})$ [17].

With this method, contact switching is constrained to occur at node points and forces the contact to remain for the duration of the relevant collocation point where the contact occurred. This is achieved by restricting when the required algebraic complementarity constraints can activate and/or deactivate. If enough freedom is given to adjust the time-step and by allowing contact to transpire at a node point, these constraints can apply the impulsive transition between modes. This has the benefit of not requiring a contact detection algorithm, but it does however require a variable time-step [17].

The complementarity constraints introduced by through-contact methods are incredibly difficult for the optimiser to solve. Two methods exit that are commonly used in literature to assist in solving these constraints. With the first being the penalty methods [57][58] and the second being the *epsilon* relaxation method [17][59].

The penalty method was not implemented in this research for multiple reasons. The first being, the objective function contains numerous terms, including the penalty and the cost function, making it unclear which is being optimized. As the contacts are being solved in the objective function, the constraints do not need to be satisfied, whereas for the *epsilon* relaxation methods it is guaranteed that the contacts are solved (within a tolerance of $\epsilon$) as this is a requirement for a feasible solution. In this research, a tolerance of 0.01 ($\epsilon = 0.01$) was used to solve the contact solutions.

The *epsilon* relaxation method transforms the complementarity constraints (equality equations, $\alpha \times \beta = 0$) into inequality constraints, $\alpha \times \beta \leq \epsilon$, which are relaxed by a variable, $\epsilon$. *Epsilon* relaxation works by solving the solution multiple times, initially $\epsilon$ starts large ($\epsilon = 1000$) and after each successful solve $\epsilon$ gets reduced by ten until a satisfactory accuracy has been reached.

Further techniques were implemented to reduce the number of complementarity constraints, making it easier for the optimiser to solve but keeping the accuracy. One such technique that made the complementarity constraints easier to solve was the inclusion of additional slack variables (inequality constraint in the form of $\alpha \times \beta + slack \leq \epsilon$ ), and the complementarity constraints were only evaluated at the node points, with the contact held for the duration of the collocation point [17]. More details about the implementation can be found in the relative section in Chapter 3.

## 2.4 Controlling Hopping Robots

Controlling legged robotics in general is a difficult task since they are inherently under-actuated [60]. This issue is commonly solved by using reduced order models as templates, making controller design more tractable [61]. Controllers designed from these reduced

order models generally rely on heuristics and frequently suffer from robustness issues [62], due to there being no formal link between the whole-body system and the template [63]. The lack of this formal link, allows the simulation to define a position the robot cannot achieve (such as a position not in the joint space).

The main problem with heuristics is identifying and extracting them from the manoeuvres. In a past research paper [64], a data-driven approach for designing and extracting the heuristics was developed. This spared them from having to develop meaningful cost functions for high-fidelity models in order to achieve the desired movements. This method allowed for the use of less accurate models and dynamics, with the results being tested and validated on the Mini Cheetah robot.

Observations made by numerous researchers in the past (notably Raibert [9]) suggest that simple controllers can result in stable dynamic movement in hopping robots using simple model-independent controllers [9]. The Raibert controller mainly consists of two phases, an aerial phase and a stance phase where the foot has contact with the ground. During the aerial phase, the leg is rotated into the correct touchdown position, while during the stance phase the controller stabilises the body. The foot position during touchdown at the end of the flight phase has a large influence on the speed of the robot [9]. For each forward speed, there exists a unique foot position that results in zero net forward acceleration, this is called the neutral point. For a non-zero forward speed, the neutral point is located in front of the body in the direction of travel, the higher the speed the further forward the neutral point [9]. The neutral point can be calculated as follows:

$$x_{f0} = \frac{\dot{x} T_s}{2} \tag{2.3}$$

where $\dot{x}$ is the forward speed of the robot and $T_s$ is the duration of time the foot has contact with the ground. Placing the foot before or after the neutral point can be used to control the speed of the robot by either accelerating or decelerating it, as can be seen in Figure 2.7.

## 2.5 ROS and Gazebo

Robotics Operating System (ROS) is an open-source software package that acts as middleware between different software and components, that provides hardware abstraction, message-parsing, and more [65]. Gazebo, is an application that comes prepackaged with the ROS software installation, and it allows for the simulation of different physical models in real time.

ROS works by having different nodes communicate with each other, the nodes are supervised by a master node that registers all the other nodes as well as handles the communi-

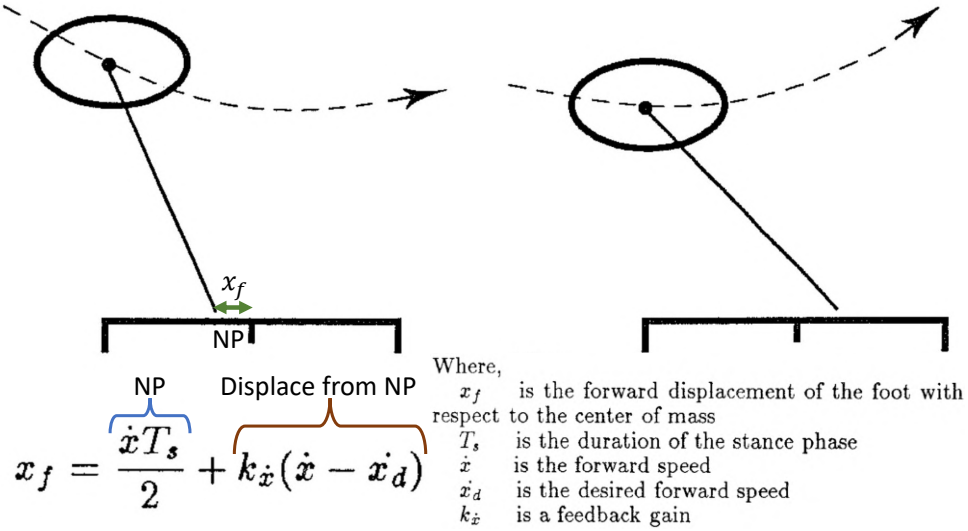
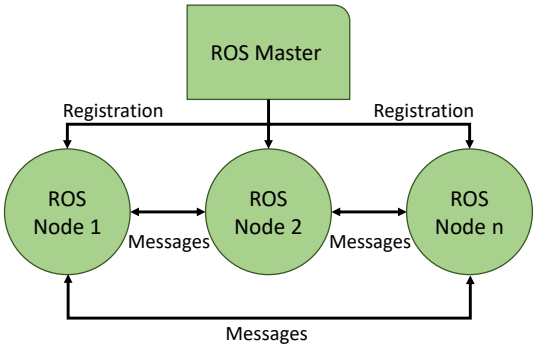
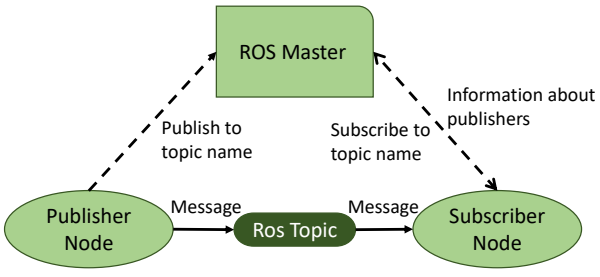



**Figure 2.7:** A modified image from [9]. The image shows how the trajectory of the body changes depending on the position of the foot. The image on the left shows the foot placed behind the neutral point (NP) and the image on the right shows the foot in front of it. If the foot is behind the neutral point, the robot accelerates and if it is in front it decelerates. The offset from the neutral point is calculated using a gain multiplied by the difference in current speed and the desired speed.

cation between them. Nodes can broadly be classified into two categories, a publisher and a subscriber, but this does not mean that a subscriber node cannot also act as a publisher and vice verse. Figure 2.8 gives a graphical representation of the interaction between nodes. The messages sent (published) between nodes are routed via a transport system known as a ROS topic. A ROS service is a specialised ROS topic that only publishes its message on request from a ROS subscriber, while a ROS topic continuously publishes its message and a ROS subscriber can access it at any time [66].




**(a)** A broad overview of ROS nodes.




**(b)** Typical communication structure between nodes.

**Figure 2.8:** A figure showing a broad overview of a ROS master and nodes, as well as a typical interaction between a ROS subscriber and publisher.

When Gazebo is launched, it creates numerous amount of ROS nodes that allow for interaction with Gazebo from ROS. These ROS nodes contain different ROS topics and services that can be subscribed or published to in order to control the model within

Gazebo. Some of the important ROS topics and services to subscribe and publish to are as follows:

- */gazebo/get_link_state*: A ROS service that can get subscribed to, which on request returns the link position within the Gazebo world.
- */gazebo/get_joint_properties*: A ROS service that on request from a subscriber returns the joint orientation of the model in the simulation.
- */model_name/joint_name/command*: Where 'model_name' is the name of the model being simulated and 'joint_name' is the name of the joint that wants to be moved. When published to this ROS topic, it moves the joint to the required position.

The physics within Gazebo are handled with the Open Dynamics Engine (ODE) and it allows for parameter tuning within the simulation [67]. With some of the most important parameters being the friction coefficients. Gazebo calculates friction according to a pyramid model, meaning that the friction forces are decoupled into two 1-dimensional constraints and then solved independently. The friction force is limited by the product of the normal force and the non-dimensional coefficients, $\mu$ and $\mu_2$, in each friction direction [67]. Figure 2.9 show a graphical representation of the friction directions.

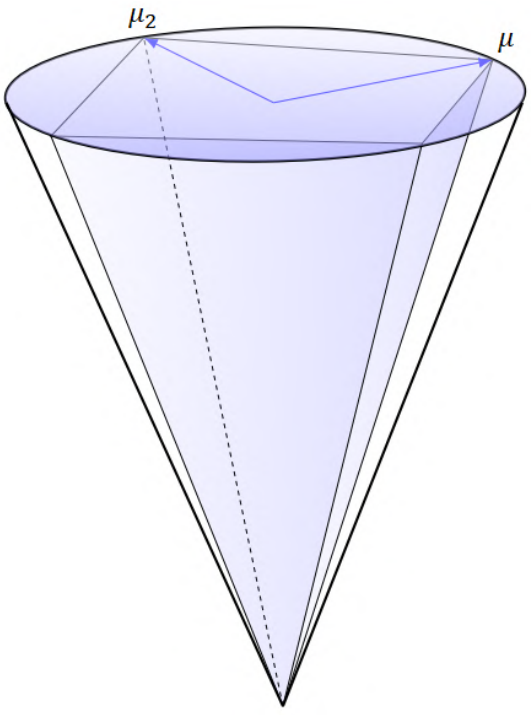



**Figure 2.9:** Figure showing the friction cone with $\mu$ and $\mu_2$, with the angle of the cone determined by the friction coefficient. The normal force corresponds to a circular slice from the cone, representing a limit on friction force magnitude in any direction [67].

Collisions within Gazebo are treated as inelastic collisions by default and are modelled using a spring and damper system. The collision between bodies or bodies and the environment are handled as follows [68]:

1. Before a simulation time step is taken, collision detection functions are called to determine what is making contact. A list of contact points are then returned by these functions. Each contact point specifies a position in space, a surface normal vector, and a penetration depth [68].

2. For each contact point, a special contact joint is created. The contact joint has extra information given to it about the contact, such as the friction present at the contact surface, how bouncy or soft the contact is, and various other properties. The contact joints are grouped in a joint "group", which allows them to be added and removed from the system quickly [68].

3. A time step in the simulation is then taken, and thereafter all contact joints are removed from the system [68].

The physics of the system are detailed within a Unified Robotic Description Format (URDF) file, this is an XML file format used in ROS to describe all the elements of the robot [69]. This holds information such as the size and the shape of the links, the inertia and how the links connect to each other to represent the model being simulated. Within the URDF file, plugins can be added that allow for sensors to be attached to the body, as well as for the capability of adding controllers to the joints [69]. The model described within the URDF file can then be imported into Gazebo, launching ROS nodes that allow the model to be controlled in simulation with the help of ROS.

# Chapter 3

# Methodology

This research investigated the feasibility of using trajectory optimisation to inspire the controller design for a hybrid pneumatic-electrical bipedal robot. To investigate this problem, a robot was developed in Inventor, followed by the development of the mathematical model of the designed robot to use in trajectory optimisation. The trajectories generated by the optimiser were analysed to aid in the controller design. In order to ensure that the trajectories generated were feasible, specific trajectory optimisation manipulations were implemented as described throughout this chapter.

## 3.1 Modelling and Simulating the System

This section of the research covers the following two main aspects, namely: the mathematical modelling of the system to obtain the equations of motion (EoM); and the trajectory optimisation techniques implemented to ensure an accurate model.

The robot concept can be seen in Figure 3.1. The robot had two legs consisting of linear actuators which were attached to the hip through torque actuators. Two models for the robot were created, a fixed body and a free body model. For the fixed body model, the body of the robot was attached to the support rig (detailed in Chapter 4) to stop it from rotating in the pitch axis, decreasing the degrees of freedom. This resulted in the problem being easier to solve. The free body model could freely rotate its body about the $y$-axis, increasing the complexity of the problem while making the robot more unstable. Both models were constrained to the sagittal plane, which meant that they could only move within the $XZ$ plane.

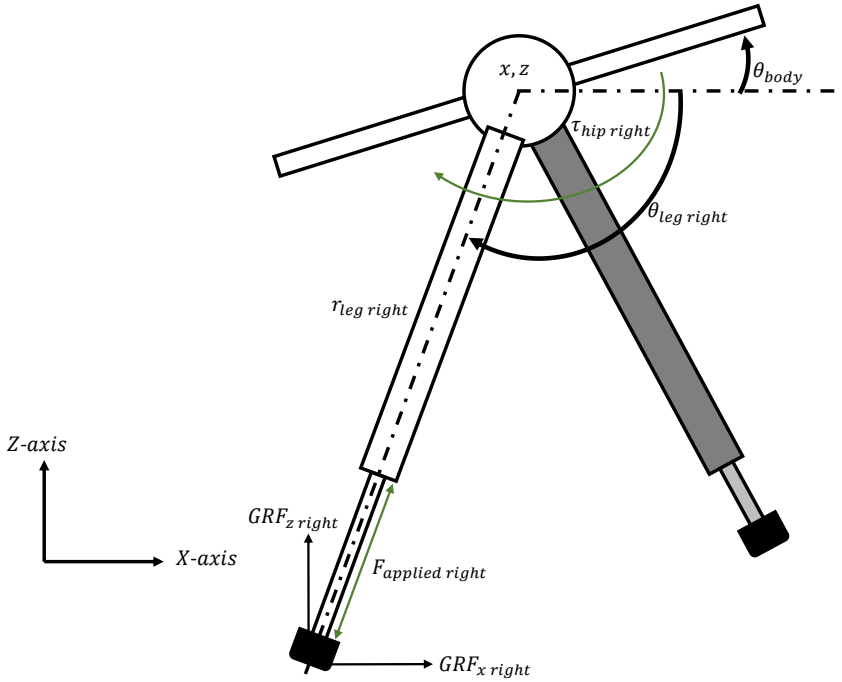



**Figure 3.1:** Model of the bipedal pneumatic-electrical robot showing the generalized coordinates as well as the applied forces and torques. These coordinates, forces, and torques also apply to the left side but were omitted for clarity.

### 3.1.1 Equations of Motion

The equations of motion were generated using the manipulator equation as shown below [53][70]:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{B}\boldsymbol{\tau} + \boldsymbol{A}\boldsymbol{\lambda} \tag{3.1}$$

where $\boldsymbol{M}(\boldsymbol{q})$ represented the mass matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ the Coriolis matrix, $\boldsymbol{G}(\boldsymbol{q})$ the gravitational potential matrix, $\boldsymbol{B}$ mapped the applied forces and torques to the generalised coordinates and $\boldsymbol{A}$ the external forces to the generalised coordinates. $\boldsymbol{q}$ was the generalised coordinates of the robot and was changed depending on the model being used. For the fixed body, the model was the following:

$$\boldsymbol{q}_{fixed} = [x, z, \theta_{leg\ left}, \theta_{leg\ right}, r_{leg\ left}, r_{leg\ right}]^T \tag{3.2}$$

The generalised coordinates for the free-body looked as follows:

$$\boldsymbol{q}_{free} = [x, z, \theta_{body}, \theta_{leg\ left}, \theta_{leg\ right}, r_{leg\ left}, r_{leg\ right}]^T \tag{3.3}$$

where $r$ defined the pneumatic actuator length in the direction of the robot's leg and $\theta$ indicated the angle of the legs as well as the body, in the case of the free body model. In

Eq. 3.1, $\boldsymbol{\tau}$ indicated the force vector, the forces working on the robot, and consisted of the following:

$$\boldsymbol{\tau} = [-b\dot{x}, F_{applied\ left}, F_{applied\ right}, \tau_{hip\ left}, \tau_{hip\ right}]^T \tag{3.4}$$

where $-b\dot{x}$ represented the damping force applied on the robot due to the friction acting on the support rig and $\tau$ was the force applied by the torque actuators at the hip. $F_{applied}$ indicated the force of pneumatic actuators at the legs of the robot, it was the net force of the pneumatic actuators. The net force consisted of the prismatic force applied by the actuator, reaction forces that occurred when it reached its maximum retraction and extension limits as well as the damping present in the pneumatic actuator, the forces were as follows:

$$F_{applied} = F_{applied\ ext} - F_{applied\ ret} - F_{reaction\ ext} + F_{reaction\ ret} - b\dot{r} \tag{3.5}$$

where $F_{applied\ ext}$ and $F_{applied\ ret}$ represented the prismatic actuation force of the pneumatic cylinder and were separated into extension and retraction, denoted by 'ext' and 'ret' respectively, to allow for maximum force constraints to be applied (discussed in more detail in Section 3.1.2.10). The forces $F_{reaction\ ext}$ and $F_{reaction\ ret}$ represented the hard stop forces that act on the cylinder when it reached its maximum and minimum extension and retraction limits (discussed in more detail in Section 3.1.2.8) and $b\dot{r}$ represented the natural damping that occurred within the pneumatic actuator. These forces were present on the right and left leg, $F_{applied\ right}$ and $F_{applied\ left}$ respectively.

The external forces, $\lambda$, in Eq. 3.1 were only the ground reaction forces since no other external forces were present, this was due to working in the sagittal plane. The forces were represented as follows:

$$\boldsymbol{\lambda} = [\lambda_x, \lambda_z]^T \tag{3.6}$$

where $\lambda_x$ and $\lambda_y$ were the ground reaction forces (duplicated for each foot) present at contact with the ground, in the $x$- and $y$-direction, respectively. The EOMs were described as an equation and were implemented as a constraint in the trajectory optimisation. Further details on how the EOMs were generated can be found in Appendix A.

### 3.1.2 Trajectory Optimisation

As mentioned in Section 2.3, trajectory optimisation required a set of mathematical equations implemented through variable bounds and constraints, which aimed to minimise or optimise a measure of performance (a cost function) [6]. Direct methods were used to

solve the trajectories, as mentioned in Section 2.3.1. Thus, the generated trajectories were discrete and divided into $N$ node points and the space between the nodes was further divided into three collocation points with a Runga-Kutta basis [17] (more on this in Section 3.1.2.1). Variable bounds narrowed the available search space, while constraints enforced certain criteria that had to be met in order to ensure the trajectories were kinematically feasible.

Since the problems that were being optimised were complex, there was no guarantee that the solver would find an optimal solution on the first attempt. To increase the likelihood that the solver would find an optimal solution as well as thoroughly explore the search space, multiple optimisations were executed from different randomly generated seed points (start point of the optimiser) using a uniform random distribution of numbers between the variable bounds. Only the generalised coordinates were randomised and the remainder of the variables were set to 0.01 (this has been shown to be successful in past research [71]). Another method that was employed to increase solvability, was to solve the seeds iteratively. Initially, the cost function was set to a constant value (the value was set to 1) and with successful iterations, more constraints were added. After each successful iteration, the optimiser was warm-started until all the constraints had been added. Once all the constraints had been added, the desired cost function was used.

In this research, the trajectories being solved by the optimiser were a long-time-horizon task. This task involved the robot starting at rest and ending at rest after travelling a fixed distance. To achieve the desired task, the trajectories were split into three separate tasks, acceleration, steady-state and then deceleration. This allowed for three smaller optimisation problems to be solved instead of one large one and allowed for more nodes, which increased the accuracy of the trajectories. These trajectories were then stitched together (as seen in Figure 3.2). The trajectories were generated in the following order:

1. Steady-state; Starting at the apex of the movement and ending at the apex of the movement after travelling a fixed distance. To cover more distance, multiple steady-state trajectories could be stitched together as they were periodic.

2. Acceleration; Starting at rest and then ending at the apex of the steady-state trajectory.

3. Deceleration; Starting at the apex of the steady-state trajectory and then ending at rest.

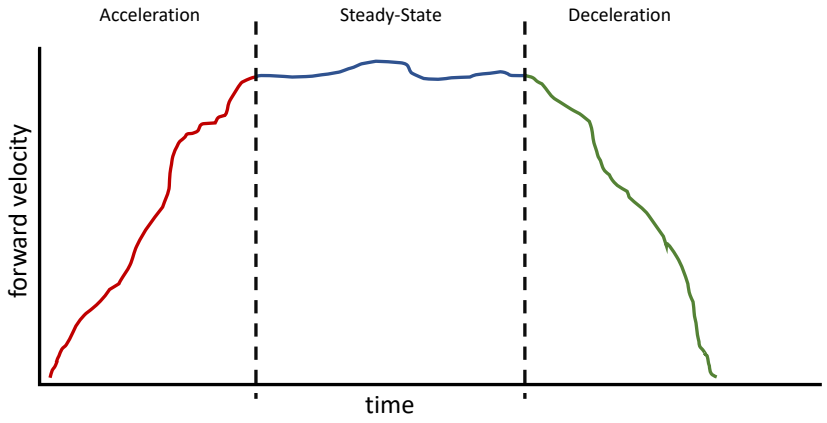



**Figure 3.2:** Figure showing a graphical representation of a stitched long-time-horizon trajectory.

The trajectories that were generated need a measurement of performance to optimise. The function being optimised is known as a cost function, while numerous cost functions exist, the following two were used to generate the trajectories:

$$\begin{aligned} J_1 &= \sum_{i=1}^{N} (\tau_{Right}(i)^2 + \tau_{Left}(i)^2) \\ J_2 &= \sum_{i=1}^{N} (\tau_{Right}(i)^2 + \tau_{Left}(i)^2) h(i) \end{aligned} \tag{3.7}$$

where $J_1$ was the cost function which aimed to minimise the absolute torque the hip motors used throughout the trajectory. $\tau_{Right}(i)$ was the right hip's motor torque at the $i^{th}$ node point and $\tau_{Left}(i)$ was the left hip's motor torque at the $i^{th}$ node. $J_2$ was a heat-based cost function which aimed to reduce the torque of the motors at every time interval $h(i)$. In both of these cost functions the torque was squared, this was done to penalise negative torque with the difference between the two cost function being, $J_1$ penalised the torque at every time interval while $J_2$ penalised small torques for long durations. The choice of cost function was depended on the optimisation task being solved (detailed in Chapter 5).

The optimisation problem was modelled in PYOMO [72] using IPOPT with the default mumps solver [56]. A number of constraints and bounds were used throughout to further guide the optimiser, as mentioned in the relative sections below.

#### 3.1.2.1 Collocation Constraint

A trajectory was split into $N$ finite elements (node points, denoted as $i$) which described the motion of each generalised coordinate along the path. The trajectory between each node was further split into more points using a Runge-Kutta basis with three collocation points, denoted as $j$ [17] (as can be seen in Figure 3.3). Thus, the trajectory was divided

into $N$ (the number of nodes was defined in Chapter 5) node points, $i \in [1, N]$, each with three collocation points in between, $j \in [1, 3]$. Additional constraints were implemented to ensure that the last value of the collocation point lined up with the value of the upcoming node, this was done to ensure a smooth trajectory and can be seen in Figure 3.3 [6]. A three-point Radau was used to solve the differential equations in Section 3.1.1, at selected time points [6]. This was done by implementing the following equations:

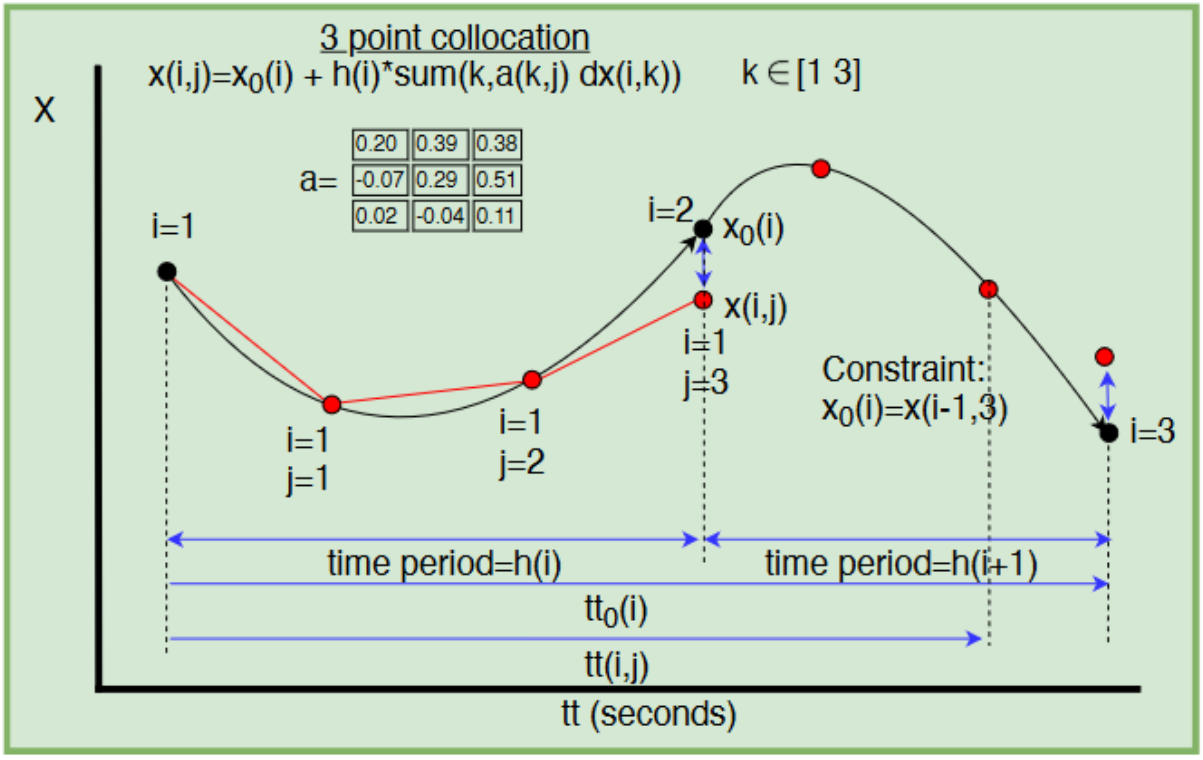



**Figure 3.3:** A figure showing a graphical representation of how the trajectory was broken up into node points ($i$) and collocation points ($j$). Additional constraints are also shown [6].

$$
\begin{aligned}
\boldsymbol{q}(i,j) &= \boldsymbol{q}_0(i) + h(i) \times \sum_{k=1}^{3} \boldsymbol{a}(k,j)\dot{\boldsymbol{q}}(i,k) \\
\dot{\boldsymbol{q}}(i,j) &= \dot{\boldsymbol{q}}_0(i) + h(i) \times \sum_{k=1}^{3} \boldsymbol{a}(k,j)\ddot{\boldsymbol{q}}(i,k) \\
\boldsymbol{q}_0(i) &= \boldsymbol{q}(i-1,3) \\
\dot{\boldsymbol{q}}_0(i) &= \dot{\boldsymbol{q}}(i-1,3)
\end{aligned}
\tag{3.8}
$$

The equations in Eq.3.8 were implemented for each state trajectory ($\mathbf{q}$ and $\dot{\mathbf{q}}$). The first two equations implemented the three-point Radue to integrate the state trajectories from node $i$ to $i+1$, whereas the last two equations ensured that the last collocation point of node $i$ was located at the same point as node $i+1$, this ensured a smooth, continuous trajectory. The difference between $\mathbf{q}(i,j)$ and $\mathbf{q}_0(i)$ was that $\mathbf{q}(i,j)$ acted on node point ($i$)

and collocation point $(j)$, whereas $\mathbf{q}_0(i)$ only acted on the node point $(i)$. The three-point collocation matrix that was used in Eq.3.8 were as follows [6]:

$$\boldsymbol{a} = \begin{bmatrix} 0.19681547722366 & 0.39442431473909 & 0.37640306270047 \\ -0.06553542585020 & 0.29207341166523 & 0.51248582618842 \\ 0.02377097434822 & -0.041548752126002 & 0.11111111111111 \end{bmatrix} \tag{3.9}$$

The through-contact optimisation techniques that were used to model the contacts required a variable time-step to ensure that a contact event is made/broken at a node point, as discussed in Chapter 2. A variable time step allowed the optimiser to increase the resolution at critical points (such as contact events) as well as decrease the resolution at less critical points (for example, the aerial phase). The time step between each element was constrained as follows:

$$\begin{aligned} 0.01h_M \leq h(i) \leq h_M \\ tt(i,j) = tt_0(i) + h(i)\sum_{k=1}^{3} \boldsymbol{a}(k,j) \\ tt_0(i) = tt(i-1,3) \end{aligned} \tag{3.10}$$

where $h(i)$ is the time-step for the $i^{th}$ node point and was bound between $0.01h_M$ and $h_M$, where $h_M$ was set according to the expected time a task would take to complete (detailed in Chapter 5). $tt(i,j)$ was the total time from the start of the trajectory to the relevant node $(i)$ and collocation point $(j)$. The first equation in Eq.3.10 bounds the time-step between a minimum and maximum value, the second equation splits the node time among the collocation points, and the third equation ensured that the start time of the $i^{th}$ node, $tt_0(i)$, matched the end time of node $i-1$ at the third collocation point, $tt(i-1,3)$ [6].

#### 3.1.2.2 Through-Contact Optimisation Method

As discussed in Chapter 2, through-contact methods were used for contact events. Using this method, the contact order was not enforced and was left to the optimiser to solve. The optimiser could determine the optimal contact order necessary to achieve the gait required for the desired manoeuvres [16]. The downside of this method was that it required numerous amounts of complimentary constraints to ensure that the GRFs were only applied when the foot had contact with the ground. This method modelled contacts as inelastic impacts that allowed for slipping due to Coulomb friction [16]. For this method to be implemented correctly, the horizontal GRF on the foot had to be split into its positive, $\lambda_x^+$, and negative, $\lambda_x^-$, components. The equation shown in Eq.3.6 was thus reformulated as follows:

$$\boldsymbol{\lambda} = [\lambda_x^+ - \lambda_x^-, \lambda_z]^T \tag{3.11}$$

With the following variables constrained to be only positive:

$$\lambda_z, \lambda_x^+, \lambda_x^- \geq 0 \tag{3.12}$$

To enforce the friction cone the following constraint was used:

$$\mu\lambda_z + \lambda_x^+ - \lambda_x^- \geq 0 \tag{3.13}$$

$\mu$, the coefficient of friction, was set to 0.75 for all simulations to allow for some slipping in the model. The following constraints were used to calculate the magnitude of the relative tangential velocity at contact:

$$\begin{aligned} \gamma + \Psi(\boldsymbol{q}, \dot{\boldsymbol{q}}) &\geq 0 \\ \gamma - \Psi(\boldsymbol{q}, \dot{\boldsymbol{q}}) &\geq 0 \\ \gamma &\geq 0 \end{aligned} \tag{3.14}$$

where $\gamma$ was the magnitude of the relative velocity, with $\Psi(\boldsymbol{q}, \dot{\boldsymbol{q}})$ being the relative tangential velocity of the foot. For these constraints to hold, it required that $\gamma$ was a positive variable, this was done with the third constraint in the set of equations in Eq. 3.14. To ensure that the GRFs were not applied to the foot when it was in the air, the following complementarity constraint was used:

$$\phi(\boldsymbol{q})^T \lambda_z = 0 \tag{3.15}$$

with $\phi(\boldsymbol{q})$ being the vertical height of the foot. This equation could only be satisfied with two conditions, when the foot is in the air the GRF had to be zero or when the GRF was applied the foot height had to be zero to ensure that it was on the ground. To ensure that the friction force lay within the friction cone when the foot slides, the following complementarity constraint was used (as seen in Figure 3.4):

$$(\mu\lambda_z - \lambda_x^+ - \lambda_x^-)^T \gamma = 0 \tag{3.16}$$

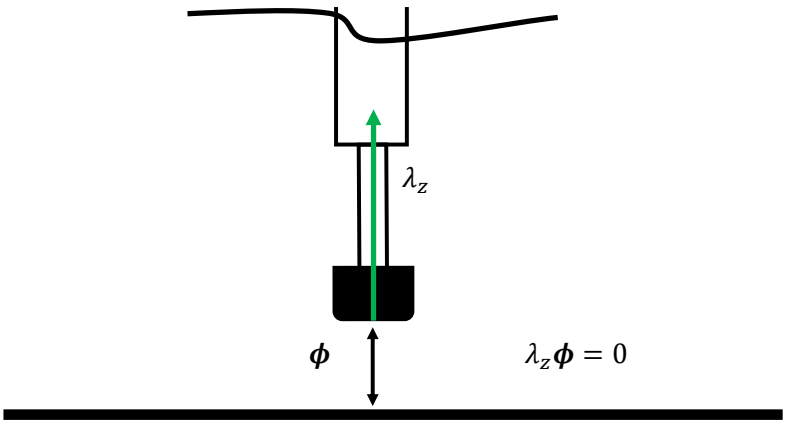



**Figure 3.4:** Figure showing a complementarity constraint for foot height and GRF. The foot height multiplied by the GRF in the z-direction must equal zero. Thus, when the foot is in the air the GRF must equal zero and vice versa to satisfy the constraint.

To ensure that the horizontal ground reaction forces opposed the motion of slipping on the foot, the following two complementarity constraints were implemented:

$$\begin{aligned}(\gamma + \Psi(\boldsymbol{q}, \dot{\boldsymbol{q}}))^T \lambda_x^+ &= 0 \\ (\gamma - \Psi(\boldsymbol{q}, \dot{\boldsymbol{q}}))^T \lambda_x^- &= 0\end{aligned} \tag{3.17}$$

The aforementioned constraints were duplicated for each foot. For these complementarity constraints to work, a variable time step integrator was used, since the complementarity constraints required the contact to be made and broken at a node point. The variable time integrator was shown in Eq. 3.10 in the previous section.

#### 3.1.2.3 Complementarity Constraints

A complementarity constraint refers to any constraint that took the following form:

$$\begin{aligned}\alpha, \beta &> 0 \\ \alpha\beta &= 0\end{aligned} \tag{3.18}$$

This constraint could only be satisfied by having one of the variables equal to zero. These types of constraints were difficult for the optimiser to solve, therefore a method called *epsilon* relaxation was used to improve convergence times [57][73]. This was done by rephrasing the constraint to look as follows:

$$\alpha\beta \leq \epsilon \tag{3.19}$$

where $\alpha$ and $\beta$ were both positive and form the two parts of the complementarity constraint [73]. To further improve the convergence rate, the complementarity constraints were summed across colocation point, $j$, and only analyse at node points, $i$, as follows:

$$\begin{aligned} \alpha'(i)\beta'(i) &\leq \epsilon \\ \alpha'(i) &= \sum_{j=0}^{3} \alpha(i,j) \\ \beta'(i) &= \sum_{j=0}^{3} \beta(i,j) \end{aligned} \tag{3.20}$$

Once solved the complementarity constraint would satisfy the value of $\epsilon$ ($\alpha(i,j)\beta(i,j) \leq \epsilon$). The value of $\epsilon$ was initially set to 1000 and the problem was solved iteratively. After each successful solve, the value of $\epsilon$ was divided by 10. As soon as a solution was not found, the optimiser was stopped and the next seed point was run. If an iteration was solved five times, the complementarity constraints were considered solved (with an acceptable tolerance, $\epsilon = 0.01$).

#### 3.1.2.4 Variable Bounds

Variable bounds were enforced on all decision variables to restrict the search space and reduce the problem size. The optimiser varied the variables between the decision bounds until the cost function was minimised.

A variable time step was used throughout the optimisation. This allowed the optimiser to vary the time step between node points ($i$) within an upper and lower bound. Varying the time step allowed the optimiser to ensure that contact occurred at a node point, it also allowed the optimiser to cluster nodes around events where the GRF were applied. To ensure rapid motion, an upper bound was enforced on the total time allowed for the trajectory, as follows:

$$\sum_{i=1}^{N} h(i) < T_{max} \tag{3.21}$$

where $T_{max}$ was obtained by estimating the expected time the optimiser would take to complete the desired trajectory (detailed in Chapter 5). It was then further manually tuned by reducing the value until the optimiser could no longer find a feasible solution.

The ground reaction forces, $\lambda$, were bound to be less than $10m_{robot}g$, where $g$ was the gravitational constant ($9.81m/s^2$). This was done to limit the forces acting on the ground to ensure that the mechanical system would not fail as a result.

#### 3.1.2.5 Start and Terminal Conditions

As the long-time-horizon task was split into three phases, each phase was generated separately and stitched together to form the complete trajectory. Therefore, acceleration, steady-state and deceleration trajectories were generated separately and then stitch to form the long-time-horizon task. These constraints describe the initial and final conditions of the trajectories. The constraints varied depending on the optimisation task being performed. The equations below describe the initial and terminal constraints used throughout the research:

- Acceleration Trajectories:
  - Start at rest:

$$\begin{aligned} \boldsymbol{q}_0(1) &= \boldsymbol{q}_{rest} \\ \dot{\boldsymbol{q}}_0(1) &= \dot{\boldsymbol{q}}_{rest} = 0 \end{aligned} \tag{3.22}$$

  - Terminate at the apex of the steady-state trajectory:

$$\begin{aligned} \boldsymbol{q}_0(N) &= \boldsymbol{q}_0(1)_{steady-state} \\ \dot{\boldsymbol{q}}_0(N) &= \dot{\boldsymbol{q}}_0(1)_{steady-state} \end{aligned} \tag{3.23}$$

- Steady-State Trajectory:
  - Start at the apex of the trajectory:

$$\begin{aligned} x_0(1) &= 0 \\ \dot{z}_0(1) &= 0 \end{aligned} \tag{3.24}$$

  - Terminate to enforce periodicity:

$$\begin{aligned} \boldsymbol{q}_0(N) &= \boldsymbol{q}_0(1) \\ \dot{\boldsymbol{q}}_0(N) &= \dot{\boldsymbol{q}}_0(1) \end{aligned} \tag{3.25}$$

- Deceleration Trajectory:
  - Start at the apex of steady-state trajectory:

$$\begin{aligned} \boldsymbol{q}_0(1) &= \boldsymbol{q}_0(1)_{steady-state} \\ \dot{\boldsymbol{q}}_0(1) &= \dot{\boldsymbol{q}}_0(1)_{steady-state} \end{aligned} \tag{3.26}$$

  - Terminate at rest:

$$\begin{aligned} \boldsymbol{q}_0(N) &= \boldsymbol{q}_{rest} \\ \dot{\boldsymbol{q}}_0(N) &= \dot{\boldsymbol{q}}_{rest} = 0 \end{aligned} \tag{3.27}$$

where $\boldsymbol{q}_0$ was the generalised coordinates only acting on the node points $(i)$, with $N$ indicating the final node point in the trajectory. Where $\boldsymbol{q}_{rest}$ referred to the generalised coordinates at rest (velocity of the robot was zero, $\dot{\boldsymbol{q}}_{rest} = 0$) with both feet touching the ground and changed depending on the model being used. The '$x$' generalized coordinate in $\boldsymbol{q}$ was exempt from the aforementioned constraints, as the robot was required to travel a certain distance.

#### 3.1.2.6 Joint Angles

The EoMs were generated using absolute angles, as these equations were more simplified, as opposed to those generated by relative angles. Bounds were placed on the leg joints to limit the motion and the velocity to a feasible range of values. The bounds were as follows:

$$\begin{aligned} lower\ bound < \theta_{hip} < upper\ bound \\ lower\ bound < \omega_{hip} < upper\ bound \end{aligned} \tag{3.28}$$

where $\theta_{hip}$ was set according to the physical limitation of the robot's legs and $\omega_{hip}$ according to the torque actuator being used on the system (the values used were detailed in Chapter 5). These bounds were set to stop the optimiser from moving the legs past the physical limits of the system, ensuring the trajectories generated were kinematically feasible.

#### 3.1.2.7 Motor Model

In order to make the trajectories realisable on an actual robotics system, the motor model for the torque actuators attached at the hips had to be accounted for. The motor was implemented using a linear motor power model, taking into account the stall torque, $\tau_{max}$, and no load speed, $\omega_{max}$, specifications for the used motors. The following constraints were implemented:

$$-\tau_{max} - \frac{\tau_{max}}{\omega_{max}}\omega(i) \leq \tau(i) \leq \tau_{max} - \frac{\tau_{max}}{\omega_{max}}\omega(i) \tag{3.29}$$

where $\tau_{max}$ was the stall torque and $\omega_{max}$ was the no-load speed of the motor and was described in more detail in Chapter 5.

#### 3.1.2.8 Hard Stops for Prismatic Actuators

For prismatic actuators, hard stops were implemented to stop the optimiser from over-extending or over-retracting the actuator. The following complimentary constraints were used:

- Minimum length hard stop:

$$\begin{aligned} l(i) - l_{min} &= \alpha(i) \\ \alpha(i) * F_{HS}(i) &= 0 \end{aligned} \tag{3.30}$$

- Maximum length hard stop:

$$\begin{aligned} l_{max} - l(i) &= \alpha(i) \\ \alpha(i) * F_{HS}(i) &= 0 \end{aligned} \tag{3.31}$$

where $\alpha(n)$ was set to determine if the actuator had reached its minimum or maximum extension or retraction limits respectively. $\alpha(n)$ was then multiplied with a force ($F_{HS}$), which represented the actuator hitting its limits, and equated to zero. The complementary constraint forced $F_{HS}$ to be applied when the actuator reached its limits, else it was zero. These constraints were duplicated for each pneumatic actuator used.

#### 3.1.2.9 Solenoid Switching

Solenoids take a set amount of time, depending on the solenoid being used, to switch from one state to another. To account for this, extra "bucket" nodes ($K$) were created to group a collection of nodes ($i$), as is seen in Figure 3.5. The prismatic force was implemented to act on the "bucket" nodes, which reduced its resolution by keeping the force at the same value for the duration of the "bucket". This prevented the actuator from rapidly extending and contracting.

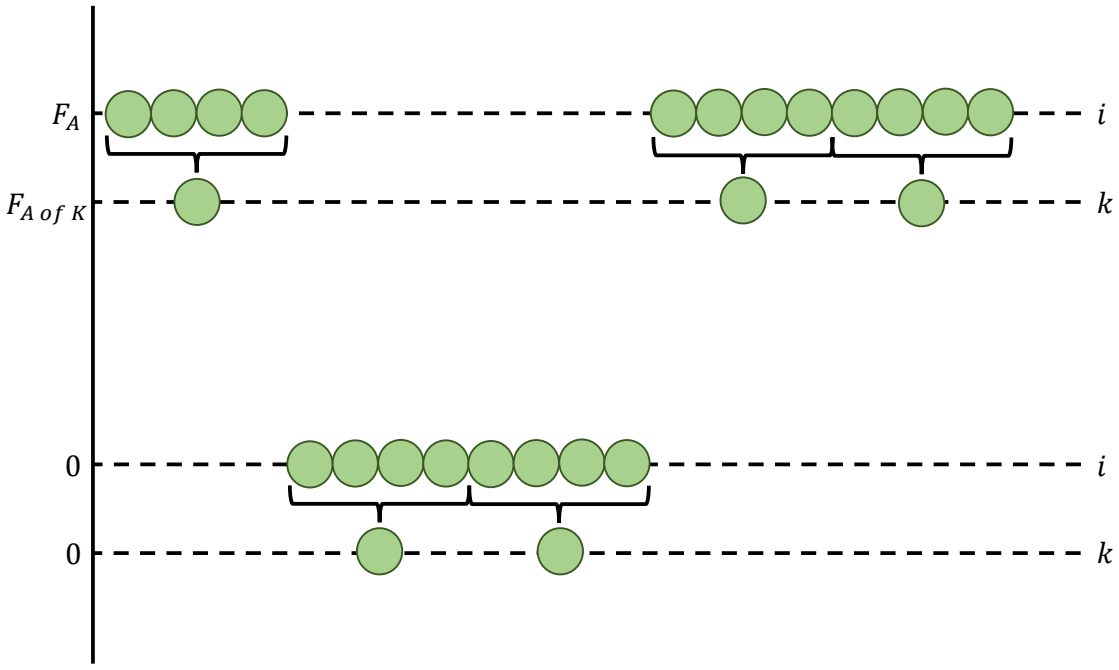



**Figure 3.5:** The graph shows a graphical representation of the nodes that are grouped for solenoid switching

Each bucket consisted of $N/K$ nodes, where $K$ was chosen based on the number of times the prismatic actuator was expected to extend or retract, and had to be a factor of $N$ to ensure that the bucket could be divided into equal amounts (the "bucket" nodes were defined in Chapter 5). Two force variables were generated, $F_{applied}$ and $F_{A\ of\ K}$, which consisted of $N$ and $K$ respectively, as shown below:

$$\begin{aligned} F_{applied}(i), i \in (1, N) \\ F_{A\ of\ K}(k), k \in (1, K) \end{aligned} \tag{3.32}$$

The following constraint ensured that the force value, $F_{applied}$, at any $i^{th}$ node point was equal to the force value, $F_{A\ of\ K}$, of the bucket it was a part of:

$$F_{applied}(i) = F_{A\ of\ K}(k), i \in (\frac{N(k-1)}{K}, \frac{Nk}{K}), k \in (1, K) \tag{3.33}$$

This method allowed for the number of nodes used to be kept high to ensure that the trajectories were smooth while ensuring that the delays of the solenoid switching states were accounted for. These constraints were duplicated for the left and right legs.

#### 3.1.2.10 Bang-Bang Dynamics

As mentioned in Chapter 2, the force a pneumatic actuator applies is either on or off with no value in-between, these dynamics had to be accounted for when the trajectory optimisation was set up. In order to achieve this, complimentary constraints were applied to the prismatic force (detailed above) working on the "bucket" nodes, $F_{A\ of\ K}$, to constrain the force to either be on or off. These constraints were implemented on $F_{A\ of\ K}$ before it was equated back to the prismatic force, $F_{applied}$, acting on the node points, $i$, as mentioned above. In Section 3.1.1, it was detailed that the applied prismatic force was split into an extension and contraction force, the complementarity constraints thus had to be applied to both of these forces as follows:

$$\begin{aligned} F_{max} - F_{A\ of\ K,extend}(k) = \alpha(k) \\ \alpha(k) * F_{A\ of\ K,extend}(k) = 0 \end{aligned} \tag{3.34}$$

$$\begin{aligned} F_{max} - F_{A\ of\ K,retract}(k) = \alpha(k) \\ \alpha(k) * F_{A\ of\ K,retract}(k) = 0 \end{aligned} \tag{3.35}$$

These constraints forced $F_{A\ of\ K}$ to either extend or retract. To satisfy the constraints, one of the products had to be zero, as detailed in Section 3.1.2.3. This meant as soon

as a force was applied $\alpha(k)$ should be zero, which forced $F_{A\ of\ K}(k)$ to be $F_{max}$, so that $\alpha(k) = F_{max} - F_{A\ of\ K}(k)$ was equal to zero. This allowed the force to always extend or retract at its maximum value. To stop the leg from extending while it was still busy contracting or vice versa, the following constraint was implemented:

$$F_{A\ of\ K,extend} * F_{A\ of\ K,retract} = 0 \tag{3.36}$$

#### 3.1.2.11 Average Velocity

The steady-state of the robot was generated with an average velocity constraint enforced, with the initial and terminal velocity being left to the optimiser to choose. Average velocity was enforced with the following constraint:

$$\dot{x}_{average} = \frac{x_0(N) - x_0(1)}{tt_0(N) - tt_0(1)} \tag{3.37}$$

where $x_0(N)$ was the final distance reached and $x_0(1)$ was the initial starting distance (this was set to 0). $tt_0(N)$ was the time it took to complete the trajectory and $tt_0(1)$ was the starting time (this was set to 0). With $\dot{x}_{average}$ being the constant that was used to enforce the average velocity on the steady-state trajectory. The average velocities were detailed in Chapter 5.

# Chapter 4

# Robotic Platform

In order to validate the trajectories, developed later in the research, a robotic platform was created. The robot consisted of pneumatic cylinders for legs attached with servos at the hip, as was described in the trajectory optimisation problem. A support rig was used to keep the robot constrained to the sagittal plane for planar motion, as well as supply the necessary power and compressed air to the actuators being used. Encoders were used to determine the state the robot was in and the data acquired was logged to a computer through a ROS communication network.

## 4.1 Platform Design

This section described the design of the robotic platform for the fixed body and free body configurations, respectively. The fixed body model was attached to the support rig to stop it from rotating, this reduced the complexity of the robot while making it more stable, while the free body model could freely rotate around the $y$-axis, increasing the complexity of the system by adding an additional DoF. This had the disadvantage of reducing the overall stability of the system. The designed robot's legs should be able to accurately and quickly move to a required position before the robot leaps into the air, allowing for it to cover an ample amount of ground with minimal effort.

### 4.1.1 Torque Actuator

The torque actuators were required to handle high impacts while being lightweight, as well as able to reach the required position with high accuracy and speed. With the torque actuators investigated in Chapter 2, a direct-drive motor was not a suitable fit since it was too heavy, does not provide high peak torque and the high backdrivability would cause the motor to move a considerable amount from the desired position due to the high impact of the legs colliding with the ground. A quasi direct-drive motor would have been ideal for the system as it does have some backdrivability protecting the gears by absorbing

some of the force from the high impacts while also providing high torque at high speeds. However, these motors are expensive and the aim of the research was to design and build a low-cost bipedal robot, hence why a geared servo motor was chosen.

Ideally, some backdrivability is preferred in legged robotics to protect the motor against the high forces experienced with impacts against the ground, whereas these servo motors typically lack backdrivability. As discussed in Chapter 2, a way to protect a motor against impacts is to add compliance to the actuator in the form of a spring. The pneumatic actuator paired with the servo would act as a series elastic actuator due to the compressibility of the air in the cylinder acting as a nonlinear spring. Available servo motors are shown in Table 4.1.

**Table 4.1:** Comparison of Servo Motors.

| Motor | Angular Range (°) | No Load Speed (rad/s) | Torque (Nm) | Mass (g) |
|---|---|---|---|---|
| DS5160 | 270 | 8.06 | 5.88 | 162 |
| LF-20MG | 180 | 6.54 | 1.96 | 60 |
| DS3235 | 270 | 8.72 | 3.34 | 60 |
| DS3225 | 270 | 6.98 | 2.45 | 60 |

From the table, the DS5160 servo motor was chosen as it provides a good power-to-weight ratio while also having a high no-load speed. This motor was capable of controlling the robot by moving the feet into the desired position at high speed and has high torque to withstand the impact of the foot colliding with the ground.

### 4.1.2 Linear Actuator

As mentioned in Chapter 2, a number of linear actuators exist in robotics however these were too heavy and slow compared to pneumatics. Thus, a pneumatic cylinder was used to allow for rapid, explosive movement of the robot. The advantages of pneumatics include: they are inexpensive, deliver high power at a low cost, and they have the added benefit of having an air cushion in the cylinder to absorb some of the high impacts of the robot landing on the ground, saving the gears of the motors from wearing down too quickly. The air cushion paired with a geared motor acts as a series elastic actuator as mentioned in Chapter 2.

As no flow values or pressure regulators were used in the design of the robot, the pneumatic cylinder always applied the maximum amount of force available to it. For this reason, the pneumatic cylinder was modelled as a prismatic actuator paired with additional constraints which enforced that when the prismatic actuator was fired the maximum force value was applied, as mentioned in Chapter 3. This proved to be a sufficient way to simplify the modelling of pneumatics [6].

The pneumatic cylinders investigated were double-action, which meant that air could be applied to both sides of the cylinder, causing it to either retract or extend, therefore solenoids were used to direct the airflow to either the top or bottom port of the cylinder. The air provided to the pneumatic cylinder was supplied by an air compressor with a maximum pressure of 8 bar. For a pneumatic cylinder to apply force, the pressure in the cylinder first had to build up to match the pressure in the compressor. The force within the cylinders quickly ramps up to the maximum value, therefore this force was assumed to be instantaneous. The maximum force a pneumatic cylinder can apply was calculated using the following equation:

$$F = P * A \tag{4.1}$$

where $F$ is the force the pneumatic cylinder could provide, $P$ is the pressure the compressor was running at and A was the area of the bore of the pneumatic cylinder. This equation was used to calculate the maximum force of the available pneumatic cylinders, shown in Table 4.2.

**Table 4.2:** Comparison of Pneumatic Cylinders.

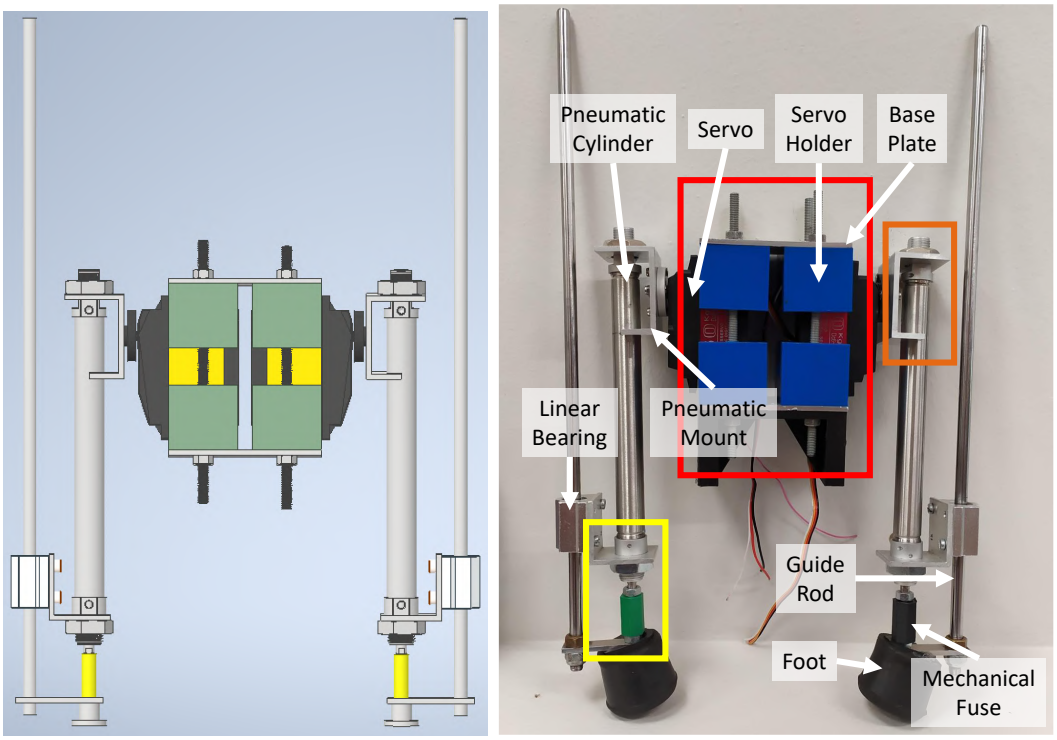
| Pneumatic Cylinder | Bore Width (mm) | Stroke Length (mm) | Maximum Force (N) | Weight ($g$) |
|---|---|---|---|---|
| DSNU-25-250 | 25 | 250 | 490.87 | 513.0 |
| DSNU-20-80 | 20 | 80 | 314.16 | 244.4 |
| DSNU-16-125 | 16 | 125 | 201.06 | 147.4 |
| C85-16-50 | 16 | 50 | 201.06 | 109.0 |

The Festo DNSU-16-125 was selected as it delivers a good power-to-weight ratio while also being long enough to act as the robot's legs. These cylinders were light and small enough to be rapidly moved to the required position. The cylinders were fed by a compressor with a maximum pressure of around 8 bar, the compressor would drop to 6 bar before charging back up to maximum again. This caused the force the pneumatic cylinders could apply to vary between 120.64 N and 160.85 N, this is equivalent to being able to lift between 12.30-16.40 kg and was more than enough force to propel the robot with a combined mass of 3.2 kg into the air. The airflow of the system was controlled by four Festo CPE10-M1BH-5L-QS-6 solenoids.

### 4.1.3 Robotic Platform

The robot was designed to be as low-cost as possible, while also being sturdy and lightweight. The CAD design as well as the final build robot can be seen in Figure 4.1. The red square indicates the servo motors that were clamped in place between two 3D-printed parts which were pressed together between two aluminium plates and then

bolted down. While the orange square indicates a machined aluminium part that slid onto the pneumatic cylinder and was fastened in place by a 16 mm nut that screwed onto the tread of the cylinder. The aluminium part was attached to a circular servo horn using screws and nylon lock nuts to stop it from vibrating loose due to the impact of the legs hitting the ground.




**(a)** Fix Body. **(b)** Free Body.

**Figure 4.1:** Figure indicating the designed model on the left and the physical robot on the right. The final design closely matched the one designed in Inventor.

The yellow square at the end of the pneumatic cylinder indicates a 3D-printed part that was designed as a mechanical fuse to protect the pneumatic cylinder from damage by breaking if something went wrong. The mechanical fuse was threaded and screwed onto the rod of the cylinder as well as the foot of the robot, joining the two. Along the length of the cylinder, a reinforcement railing system was added, which took the brunt of the impact, reducing the force on the cylinders. From the figure it can be seen that the feet on the physical robot looked different from the one designed, since the smooth aluminium feet caused the robot to slip, hence rubber was placed over the feet to increase the friction. Figure 4.2 shows a front, top and side view of the robot.

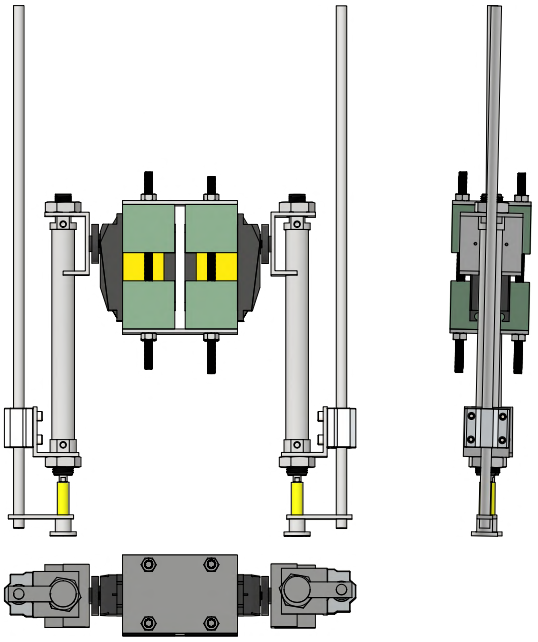


**Figure 4.2:** Figure showing an orthographic view of the robot.

A support rig was needed to constrain the robot to the sagittal plane. To achieve this, the rig was designed as shown in Figure 4.3. The rig consisted of an RS PRO linear guide rail, with a length of 1 m and a width of 28 mm, attached to a baseplate. On the linear rail rode a low friction THK Linear Guide Carriage, which gets pulled along as the robot moves down the track. On top of the guide carriage was a 3D-printed block (this was printed instead of using metal to keep the weight down) with an RS PRO 12 mm pillow block bearing attached, as well as a mount for a Generic 1000PPR Optical Encoder. The guide carriage was attached to a rubber belt which was held in place by two 3D printed pulleys, one of the pulleys was attached to an encoder. As the guide carriage moved down the rail, the belt rotated the pulleys which caused the encoder to increment, this made it possible to calculate the horizontal displacement of the robot. The yellow pulleys and belt can be seen in Figure 4.3.

An arm was mounted to the bearing on a shaft and the other end of this shaft was connected to the encoder to capture the angle the arm made with the horizon, this angle was used together with Pythagoras to calculate the height of the robot from the ground. The arm was then attached and bolted to the body for the fix body robot and for the free body robot the arm was attached to the body using a bearing on a shaft which enabled it to rotate freely. For the free body, an encoder was mounted to the arm using a 3D-printed mount and gear system which captured the angle of the body as it rotated, this configuration can be seen in Figure 4.4.

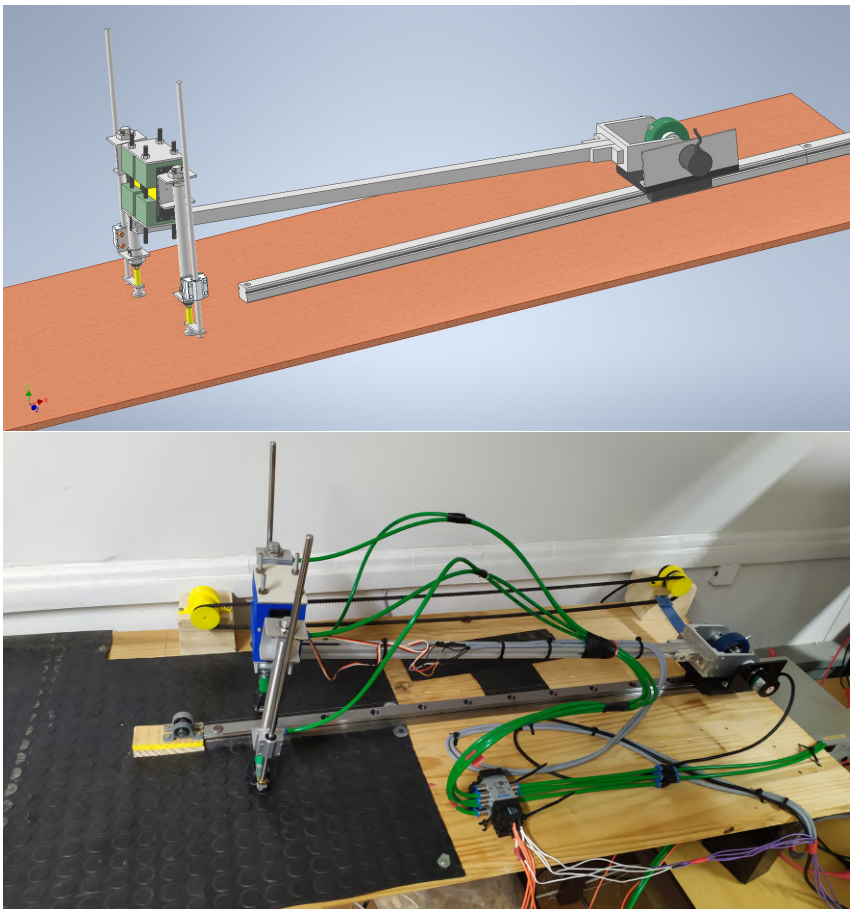


**Figure 4.3:** Inventor model compared to fully built platform.

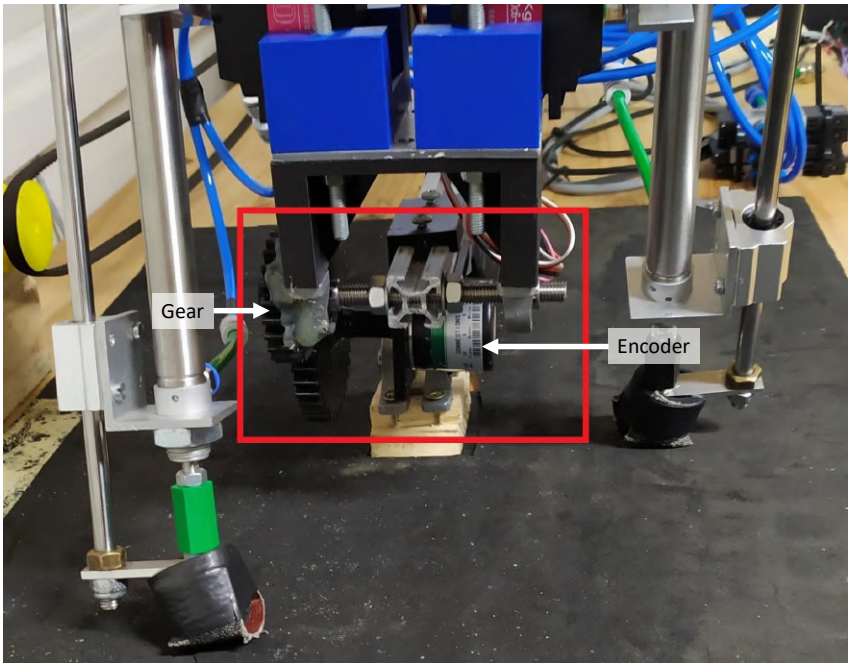



**Figure 4.4:** A figure showing the pivot point of the free body robot, with a gear fixed to the body and the other end of the encoder to capture the body angle of the robot.

## 4.2 Electrical Design

This section detailed the electrical design of the platform, as shown in Figure B.1 (in Appendix B). It also described the communication protocols used between the sensors, actuators, and microcontrollers.

### 4.2.1 Communication

In order to control the system, platforms were used that ran on different processor architectures. Figure 4.5, shows the different platforms, as well as the communication protocols used between them.

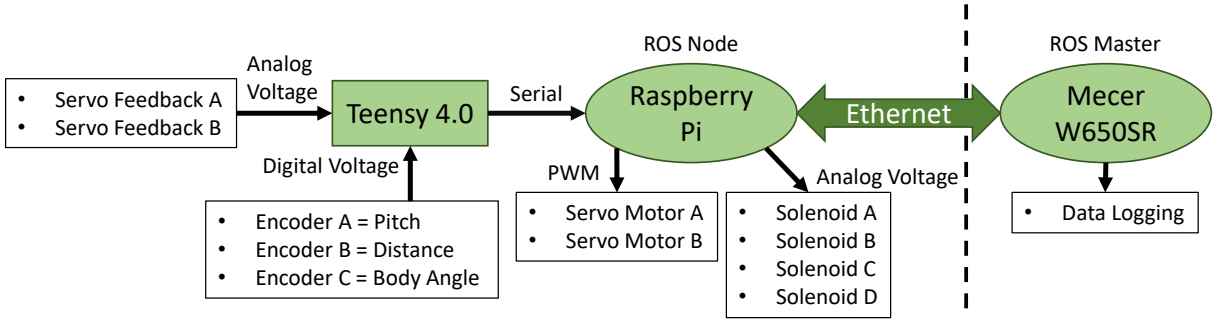



**Figure 4.5:** Figure showing the communication between the processors, sensors, and actuators. The left side of the stippled line indicates the system running on the robot and the support rig, while the right side indicates the system running on the PC

To implement the controller, three different processing platforms were used for the different levels of the control, as can be seen below:

- Teensy 4.0:
  - **Chip:** ARM Cortex-M7 @ 600 MHz.
  - **Important Features:** 40 digital I/O pins and 14 analogue input pins. Universal serial bus (USB) communication.
  - **Level of the controller:** Read sensor data.
- Raspberry Pi 4:
  - **Chip:** ARM Cortex-A72 @ 1.5 GHz quad-core.
  - **Important Features:** 28 digital I/O pins. USB communication. Ethernet / Wi-Fi.
  - **Level of the controller:** Controls the actuators.
- Mecer W650SR:
  - **Chip:** Intel i7-4700MQ @ 3.4 GHz quad-core $\times 2$.

- **Important Features:** None. USB communication. Ethernet / Wi-Fi
- **Level of the controller:** Implements the state machine that controls the robot.

The two microcontrollers (MCU) handled the sensor data and the actuation of the actuators, while the personal computer (PC) handled the high-level control of the robot. In Figure 4.5, it can be seen that the Teensy 4.0 read the sensor data before it was sent to the Raspberry Pi through Universal Asynchronous Receive-Transmit (UART) serial communication at 100 Hz, the data was received in parallel with the rest of the system running on the Raspberry Pi as to avoid interrupts. The Raspberry Pi acted as a ROS node and the PC as the ROS master, this allowed for communication without the need of changing coding languages and data types. Having the PC handling the control of the state machine kept the Raspberry Pi and the Teensy 4.0 free to control the actuators and read sensor data, respectively, without getting slowed down by controlling the robotic platform.

The ROS communication worked by having both ROS devices on the same local network, this meant that either an Ethernet cable or Wi-Fi could be used to set up the connection between them. An Ethernet connection was selected above Wi-Fi because it allowed for a more stable connection with less latency. The reason ROS was used for the communication between the robotic system and the PC was that ROS allows for more plug-and-play scenarios, as long as the data structure of the message being transmitted remained the same, the controller could easily be substituted for a different one or be applied to Gazebo simulation.

The ROS communication had two different message nodes running at any given time, both sending a custom-created message, that transmitted the data as a float64 array at 100 Hz. The PC acted as the master node between itself and the Raspberry Pi. The ROS nodes were set up within Python scripts used to control the robot. In the scripts, the ROS nodes were given a name as well as what ROS topics to subscribe and publish to. The Raspberry had two Python scripts running on it, the one published the sensor data while the other one received data from the PC containing the actuator positions. The PC had a Python script that used the sensor data for the control loop and then published the actuator positions back to the Raspberry Pi, the sensor data was then also logged to a text file on the PC. Figure 4.6 shows a graphical representation of the interaction between the ROS nodes within the Python Scripts.

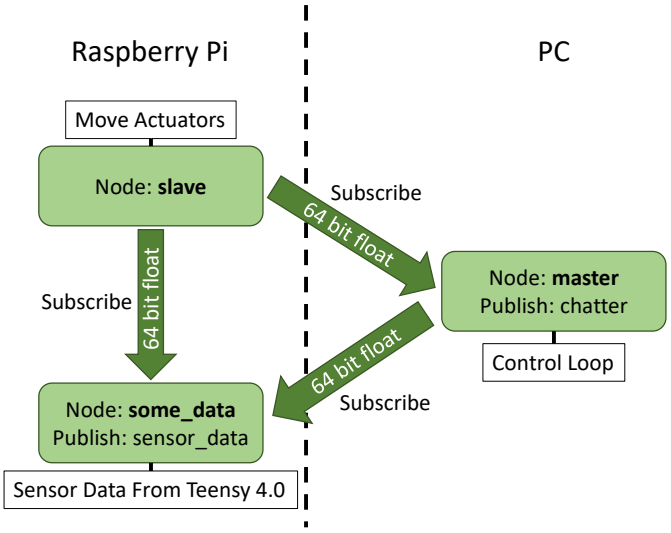



**Figure 4.6:** The figure shows the data passed between the ROS nodes running on the Raspberry Pi and the PC. The arrow indicates to which ROS topic a node subscribed. The data transmitted between nodes were sent as a message containing a float64 array.

### 4.2.2 Electronics

The printed circuit board (PCB) from the design of the electronic system (detailed in Figure B.1) can be seen in Figure 4.7. The empty connections were designed to allow for future component additions.

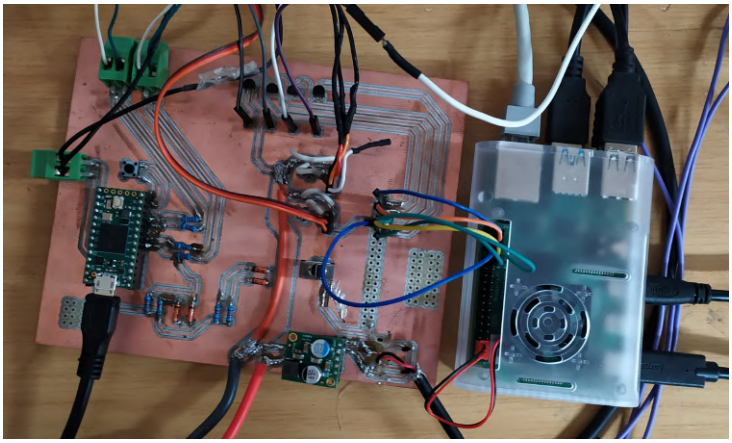


**Figure 4.7:** Figure showing the PCB used for the electrical connections.

The sensors, actuators and microcontrollers required different supply voltages to operate as indicated in Table 4.3. A power distribution circuit was thus designed to deliver the required voltage to the different components. The power distribution was split between two different power supplies, one with a single channel output capable of delivering 30 A and the other one having two output channels capable of 10 A each, this was done to lower the amount of voltage step-down required for the system. The Teensy 4.0 was powered through the Raspberry Pi using a USB cable, which also handled the serial communication between the two devices. The Raspberry Pi and servo motors received power from the

30 A power supply. As the servo motors deliver peak power with a supply voltage of 8.4 V with a stall current of 6.2 A and the Raspberry Pi required a supply of 5 V with a maximum current draw of 3 A. The power supply was set to 8.4 V @ 20 A to allow for the required current draw of the components. For the Raspberry Pi, a 5 V step-down regulator (S13V30F5, 5 $V_{out}$ @ 4 A) was used to get the required supply voltage from the 8.4 V connection. The encoders as well as the solenoids were connected to a separate channel on the power supply. The channel supplying the encoders was set to 10 V @ 250 mA (the encoders had a total current draw of $40 \times 3 = 120$ mA ) and the channel supplying the solenoids was set to 24 V @ 350 mA (the solenoids had a total current draw of $54 \times 4 = 216$ mA).

**Table 4.3:** Supply voltages required for the robotic platform.

| Device | Voltage (V) |
|---|---|
| Teensy 4.0 | 5.0 |
| Raspberry Pi | 5.0 |
| Generic Optical Encoder 1000PPR | 10.0 |
| DS5160-270-FB Servo | 8.4 |
| Festo CPE14-M1BH-5JS-1/8 Solenoid | 24.0 |

As the solenoids were connected to a power supply, transistors were needed to allow the Raspberry Pi to control when the solenoids switched on or off. The transistor chosen was the BS107A MOSFET, because the gate is isolated from the source and drain, requiring no extra connection in order to operate the transistor. The chosen transistor was capable of withstanding a source-drain voltage of 30 V, this was a necessity as the solenoids would require a source-drain voltage of 24 V across the transistor for it to be powered. The source and the drain of the transistor were placed between the ground connection of the solenoid and the ground of the power supply, respectively. This created an open circuit when the gate of the transistor was powered. The gate of the transistor was connected to a GPIO pin on the Raspberry Pi, allowing it to be closed when the GPIO pin on the Raspberry Pi is toggled “high”.

The Raspberry Pi was responsible for controlling the servo motors, from its GPIO pins, by transmitting a pulse-width modulated (PWM) signal to the motors. The duration of the duty cycle of the PWM signal controlled the angle of the servo motors. The feedback line of the servo motors were connected to the Teensy 4.0 since the Raspberry Pi was not capable of reading the analogue voltage supplied by this wire. Testing on the feedback wire indicated that its voltage did not reach a value greater than 3.3 V. This indicated that no voltage protection was required for the feedback wire connected to the Teensy 4.0 (the Teensy 4.0’s pins were only rated for voltages up to 3.6 V).

A combination of encoders was used to obtain the vertical displacement, horizontal displacement and the body angle of the robot. The optical encoders have two data channels

A and B, these two channels were pulse signals that are out of phase by 90°, this indicated the direction the encoders were turning. The A and B channels of the encoders were connected to digital pins on the Teensy 4.0. The encoders required a pull-up resistor to pull the voltage of the data lines to a readable level. Pull-up resistors of 1.2 $k\Omega$ were connected between the data lines of the encoder and the 3.3 V rail of the Teensy 4.0 in order to pull the voltage up. The encoders were powered using 10 V meaning that a potential fault in the encoders could cause a voltage spike up to 10 V. To protect the Teensy 4.0 against this, a voltage protection circuit was implemented. A Zener diode voltage protection circuit was built for this purpose, shown in Figure 4.8. A 1N746A Zener diode was used as protection on the encoder data lines and had a reverse breakdown voltage of 3.3 V.

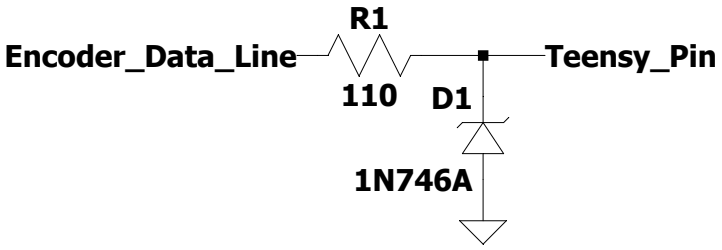



**Figure 4.8:** Figure showing the Zener protection circuit.

The resistor was calculated as follows:

$$\begin{aligned} R &= \frac{V_R}{I_\mathrm{T}} \\ R &= \frac{10 - 3.3}{0.02 + 0.04} = 110\ \Omega \end{aligned} \tag{4.2}$$

where $V_R$ was the voltage across the resistor (10 V was the maximum voltage the Zener had to drop and 3.3 V was the output voltage of the Zener diode) and $I_T$ was the total current in the circuit (40 mA for the current from the encoder and 20 mA for the biasing current of the Zener diode). The 110 Ω resistor was placed between the encoder data line and the Teensy 4.0 pin and would protect it in the case of a fault.

# Chapter 5

# Controller Design

Controlling the robot using the generated trajectories was a difficult task. Although methods were used to model the robot as accurately as possible, discrepancies between the simulated robot model and the actual physical robot will always exist. This was due to using approximations to simplify the mathematical model of the system, which included how the pneumatic actuators were modelled as bang-bang actuators, as well as the friction coefficients differing, weight distributions not being the same, the effect of the support system had on the robot as well as optimisation techniques modelling a continuous system as a discrete system. The bang-bang dynamics and switching delays were difficult to model accurately, making it a challenging task using classical feedback control schemes. A method of control was thus needed that allowed the overall motion of the generated trajectories to be mimicked on the platform. In order to achieve this, the trajectories generated were analysed to aid in the controller design of the system.

## 5.1 Trajectory Analysis of Fixed Body Model

To verify the methods and to keep the initial complexity low, a fixed body model was first used, which was then followed by the free body model, once the methods were verificated. The fixed body was constrained from rotating around the $y$-axis by being attached to the boom arm, removing one degree of freedom from the optimisation, making it a more simple problem to solve and eventually design controllers for. As mentioned, the robot was constrained to the sagittal plane, constraining its movement to the $XZ$ plane.

### 5.1.1 Optimiser Setup

To ensure accurate trajectories, the parameters of the trajectory optimisation were implemented to match the robot designed in Chapter 4. In this section, the parameters used for the fixed body model were described in detail.

### Parameter Identification

As mentioned in Section 3.1.2.4, a variable time-step was used throughout the optimisation. This meant that the time step between nodes was bound between an upper and lower value. The lower bound was set to 0.001 sec and the upper bound was set to 0.1 sec. To ensure that the trajectories completed the task within the desired time, an upper bound, $T_{max}$, was placed on the total time allowed for a trajectory (shown in Eq. 3.21), where $T_{max}$ was obtained by decreasing the value until the optimiser could not reach a solution, this was done to keep the generated trajectories as fast as possible.

Table 5.1, shows the time allowed for the trajectory to finish, as well as the distance the trajectory had to travel. The robot was confined to a 1 m rail which meant that the acceleration, steady-state and deceleration had to happen within the distance of the rail. Thus, the distance was split up into segments, allowing the greatest distance to be covered by the steady-state phase of the trajectory since this is where the robot would achieve its optimal gait pattern. The distance each trajectory should travel was determined by decreasing the allowed distance for the acceleration and deceleration phase until the optimiser reached a point of infeasibility as rapid acceleration and deceleration motions were wanted, the remainder of the distance was assigned to the steady-state phase. Additional constraints were implemented on the steady-state portion of the trajectory that enforced an average velocity (discussed in Chapter 3) and periodicity.

**Table 5.1:** Time Bounds and Travel Distance for Fixed Body Robot.

| Trajectory | $T_{max}$ $(sec)$ | Distance (m) |
|---|---|---|
| Acceleration | 0.3 | 0.2 |
| Steady-State | 2.0 | 0.65 |
| Deceleration | 0.4 | 0.15 |

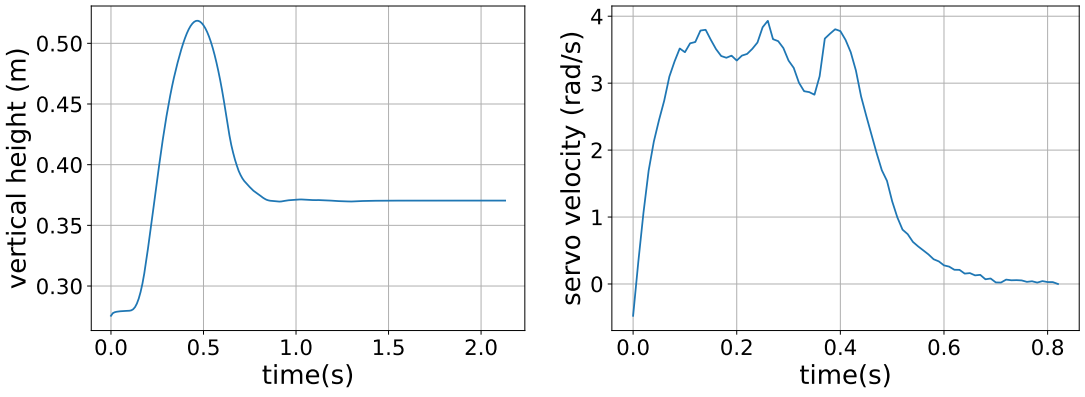
For the optimisation to match the system, the robot's parameters were required. Due to an off-centre load being attached to the actuators as well as the fact that actuators very rarely match the specification listed, additional tests were done to identify the parameters, so the optimisation could more accurately match the physical system. The parameters obtained are shown in Table 5.2.

The pneumatic force as well as the pneumatic damping coefficient were obtained by first determining the vertical height the physical system could reach. The robot reached a vertical jump height of 0.51 m. A vertical jump test was set in the optimiser and the cost function was changed to maximise the vertical height, the pneumatic force was then tuned until the optimiser reached the same height as the system. The pneumatic damping coefficient was tuned to a value that ensured that the cylinder's rod would not achieve a velocity greater than 1.5 m/s (the velocity was obtained from the datasheet [74]). In order to obtain the speed of the motor, the leg was swung from 0° to 180°, the data was

**Table 5.2:** Robot Parameters.

| Variable | Value |
|---|---|
| Nodes | 100 |
| "Bucket" Nodes | 10 |
| Body Weight | 1.20 kg |
| Leg Weight per Leg | 0.45 kg |
| Minimum Leg Length | 0.07 m |
| Maximum Leg Length | 0.19 m |
| Bearing Damping Coefficient | 9.00 |
| Pneumatic Damping Coefficient | 90.00 |
| Pneumatic Force | 100.00 N |
| Maximum Motor Torque | 5.00 Nm |
| Maximum Motor Velocity | 3.93 rad/s |

recorded and was then used to calculate the velocity to be 3.93 rad/s. The results of these two tests can be seen in Figure 5.1. After some testing, it was found that the motor could not keep up with the demand that was placed on it. The maximum torque of the motor within the optimisation was thus lowered by 15% to accommodate for this.




**(a)** Vertical jump test. **(b)** Servo velocity test.

**Figure 5.1:** Figure showing the results of the jump test and servo velocity test.

Variable bounds were placed on the joint angles, as mentioned in Chapter 3. The joints of the robot were bounded between 0° and 180°, with the angular velocity bound between -3.39 rad/s and 3.93 rad/s. Figure 3.1 shown in Chapter 3 indicates the axis system of the robot.

#### 5.1.1.1 Optimisation Results

Three steady-state trajectories were generated with three different average velocities, which were then analysed to observe common trends. As previously discussed in Chapter 3 the cost function, $J_1$ minimised the torque over the entire trajectory while $J_2$ minimised the torque at every time instant. The trajectory with the lowest average velocity used $J_2$ to generate the motions of the robot, while the other two trajectories used $J_1$. Each of the

steady-state trajectories consisted of two steps and started and ended at the same height and angles, this was done so that the trajectories were periodic in order to stitch multiple steady-state trajectories together to achieve a greater distance. This was the case for both models being generated. The common trends looked at in these steady-state trajectories were as follows:

- The leg angle at touchdown.
- The leg angle at lift-off.
- If the robot land with the prismatic actuator extended or retracted.
- If the robot takes a step or directly hops into the air after landing.
- Amount of steady-state trajectories required to reach the end of the 1 m rail.

Figure 5.2 shows the vertical height of the generated trajectories, as well as at what height the pneumatic cylinders were extended or retracted. Table 5.3 indicates the touchdown angles and lift-off angles for each of the three trajectories, the angles were obtained by analysing when the ground reaction forces on the feet were applied and broken.

From the three steady-state trajectories' data in Figure 5.2 and Table 5.3, it was noted that the legs impacted the ground with an angle between 90-100° and left the ground around 110-120°, this indicated that the robot gave a small step between touchdown and lift-off. The robot landed with its one foot at the touchdown angle, swung the leg back wards to the lift-off angle, and leapt into the air. The robot started this pattern with its left foot before alternating to the right. For each of the three trajectories, the robot landed with its feet retracted and extended them when leaping, as soon as the robot reached the apex of the jump it started retracting its legs. The robot's legs started to retract at the apex, and when it landed, it took a step on the ground before firing the pneumatic cylinder and leaping into the air.

**Table 5.3:** Analysis of Touchdown and Lift-Off Angles for Fixed Body.

| Trajectory | Touchdown Angle (°) | | Lift-Off Angle (°) | |
|---|---|---|---|---|
| | Left Leg | Right Leg | Left Leg | Right Leg |
| 0.15 m/s | 99.72 | 97.38 | 110.32 | 113.52 |
| 0.30 m/s | 94.70 | 90.56 | 116.20 | 110.72 |
| 0.40 m/s | 105.79 | 91.19 | 118.49 | 112.18 |

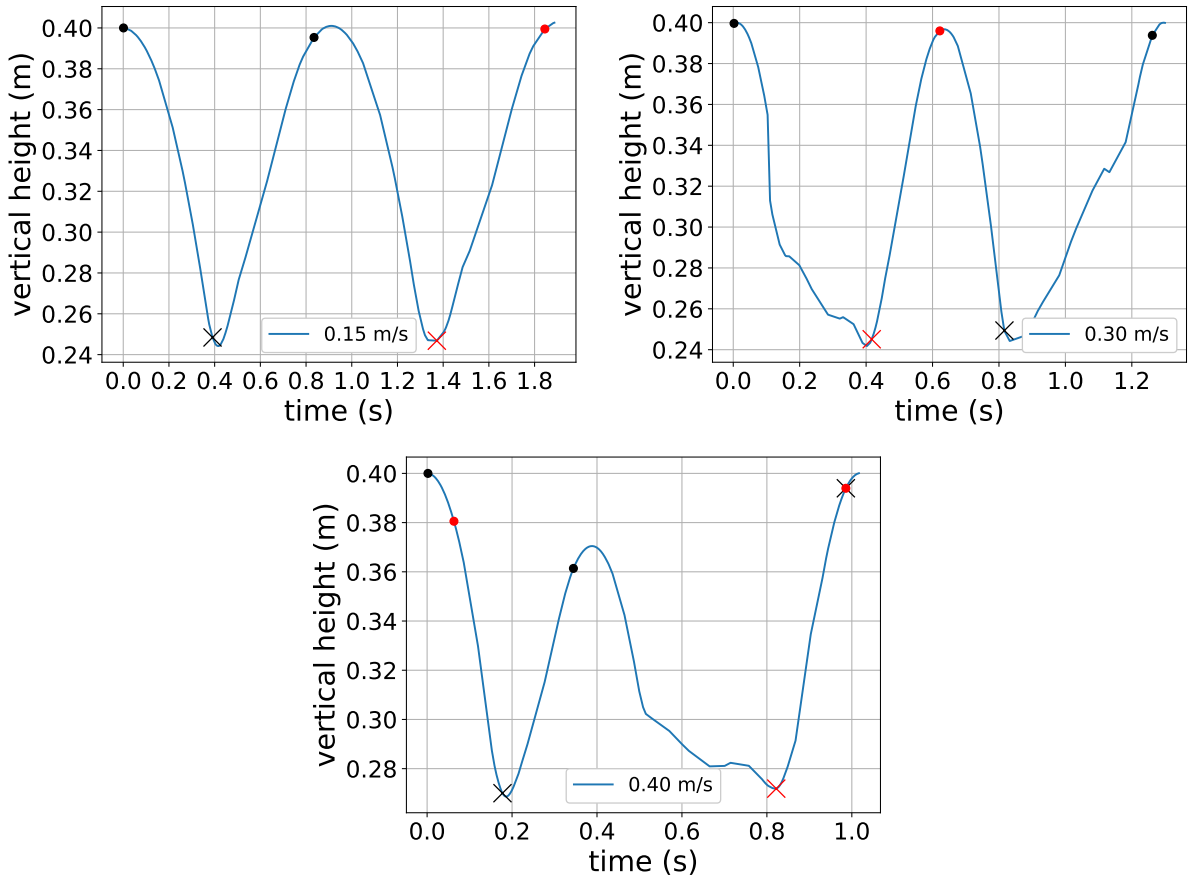



**Figure 5.2:** Figure showing the results obtained from the optimiser for the fixed body while enforcing three different average velocities. The maximum velocity was obtained by increasing the enforced average velocity until the optimiser returned infeasible results. The red and black '×' and 'o' indicates when the prismatic actuator activates and retracts for the right leg and left leg, respectively.

Analysing the generated trajectories, it was clear that a pattern emerged in the motion of the robot. The motion of the simulated robot had the following pattern:

1. Prepares leg for touchdown.
2. Takes a step on the ground before leaping in the air.

Due to this repeating pattern, a state-machine controller emerged to control the robot's motion while in the air and on the ground. To design the controller, the lift-off, and touchdown angles observed in the trajectories were used to move the legs to the required position, while the heights at which the robot extended and retracted the cylinders were used as control heights to toggle the state of the robot. Figure 5.3 indicates the basic shape of the state-machine controller. The first control height was just before the apex of the movement, this moved the robot's legs into the required position to prepare for touchdown and below the second control height, the controller moved the robot to prepare for lift-off.

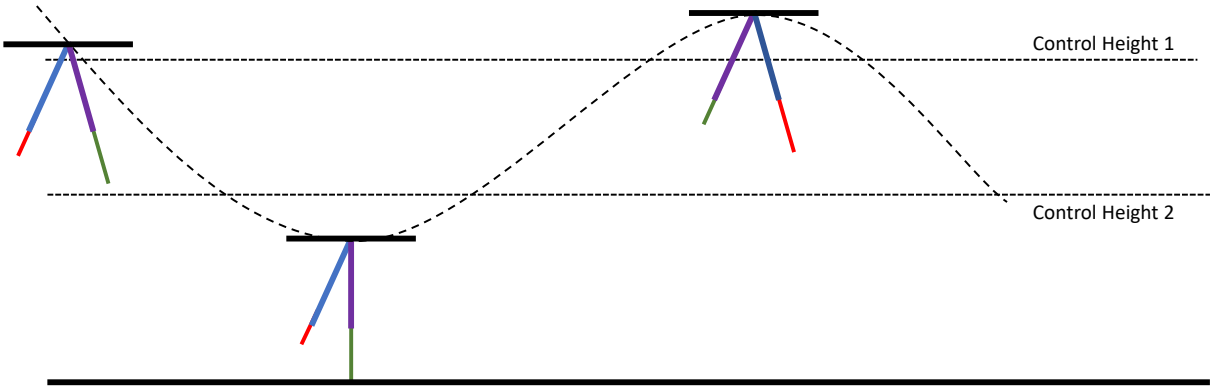



**Figure 5.3:** Figure showing the two control heights for the state machine. These heights transitioned it from one state to the next.

The state machine which emerged from the trends in the trajectory optimisation closely resembled the Raibert controller mentioned in Chapter 2. The foot placement of the robot was thus observed during the steady-state and deceleration trajectories to see if it correlated with the neutral point analysis done by Raibert [9]. As mentioned in Chapter 2, the neutral point (distance away from the body the foot should be placed for unaccelerated movement) of a hopper robot's foot can be calculated through Eq. 2.3 [9].

Table 5.4, shows the neutral point as calculated from the Raibert controller and compared it to the position the optimiser placed the robot's feet away from its body. From analysing the results, it was clear that for the fixed body robot during the steady-state the optimiser placed the robot's feet close to the neutral point calculated through the Raibert controller, while for deceleration the feet were placed in front of the neutral point. For the fixed body trajectories, it was thus clear that a form of the Raibert controller emerged, as the robot placed its feet near the centre point to achieve unaccelerated travel and in front of it to decelerate. These calculations were not done for the acceleration phase, as the robot accelerated from rest with only one initial contact, which meant that the neutral point would be zero. As mentioned in Chapter 2, the Raibert controller indicated that a foot placement behind the neutral point resulted in acceleration, a foot placement in front of the neutral point resulted in deceleration and to maintain a given speed the foot was placed close to the neutral point of the body [9].

**Table 5.4:** Neutral Point Analysis of the Fixed Body Robot.

| Trajectory | Contact | Neutral Point (m) | Foot Placement (m) |
|---|---|---|---|
| 0.15 m/s | 1 | 0.048 | 0.046 |
| | 2 | 0.060 | 0.069 |
| | Deceleration | 0.032 | 0.097 |
| 0.30 m/s | 1 | 0.287 | 0.267 |
| | 2 | 0.155 | 0.152 |
| | Deceleration | 0.074 | 0.143 |
| 0.40 m/s | 1 | 0.068 | 0.070 |
| | 2 | 0.201 | 0.171 |
| | Deceleration | 0.100 | 0.151 |

With the steady-state trajectories generated, acceleration and deceleration trajectories were generated to stitch to the steady-state in order to obtain a full trajectory. The optimiser accelerated the robot from the rest phase by starting both legs at 100° and firing both pneumatic cylinders and decelerated by placing the foot down in front of the body and then moved both legs to match the orientation of the rest phase at the start. Figure 5.4 indicates the robot's orientation at the start of the rest phase.

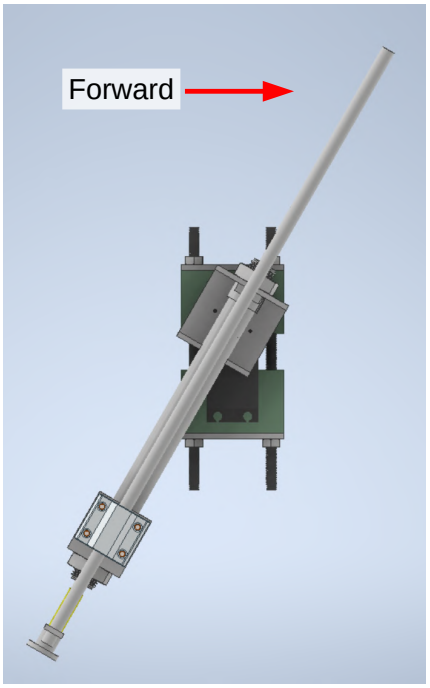



**Figure 5.4:** A figure showing the fixed body robot at the start of the acceleration phase.

The fully stitched trajectory, that was used to compare to results obtained for the fixed body in the following chapters, is shown in Figure 5.5. This trajectory was the 0.15 m/s steady-state stitched, with an acceleration and deceleration phase at the start and end of the steady-state respectively. The trajectory was periodic, meaning that after an acceleration trajectory numerous amounts of steady-state trajectories could be added to increase the distance travelled stitched by a deceleration trajectory at the end to stop at

rest. For the fixed body model the trajectories were generated using a halfway periodicity, as this allowed for half of a steady-state trajectory to be stitched to a full steady-state trajectory, allowing the robot to cover more ground. A video of the optimiser executing the fixed body trajectory can be found at the following link: `https://youtu.be/DdovEx_9Ge4`

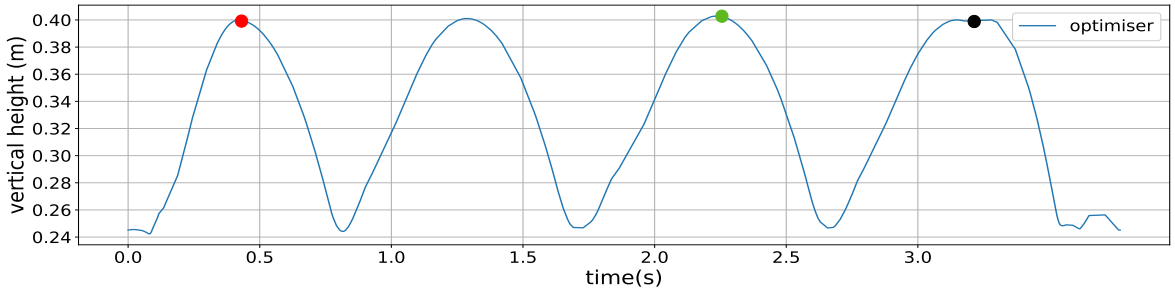



**Figure 5.5:** A figure showing the 0.15 m/s trajectory stitched with an acceleration and deceleration phase. The red dot indicates where the acceleration trajectory was stitched with the steady-state trajectory, the green dot indicates where the steady-state trajectory was stitched with half of the steady-state trajectory as it was found that stitching it with the entire steady-state moved the robot further than the allowed 1 m, and the black dot indicates where the steady-state was stitched with the deceleration trajectory.

## 5.2 Trajectory Analysis of Free Body

The free body robot's body could rotate around the $y$-axis and this had to be updated in the optimiser model. To accomplish this, the manipulator equation setup in Eq. 3.1 was adjusted to include the extra degree of freedom for the rotation of the body, $\theta_{body}$. The addition of a free rotating body caused the body to rotate due to reaction torques from rotating the legs, these internal forces were accounted for when the new equations of motion were set up.

### 5.2.1 Optimiser Setup

The setup of the optimiser remains mostly unchanged, from Section 5.1.1, for the free body. The equations of motions were updated to include $\theta_{body}$ and extra bounds were added to stop the body from rotating past $-90°$ and $90°$.

### 5.2.2 Optimisation Results

The same method as was used to obtain controllers for the fixed body was applied in order to obtain controllers for the free body. Three steady-state trajectories were generated, and then common trends from the three trajectories were observed. As in the case for the

fixed body, $J_2$ was used to generate the trajectory with the lowest average velocity and $J_1$ was used for the other two.

From the trajectories' data in Figure 5.6 and Table 5.5, it can be noted that the robot made contact first with the right foot and then with the left foot. The robot touched down with a foot in front of the body around 65-90° and rotated it to the lift-off angle around 100-110° before propelling itself into the air. From the graphs, it could be seen that the robot retracted its leg at the apex and took a step when it made contact with the ground before leaping into the air.

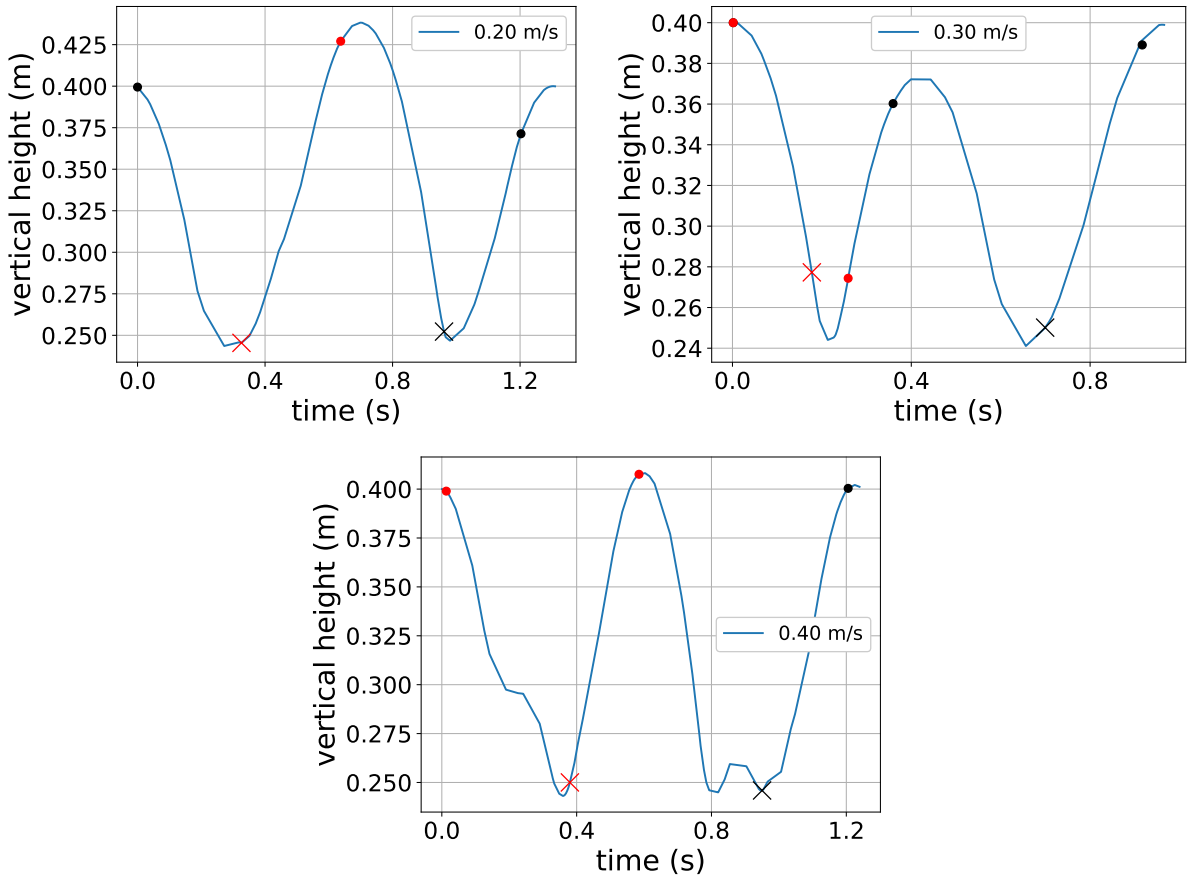



**Figure 5.6:** Figure showing the results obtained from the optimiser for the free body while enforcing three different average velocities. The red and black '×' and 'o' indicates when the prismatic actuator activates and retracts respectively for the right leg and left leg. The maximum velocity was obtained by increasing the average velocity until the optimisation became infeasible.

**Table 5.5:** Analysis of Touchdown and Lift-Off Angles for Free Body.

| Trajectory | Touchdown Angle (°) | | Lift-Off Angle (°) | |
|---|---|---|---|---|
| | Left Leg | Right Leg | Left Leg | Right Leg |
| 0.20 m/s | 67.75 | 93.00 | 100.27 | 106.06 |
| 0.30 m/s | 89.34 | 84.87 | 101.57 | 100.19 |
| 0.40 m/s | 70.52 | 72.25 | 99.30 | 102.47 |

As was done in the case of the fixed body, the neutral point according to Raibert was calculated to observe if the same behaviour emerged in the case of the free body robot. The neutral point analyses according to the Raibert controller (using Eq. 2.3) compared with the optimiser's foot placement can be seen in Table 5.6. The optimiser placed the feet close to the values calculated from the Raibert controller during the steady-state movement and placed the feet in front of the neutral point for deceleration. This indicated that, as was in the case of the fixed body, a form of the Raibert controller emerged in the optimiser for the free body model.

**Table 5.6:** Neutral Point Analysis for the Free Body Robot.

| Trajectory | Contact | Neutral Point (m) | Foot Placement (m) |
|---|---|---|---|
| 0.20 m/s | 1 | 0.050 | 0.059 |
| | 2 | 0.083 | 0.085 |
| | Deceleration | 0.075 | 0.127 |
| 0.30 m/s | 1 | 0.155 | 0.138 |
| | 2 | 0.111 | 0.107 |
| | Deceleration | 0.142 | 0.203 |
| 0.40 m/s | 1 | 0.018 | 0.022 |
| | 2 | 0.100 | 0.100 |
| | Deceleration | 0.046 | 0.093 |

In order to have a full trajectory, acceleration, and deceleration trajectories were generated and stitched to the steady-state trajectory. The optimiser accelerated by starting the left leg at 70°, the right leg at 110° and by firing the left leg's pneumatic cylinder the optimiser propelled the robot into the air. The robot decelerated by landing with the right leg at the touchdown angle in front of the body, before moving into the same orientation the acceleration phase started. Figure 5.7 shows the orientation of the free body robot at the start of the acceleration trajectory.

The stitched trajectory, in Figure 5.8, was used to compare to results obtained for the free body testing in the following chapters. This trajectory was the steady-state obtained for 0.20 m/s with an acceleration and deceleration trajectory stitched to the start and end of the steady-state phase respectively. A video of the optimiser executing the free body trajectory can be found at the following link: `https://youtu.be/rnM-ANJwcfo`

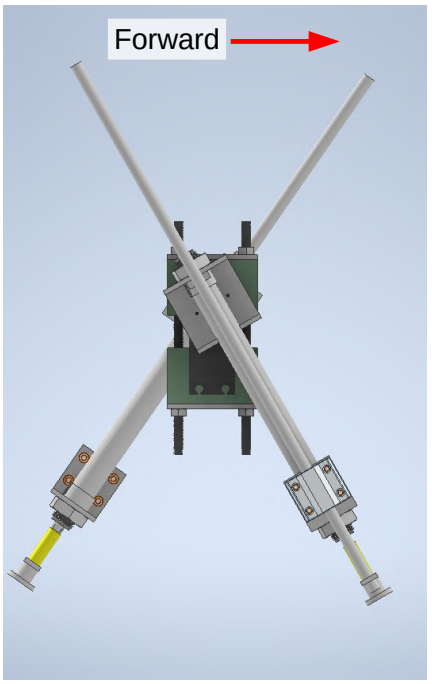



**Figure 5.7:** A figure showing the free body robot at the start of the acceleration phase.

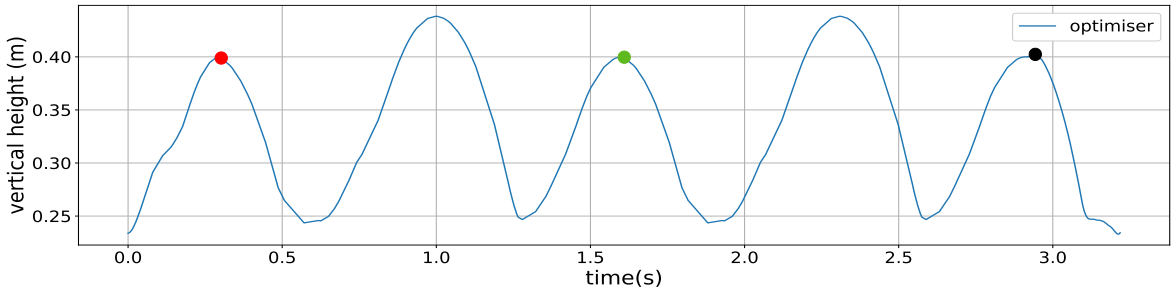



**Figure 5.8:** A figure showing the 0.20 m/s trajectory stitched with an acceleration and deceleration phase. The red dot indicates where the acceleration trajectory was stitched with the steady-state trajectory, the green dot indicates where the steady-state trajectory was stitched with another steady-state trajectory, and the black dot indicates where the steady-state was stitched with the deceleration trajectory.

## 5.3 Pneumatic Control

The classical Raibert controller [9], defines three stages of control:

- Control the hopping height.
- A state machine that tracks the hopping cycle.
- Control the forward speed.

For controlling this robot, the forward speed was determined by the optimiser which choose the best touchdown and lift-off angle to maintain the forward speed of the robot. This was handled by the servo control part of the controller. Controlling the hopping

height and the state machine that tracked the hopping formed part of the pneumatic controller.

The generated trajectories indicated when the pneumatic cylinders had to contract or extend and at which heights this happened. To ensure that the robot reached the correct heights when jumping, the switching delay of the solenoids and the retraction and extension time of the cylinder were accounted for in the optimiser (detailed in Chapter 3).

The pneumatic control, therefore, controlled the extension and retraction of the pneumatic cylinders, allowing the cylinder to fully extend or retract before allowing it to switch to the next state to ensure that the correct heights were achieved. A state machine that tracked the cycle of the states also formed part of the pneumatic control, Figure 5.9 shows the states of the controller.

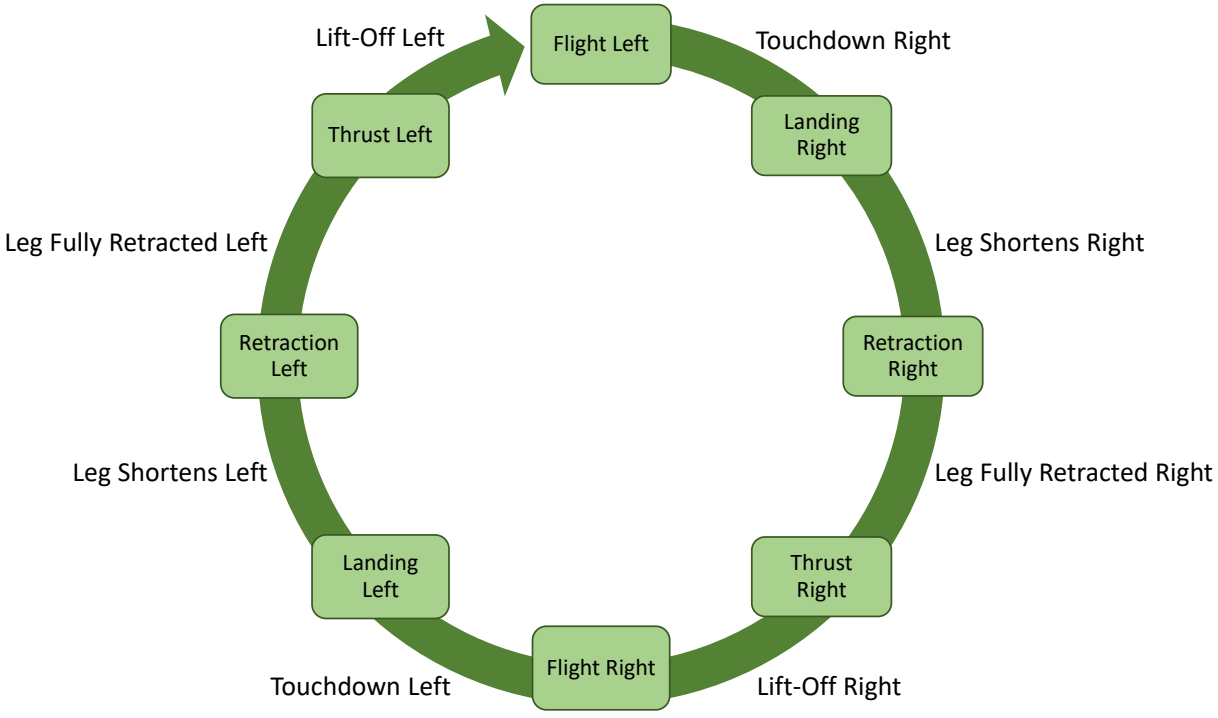



**Figure 5.9:** Figure showing the structure of the state-machine controller.

The control heights used to toggle the states of the robot, as indicated in Figure 5.3 were as follows:

- Fixed Body:
  - Control Height 1 $\geq$ 0.37 m
  - Control Height 2 $\leq$ 0.27 m
- Free Body:

- Control Height 1 $\geq 0.3$ m
- Control Height 2 $< 0.3$ m

In the trajectory optimisation of the fixed body robot, the optimiser retracted its legs at the apex, landed, took a step and then leapt into the air. However, after some testing it was found that the physical robot could not take a step with both legs retracted as both feet were touching the ground, this caused too much friction due to one of the feet dragging across the ground as the robot tried to take a step. To resolve this, the retracting and extension of the legs were adjusted as follows to allow the robot to take a step. Above a height greater than "Control Height 1" the robot would retract the leg it was not landing on while it kept the other leg extended. At a height below "Control Height 2", the robot landed on the extended leg, moved it back to the take-off angle, retracted it, and then propelled itself into the air.

For the free body, both control heights were selected above and below 0.3 m respectively. This was done to allow enough time for the robot to fully extend its piston as well as retract it before alternating legs in the air, as it was found that switching legs while the pistons were still extended caused reaction torque on the body due to the large inertia of the leg, causing it to topple to one side. "Control Height 2" was below 0.3 m as it allowed the pneumatic cylinders to extend, which enabled the robot to leap as soon as it touched down with the ground.

## 5.4 Servo Control

As mentioned, the angles of the legs were studied at lift-off and touchdown, and this was controlled by two servo motors attached to the hips of the robot. Thus, this controller specified the angle of the legs when the robot touched down with the ground, as well as the angle at which it propelled itself into the air. Both of these events affected the robot's speed and acceleration, therefore it was important to ensure that the angles were correct on touchdown and launch.

### 5.4.1 Fixed Body

The fixed body was attached to the boom arm, this caused the angle of the body to rotate with the arm as the robot leapt into the air. This change in angle had to be accounted for in the controller to ensure that the legs moved to the correct angles. The control for this looked as follows:

$$\theta_{leg}(t) = \theta_{boom\ arm}(t) + \theta_{desired} \tag{5.1}$$

where $t$ indicated the current time of the robot. The touchdown and lift-off angles that were used for the controller as obtained from the optimiser were as follows:

- Touchdown Angle = 90°
- Lift-Off Angle = 120°

For the fixed body robot, the controller moved the leg to the touchdown angle while the robot was in flight. When the robot touched down, it took a step with the same leg it landed on, moving the leg to the lift-off angle before leaping into the air.

### 5.4.2 Free Body

Since this robot model had a free-rotating body, any disturbance felt in the system would cause the angle of the legs to deviate from the desired position. To compensate for this change in body angle, the relative leg angle was adjusted according to the body angle and boom angle. The following equation implemented the free body servo control:

$$\theta_{leg}(t) = \theta_{body}(t) + \theta_{boom\ arm}(t) + \theta_{desired} \tag{5.2}$$

This ensured that the legs remained at the required position while adjusting to the movement of the body. The touchdown and lift-off angles used for the free body robot were as follows:

- Touchdown Angle = 70°
- Lift-Off Angle = 105°

As discussed above, the free body robot landed with a leg at around 70° before swinging it back towards 105° and propelling itself into the air. After some testing, it was found that this caused the robot to topple forward as soon as it swung the leg back. The controller was thus modified to prevent the robot from falling forward by placing a leg in front of the body at the touchdown angle, while a leg was placed behind the body to propel the robot forward. This controller kept the robot's legs in the shape of an inverted 'V' to ensure that the moments of inertia around the pivot point were cancelled out by the position of the opposing legs. The leg at the touchdown angle stopped the robot from falling forward when hitting the ground, while the back leg at the lift-off angle was used to propel the robot into the air.

# Chapter 6

# Gazebo Simulation

As mentioned in Chapter 2, Gazebo is a prepackaged application available with ROS and allows for the simulation of models in real-time. Gazebo, together with ROS was used to simulate the designed robot models and controllers before applying it on the physical robot. This was done as an intermediate step to verificate the trajectory optimisation-inspired controllers on a simulated model before implementing them on the physical robot.

## 6.1 Simulation Setup

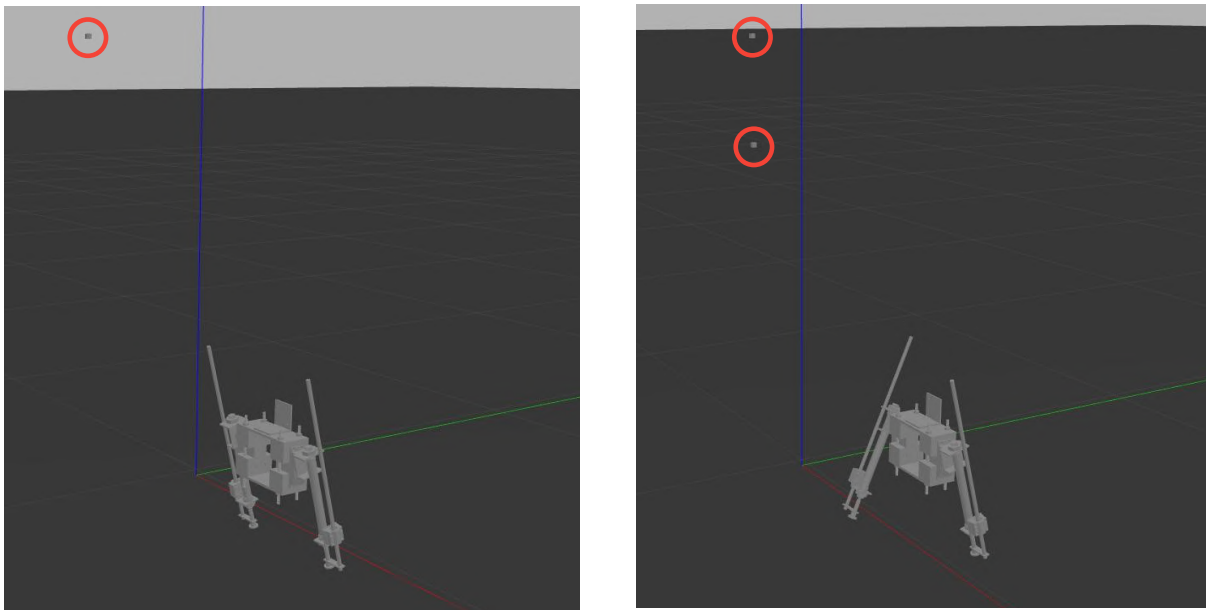
In order to create the simulation, the model created in Inventor (the robot in Figure 4.3) had to be imported into Fusion 360, this is a 3D design application under the Autodesk package. The joints for the robot were reapplied with the correct ranges, thereafter a plugin [75] for Fusion 360 was used to extract the model to a Universal Robot Description Format (URDF), which is an XML file that Gazebo requires in order to describe the model being imported. There were four separate URDF files created by the exporter, they contained the following information about the model:

- The links of the robot and their origins within Gazebo, as well as the joints between the links.
- The physics used within Gazebo, including the friction between the links and the ground.
- The joints that had controllers working on them alongside the description of the type of controller being used. All the controllers used on the joint were set as effort controllers.
- The material that was used to create the body as well as the colour.

Changes were made to the files created by the exporter plugin to ensure a successful simulation. The friction between the feet and the ground was increased to match the

friction coefficient of 0.75 which was used in the optimisation. The servo motors were implemented as revolute joints, and the pneumatic cylinders were implemented as prismatic joints. The limits on the joints were adjusted to match the bounds of the optimisation and the actual robot. Controllers were applied to the required joints through the ROS control plugin, and the output force and torque from the actuator were set for each joint to match the values of the actuators used in the trajectory optimisation. Additional links were created to constrain the robot to the sagittal plane. This was achieved by creating a small block with a low mass which was constrained to the world using a prismatic joint that only moved in the $x$-direction. The robot was then joined to the small block using a prismatic joint, allowing it to move up and down the $z$-axis.

For the free body model, an extra block was created, which was connected to move up and down the previously created prismatic joint, the robot was constrained to rotate along this block using a revolute joint. These additional links and joints allowed the robot to operate as was intended when the platform was designed. Figure 6.1, shows the robot in the simulation environment for both the fixed body and free body, respectively.



**(a)** Fixed Body. **(b)** Free Body.

**Figure 6.1:** A Figure showing the robot within the Gazebo environment. If observed closely in the red circles, the links could be noticed that were used to constrain the model to the sagittal plane.

In Chapter 2 it was mentioned that once a ROS simulation was launched, it spawned several ROS topics and ROS services, the important ROS topics and services were listed and detailed. In order to successfully control the robot, the height of the robot was required along with a way to actuate the required joints, the aforementioned ROS services and topics in Chapter 2 were used to achieve this.

To control the simulation, the controllers designed in Chapter 5 had to publish over the ROS network established in Chapter 4. The Python script published the actuators' required positions to the four ROS topics responsible for actuating the robot and a ROS service call was made to obtain the vertical height of the robot, which allowed the states to transition from one state to the next. Figure 6.2 shows a graphical representation of the ROS services and topics used to control the robot within the simulation. The ROS services returned the height of the robot, while the ROS topic moved the actuators to the required positions. In the case of the pneumatic cylinder, the position was moved between the maximum and minimum value at a high force and speed to mimic the bang-bang nature thereof.

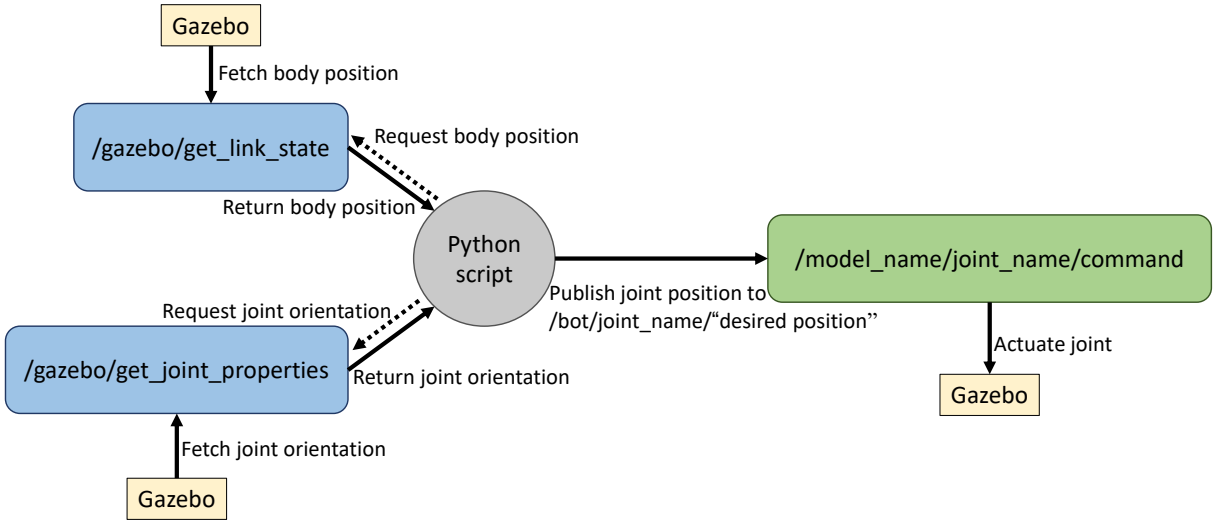



**Figure 6.2:** A diagram showing the interaction between the Python script running the control loop and the ROS topics and services required to control the robot. The blue boxes indicate ROS services, while the green box indicates a ROS topic.

## 6.2 Gazebo Results

The results obtained in this section aimed to verificate the controller design in simulation before moving on and implementing it on a physical system. The simulation was done for both the fixed body and free body models.

### 6.2.1 Fixed Body

After creating the controllers as designed in Chapter 5, it was implemented in the simulation environment to test the state machine design from the trajectory optimisation results. This was done as a preliminary step to see if any adjustments had to be made before moving on to the practical system. Figure 6.3 shows the results from the Gazebo simulation environment for the long-time-horizon task.

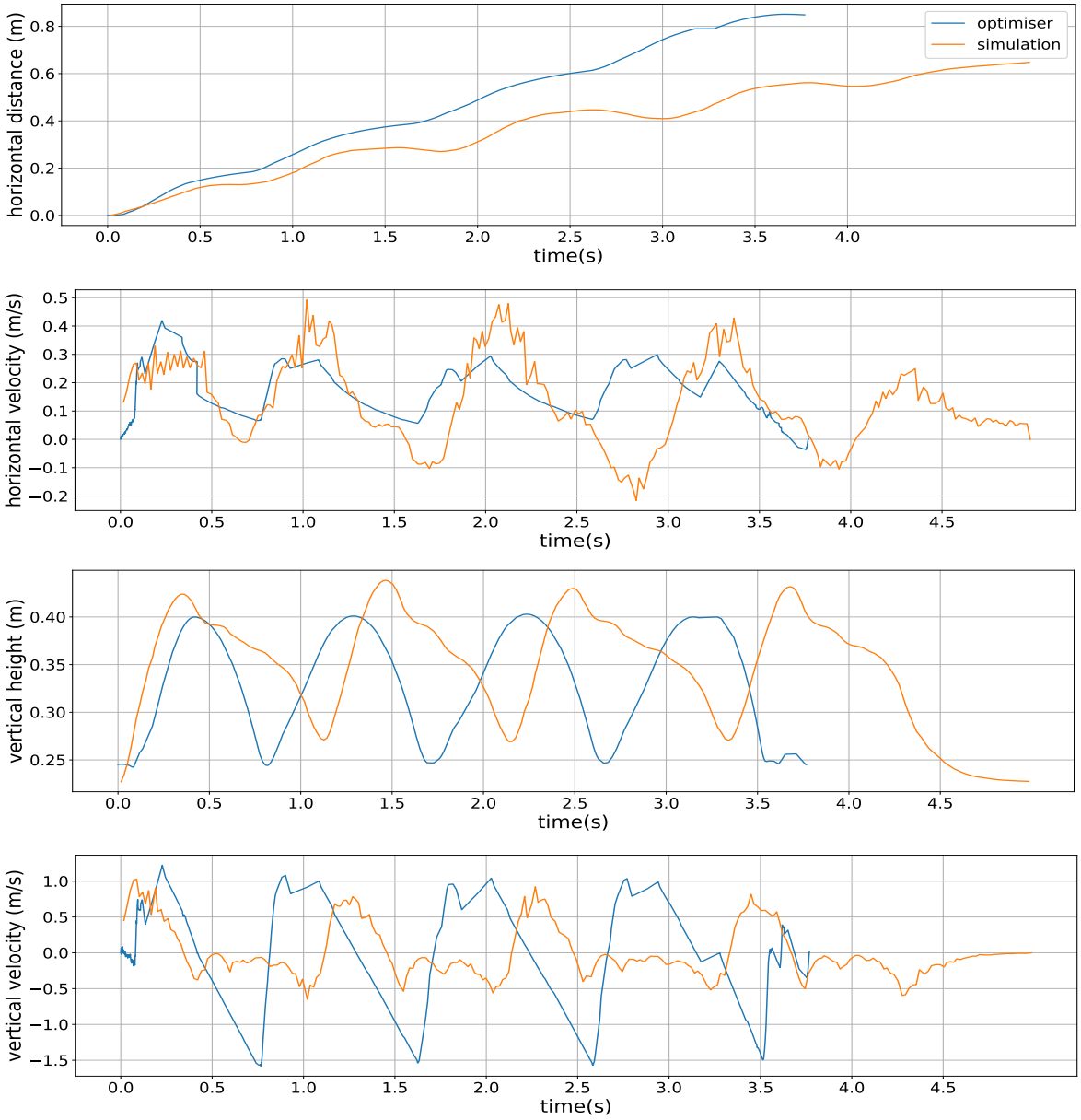



**Figure 6.3:** A figure comparing the results obtained from the trajectory optimisation to the result from the Gazebo simulation for the fixed body. The graphs show the horizontal distance and velocity as well as the vertical height and velocity.

From Figure 6.3, it can be noted that the optimisation achieved a slightly greater final distance compared to the simulation data (a 0.2 m difference in the final distance), while the simulation also took 1.2 sec longer to achieve this. This is due to the Gazebo simulation having a real time factor which was influenced by the hardware the simulation was implemented on. Since the hardware was not as powerful, it had a real-time factor of around 0.9, which indicated that the simulation was running at 90% of the real-time speed. Another reason for discrepancies between the trajectory optimisation and the simulation results was due to inaccuracy in the trajectory optimisation model. Studying the vertical height of the simulation compared to the optimiser, it could be seen that the simulation's jump pattern had roughly the same shape as the optimiser, with the simulation

achieving slightly higher jump peaks, around 0.45 m.

From the graphs indicating the velocities, it was evident that the horizontal velocity of the simulation had a large swing into the negative area and that the vertical velocity did not achieve the same speed downwards as the optimisation. This was due to links constraining the model to the sagittal plane having their own dynamics that influenced the system, even though the damping and the inertia of the links were set as low as possible it still affected the system. The optimiser had an average steady-state horizontal velocity of 0.150 m/s while the simulation achieved an average velocity of 0.148 m/s. The peak vertical velocities of the optimiser as well as the simulation were around 1.0 m/s. A video showing the fixed body simulation can be found at the following links:

- Normal speed: `https://youtu.be/ds1wefx-ZJ8`
- Slowed down: `https://youtu.be/McuRiSW0kWU`

The fixed body testing done in Gazebo proved that the designed controllers could control the more stable fixed body robot within the simulation environment. Further testing had to be done on the more unstable free body robot to investigate if it reached the same result.

### 6.2.2 Free Body

After the testing on the fixed body model was completed, the Gazebo model was updated to include the extra degree of freedom by adding the aforementioned additional link and joint into the Gazebo simulation model. The data obtained from the free body testing compared to the results obtained from the trajectory optimisation can be seen in Figure 6.4.

From the results obtained in Figure 6.4, shows that the robot in the simulation achieved a slightly further distance than the one obtained from trajectory optimisation, moving 0.13 m further. The simulation took 0.6 sec longer than the optimisation to reach the final distance while still running at 90% of real-time speed. The robot managed to achieve a faster average horizontal velocity in the Gazebo simulation (0.30 m/s) compared to the 0.20 m/s average horizontal velocity of trajectory optimisation. This could be due to the actuators in the simulation being modelled differently from the ones in the optimisation while set to exactly the same values. The robot in simulation required four contacts in total to reach the end goal, the same as what was observed in the optimisation and the jumping patterns loosely matched the pattern obtained from the trajectory optimisation, achieving a maximum vertical height of 0.38 m compared to the average maximum height of 0.45 m reached by the optimiser.

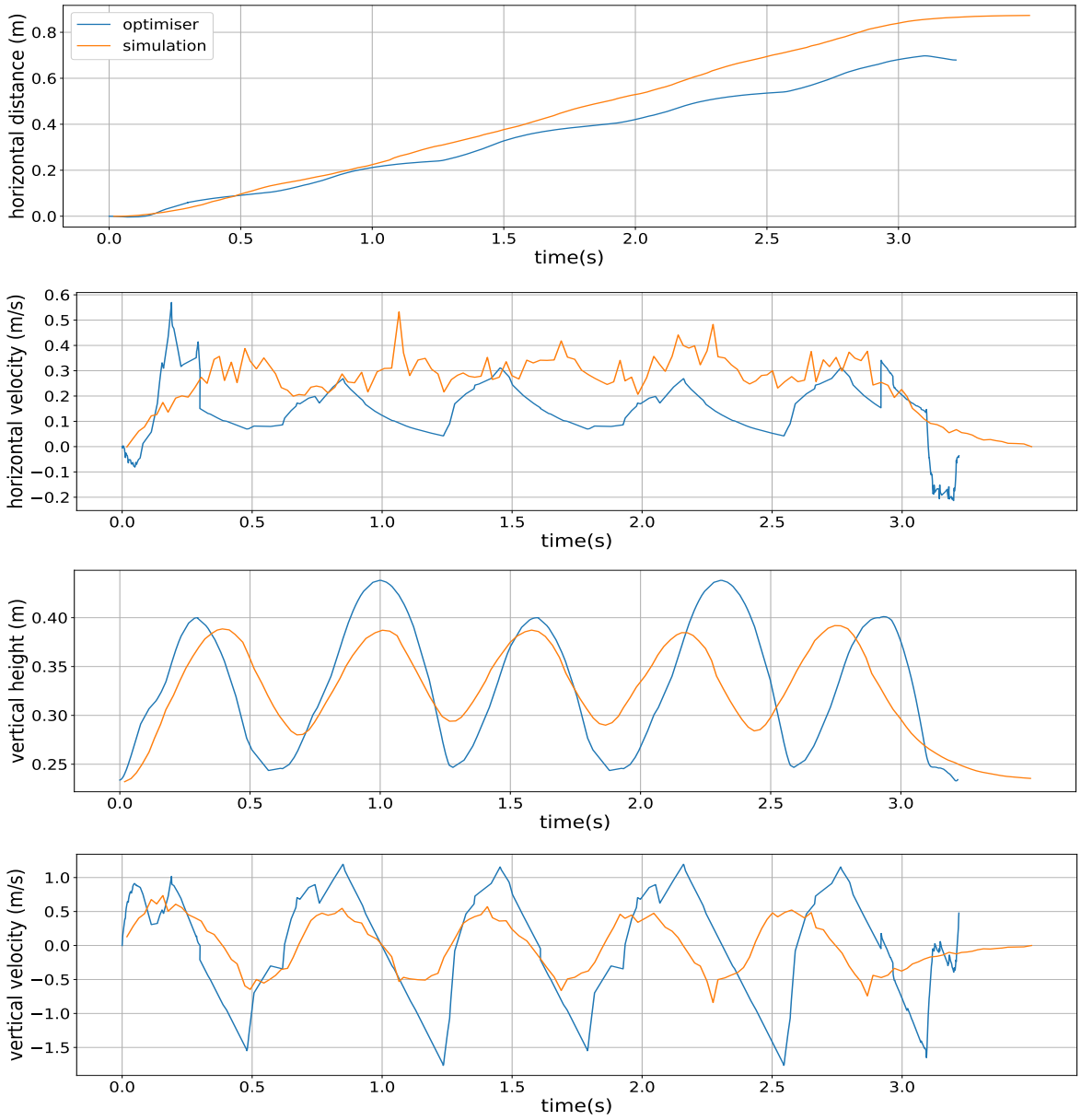



**Figure 6.4:** A figure comparing the results obtained from the trajectory optimisation to the result from the Gazebo simulation for the free body. The graphs show the horizontal distance and velocity as well as the vertical height and velocity.

Analysing the vertical velocity, it was observed that the simulation's peak vertical velocity was under the 1 m/s the trajectory optimisation achieved. The reason for the difference in the minimum vertical velocity was previously discussed in the fixed body section of this chapter, and the same damping and inertia of the links had occurred for the free body testing. Figure 6.5 indicates the body angle between the two tests, it can be seen that the optimiser's body angle rotated between -2° and 2° while the simulation had little to no rotation. This could be due to the inertia of the robot being simulated being more than the inertia of the optimised model, as well as the joint responsible for the body rotation having inertia and damping of its own. The free body simulation video can be found at the following links:

- Normal speed: `https://youtu.be/QLcoFe5JPeM`
- Slowed down: `https://youtu.be/sxFQbwEcX8U`

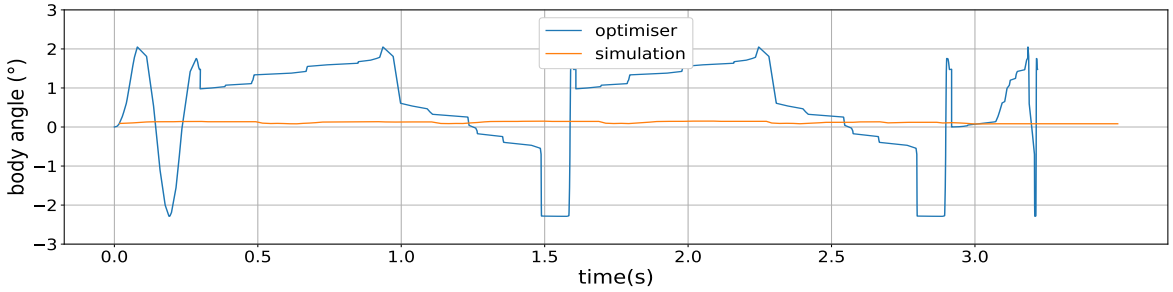



**Figure 6.5:** A figure showing the body angle obtained from the free body testing compared to the simulation.

The results from the testing done within the Gazebo simulation indicated that the controllers worked in a simulation scenario and that it could be applied to a more realistic robot model than the model represented in the trajectory optimisation and still achieve acceptable results. A step up from the simulation testing was applying these controllers to a real-world system and analysing the data to see if it led to satisfactory results.

# Chapter 7

# Testing and Results

The viability of the trajectory optimisation-inspired controllers had to be tested and validated to see if the assumptions and reduced models used within the optimisation could lead to successful controller design. The testing was first done on an ideal rigid surface to validate that the controllers could successfully control the robot. Thereafter, a "rough" surface was created by placing wooden blocks in the path of the robot to investigate if the designed controller could overcome these obstacles or how it would hinder the resulting motion.

## 7.1 Fixed Body

As a preliminary step, the fixed body robot was used for validation before moving on to the free body robot. The fixed body had improved stability resulting in simplified trajectories, and controller, for the robot. This section validated the trajectory optimisation and controller design as mentioned in Chapter 3 for the fixed body robot. The trajectories were executed on the robot to prove that these trajectories could lead to controllers for the robotic platform.

### 7.1.1 Rigid Surface

Figure 7.1 shows the data obtained from the fixed body testing of the robot on a rigid surface and compared it to the data obtained from the trajectory optimisation. From the figure, it can be seen that the robot achieved a final distance of 0.75 m compared to the optimiser's final distance of 0.81 m. From the graph indicating the horizontal velocities, it can be noted that the physical robot had a negative horizontal velocity even though the robot did not move backwards, this is due to the belt on the encoder shaking under the impact of the robot's feet with the ground. The robot managed to achieve an average forward velocity of 0.18 m/s for the rigid surface, compared to the 0.15 m/s average horizontal velocity set within the optimiser.

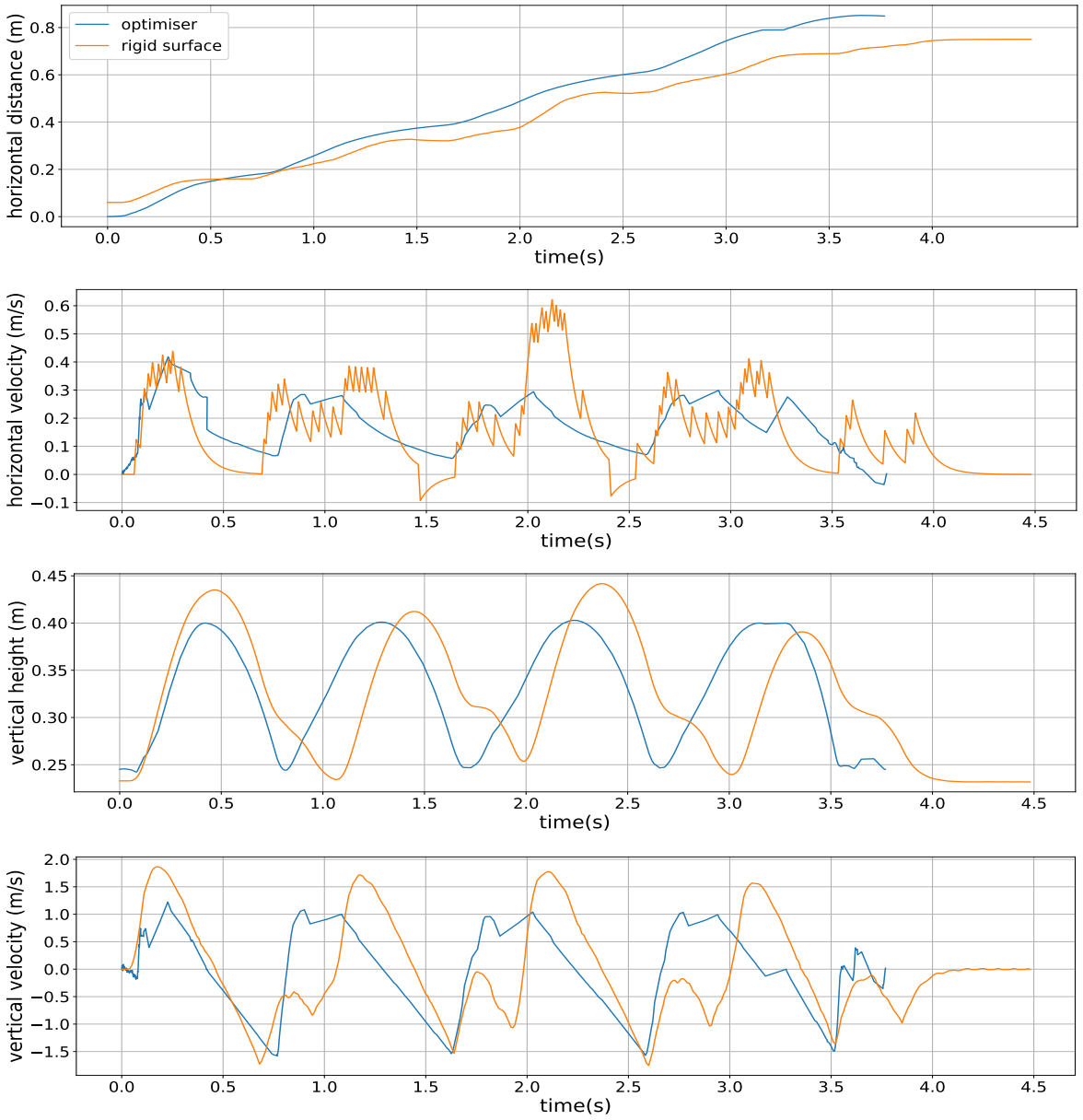



**Figure 7.1:** Figure showing data obtained from testing of the fixed body robot on a rigid surface. The graphs show horizontal distance and velocity as well as vertical height and velocity.

The physical robot achieved a slightly higher maximum jump height (around 0.43 m) compared to the optimiser which constantly reached 0.4 m. This is due to the fact that the robot was attached to a boom arm, which led it to jump more in an arc than straight upwards and caused a higher recorded peak jump height. The fixed body robot required three steps to reach the end of the track, as was the case for the optimisation. The vertical velocities of the physical robot matched closely to the one generated by the optimiser while the physical system's maximum vertical velocity was around 1.6 m/s compared to the optimiser's maximum vertical velocity of 1.0 m/s. This was due to the higher maximum jump height of the physical system, which had an impact on the velocity that was achieved. A video of the fixed body testing on the rigid surface can be found at the following links:

- Normal speed: `https://youtu.be/XFybz4wtVx0`
- Slowed down: `https://youtu.be/H3pQtXqXD-Y`

With the testing on the rigid surface being concluded with acceptable results. The test surface was changed to accommodate for rough surface testing.

### 7.1.2 Rough Surface

After the rigid testing was done, wooden blocks were placed in the path of the robot to act as a disturbance, as can be seen in Figure 7.2. These blocks acted as a hindrance in the path of the robot, and the effect thereof on the system was investigated.

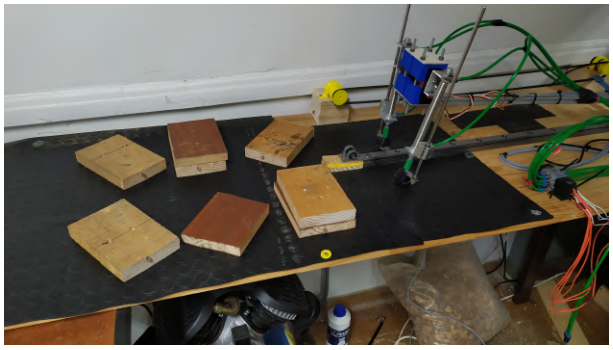


**Figure 7.2:** Figure showing the wooden blocks placed in the path of the fixed body robot.

Figure 7.3 shows the data obtained from testing on the rough surface compared to the testing done on the rigid surface. From the graphs, it can be seen that the robot managed to achieve roughly the same final distance on the rigid and rough surface, which was 0.75 m. The robot also had a slightly higher average horizontal velocity on the rough surface (0.20 m/s) compared to the rigid surface (0.18 m/s). The robot reached the end of the track in a faster time on the rough surface compared to the rigid surface, this could be due to several reasons such as the compressor actuating the cylinders being charged to the maximum pressure providing more force to them, as well as the servo motors being replaced for new ones in between the different tests, resulting in a better performance. Studying the vertical height of the robot, it can be seen that the shape of the movement was very similar, with the rigid surface achieving a maximum higher jump height of around 0.43 m compared to the maximum height on the rough surface being around 0.39 m. This was due to the feet slightly slipping over the wooden blocks, causing the robot not to propel itself into the air properly. The vertical velocities were nearly identical, with the one on the rough surface being completed slightly faster for the reasons discussed. The video of the rough surface test can be found at the following links:

- Normal speed: `https://youtu.be/SM5f6oZdQ-A`

- Slowed down: `https://youtu.be/V4aGQa-qa7M`

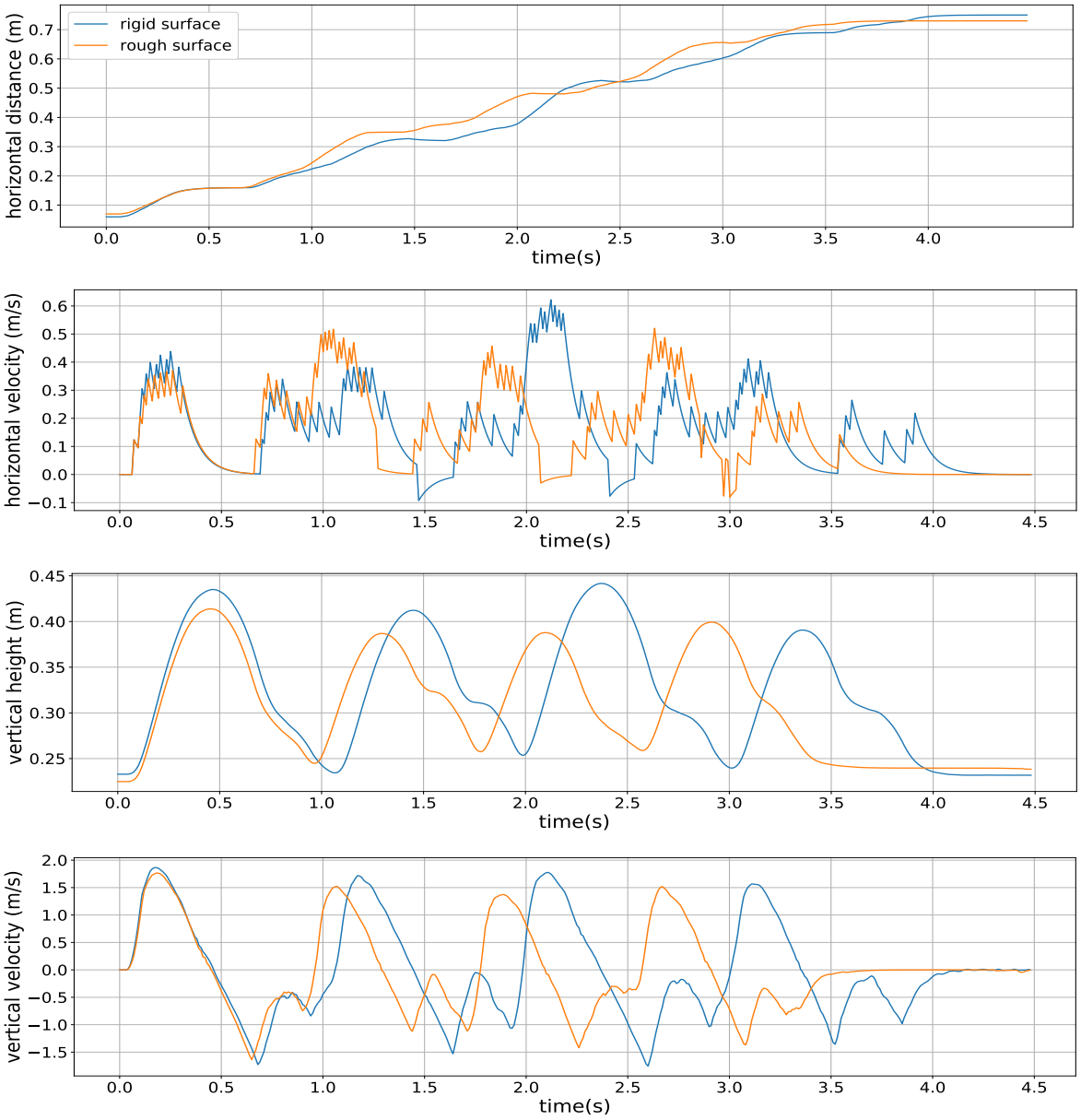
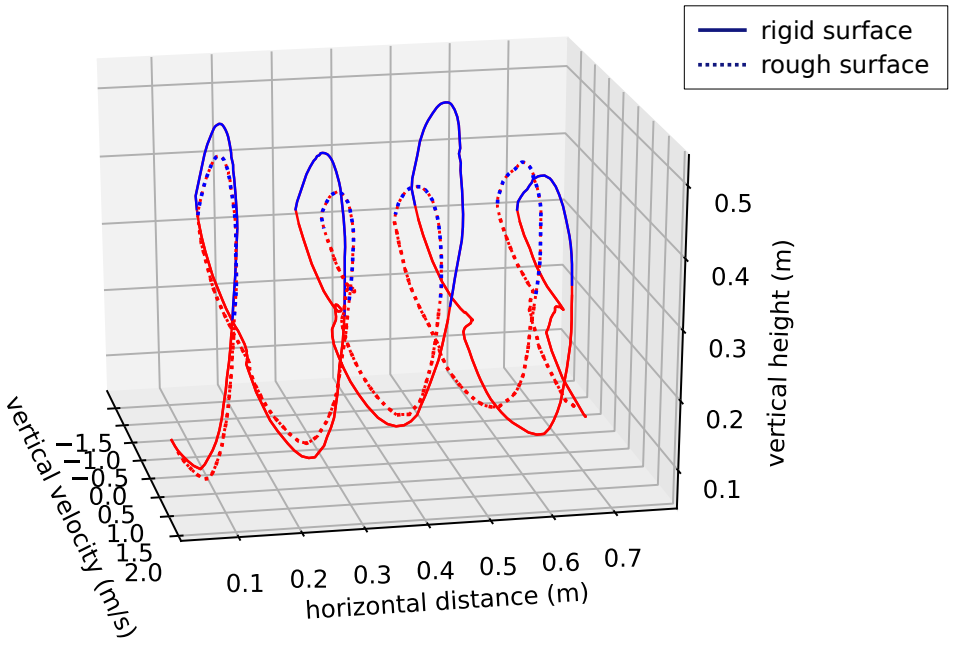
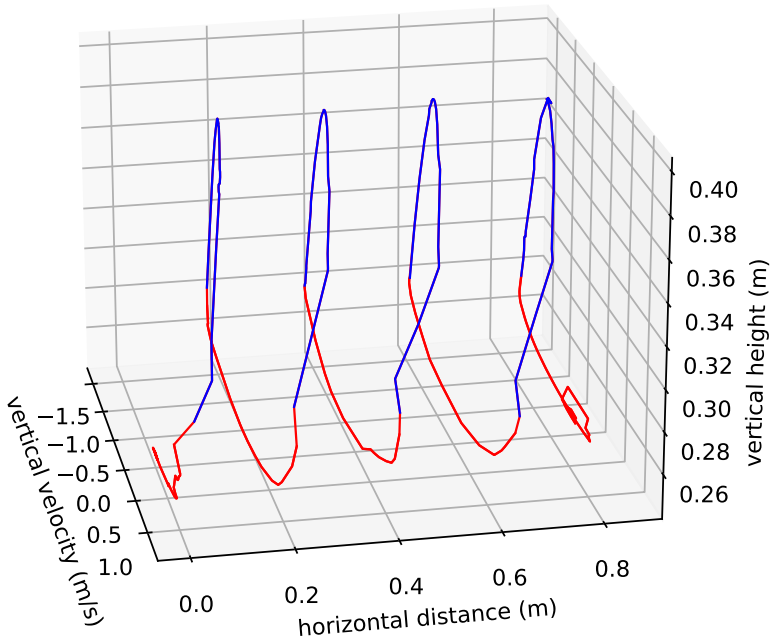



**Figure 7.3:** Figure showing data obtained from testing of the fixed body robot on a rigid surface compared to the rough surface. The graphs show horizontal distance and velocity as well as vertical height and velocity.

Figure 7.4, indicates the limit cycle of the physical system and the optimiser. From the figure, it can be noted that even with the reduced order model used in the trajectory optimisation, along with the non-linearities of the system and the disturbance from the uneven terrain, the robot was able still to achieve a limit cycle until the end of the trajectory.




**(a)** Physical System.




**(b)** Optimiser.

**Figure 7.4:** Graph showing the result of the limit cycle of the fixed body robot along the limit cycle obtained from the model used in the trajectory optimisation, the red line indicates when the robot was on the ground and blue when it was in the air. The solid line indicated the data for the robot on a rigid surface, and the dotted line indicates the rough surface data.

These results indicated that the controller designed for the fixed body robot satisfied the long-time-horizon task being tested and allowed the robot to reach the end of the trajectory even with disturbances added through the rough surface. The controller could thus be implemented on the free body robot to investigate if it reached the same results.

## 7.2 Free Body

The modified controller for the free body system could now be tested on the physical platform. The extra degree of freedom in the equations of motion of the free body, which allowed it to rotate along the boom arm, increased the complexity of the system, especially when it came to the rough surface. The disturbances introduced through the uneven terrain would have a large impact on the angle of the body of the robot. The controllers implemented proved that it was capable of handling these hindrances.

### 7.2.1 Rigid Surface

Figure 7.5 shows the data obtained from the free body testing of the robot compared to the data obtained from the trajectory optimisation. From Figure 7.5, it can be seen that the free body robot reached a final distance of 0.61 m compared to a final distance of 0.81 m achieved by the optimiser. It required four steps for the free body, physical robot and optimisation, to reach the end of the track, compared to the three required for the fixed body. The physical robot achieved an average horizontal velocity of 0.18 m/s compared to the average steady-state velocity of 0.20 m/s achieved by the optimiser. The reason for the discrepancies in data was due to the legs of the robot not being fully utilized, as taking larger steps or moving the legs further back during take-off would result in the robot travelling further and achieving a higher average velocity. However, this would lead to the system becoming more unstable, as such the focus was more is keeping the body stable making the system more reliable.

The physical robot achieved an average maximum jump height of 0.35 m on the rigid surface and 0.41 m in the trajectory optimisation. The vertical height obtained from the free body robot, compared to the fixed body was more varied, this is due to the body angle changing with each consecutive jump. The robot achieved a vertical velocity similar to the one observed by the optimiser with a maximum velocity of around 1.2 m/s. The free body testing can be seen at the following links:

- Normal speed: `https://youtu.be/AQejzh7CHKM`
- Slowed down: `https://youtu.be/y0ETCdtMKiY`

These tests were deemed valid for the rigid surface testing. The surface was once again changed to test the effects the rough terrain would have on the rotating body.

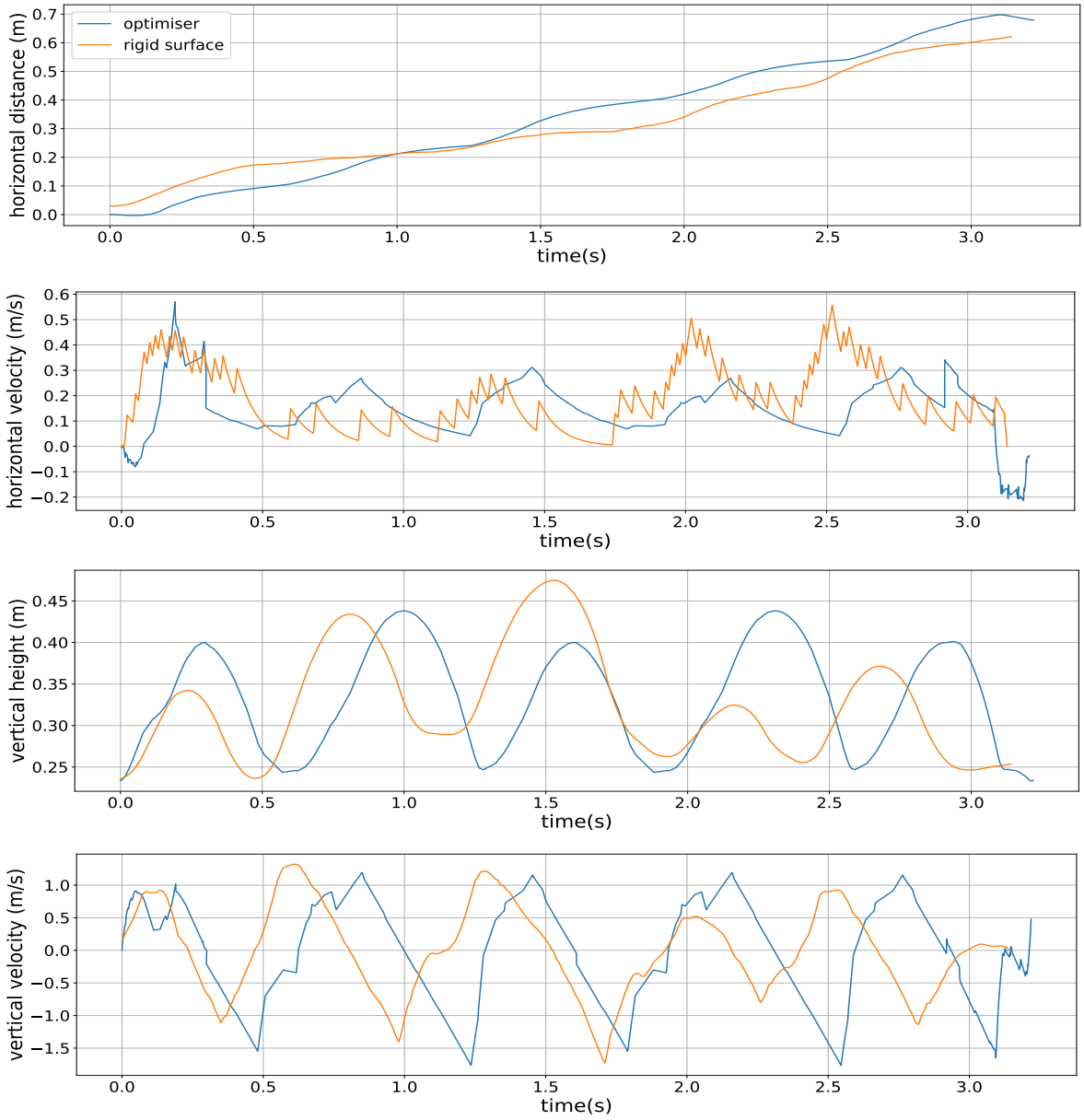



**Figure 7.5:** Figure showing the data obtained from testing of the free body robot on a rigid surface. The graphs show horizontal distance and velocity as well as vertical height and velocity and compared it to data obtained from the trajectory optimisation.

### 7.2.2 Rough Surface

As in the case of the fixed body robot, the wooden blocks were placed to investigate the effects of the disturbances on the system. The results obtained from the free body testing on the rough surface can be seen in Figure 7.5, these results were compared to the data obtained from the free body rigid surface testing.

From the graphs, it can be noted that the robot on the rigid surface was able to achieve a slightly greater final distance compared to the rough terrain robot (0.61 m for the rigid surface and 0.50 m for the rough surface). The free body achieved an average horizontal velocity of 0.18 m/s on the rigid surface and an average velocity of 0.15 m/s on the rough

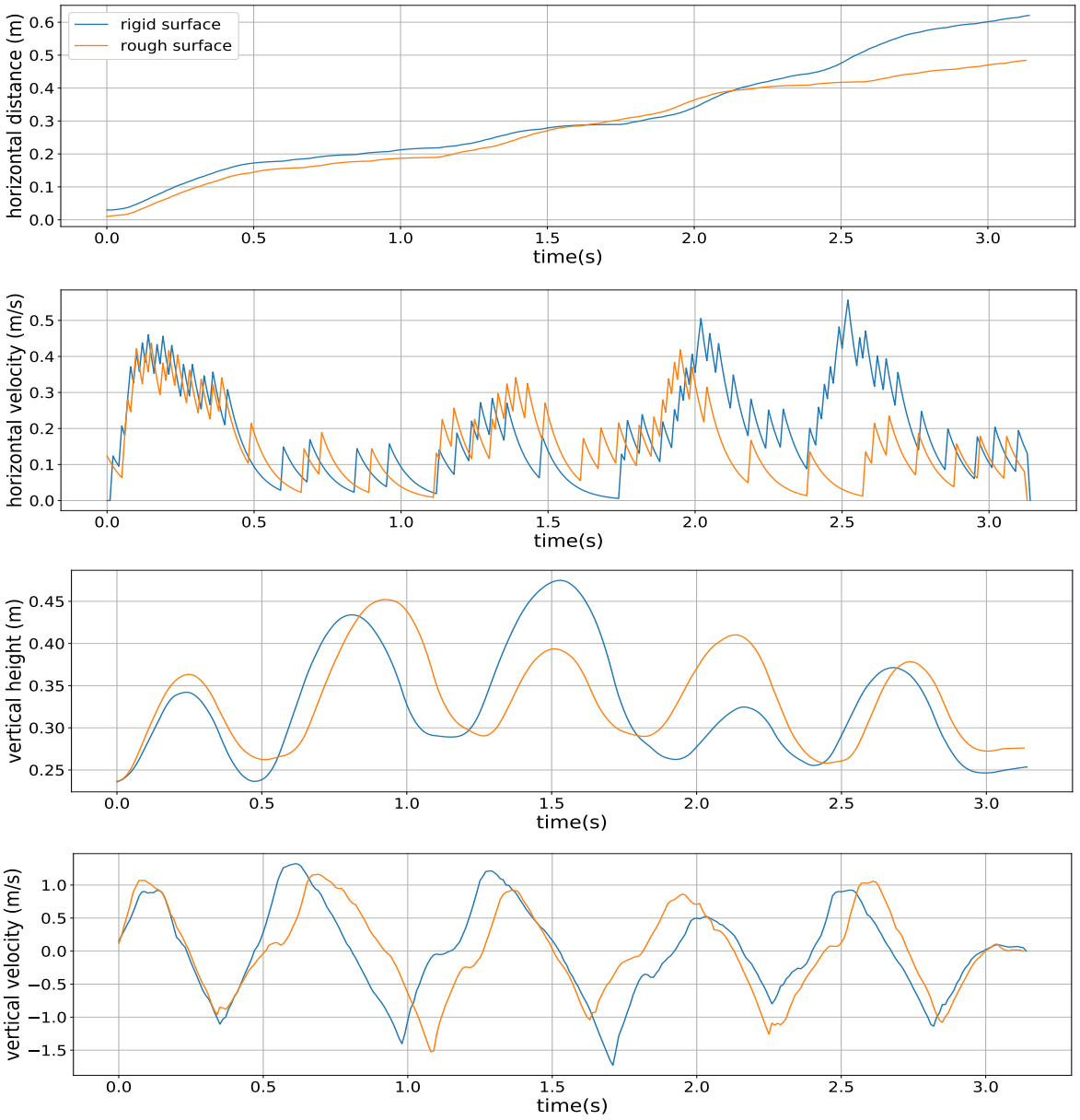



**Figure 7.6:** Figure showing the data obtained from testing of the fixed body robot on a rigid and rough surface. The graphs show horizontal distance and velocity as well as vertical height and velocity.

terrain. The vertical height graphs had nearly an identical shape, with the robot on the rigid surface achieving a higher maximum height. It can be noted that the average height of the robot on the rough surface was higher compared to that on the rigid surface. This was due to the block lifting the robot slightly higher off the ground before leaping into the air, causing it to jump from a lifted platform. The vertical velocities between the two tests were nearly identical, with the robot on the rigid surface reaching a maximum vertical velocity of around 1.2 m/s compared to the value of 1.1 m/s achieved on the rough surface.

Figure 7.7 indicates the body angle of the robot in the two tests. It can be noted that the wooden blocks caused the free body to rotate more on the rough surface test compared

to the rigid surface. On the rough surface, the body moved to a minimum and maximum angle of 70° and 182° compared to a minimum and maximum value of 78° and 163° on the rigid surface. A video of the free body testing on the rough surface can be found at the following links:

- Normal speed: `https://youtu.be/QfocvOveJKc`
- Slowed down: `https://youtu.be/mtl_3RXq3YA`

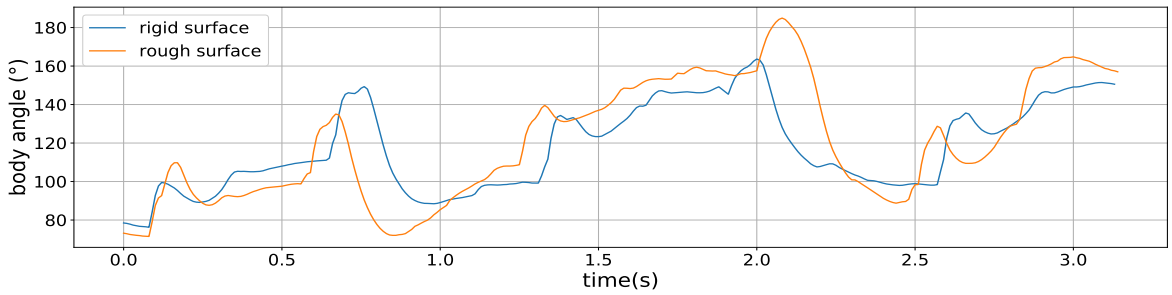
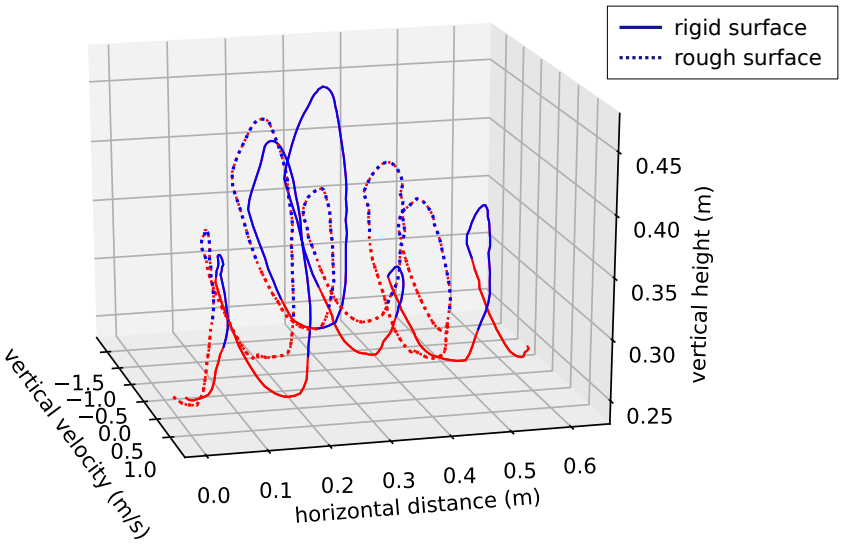


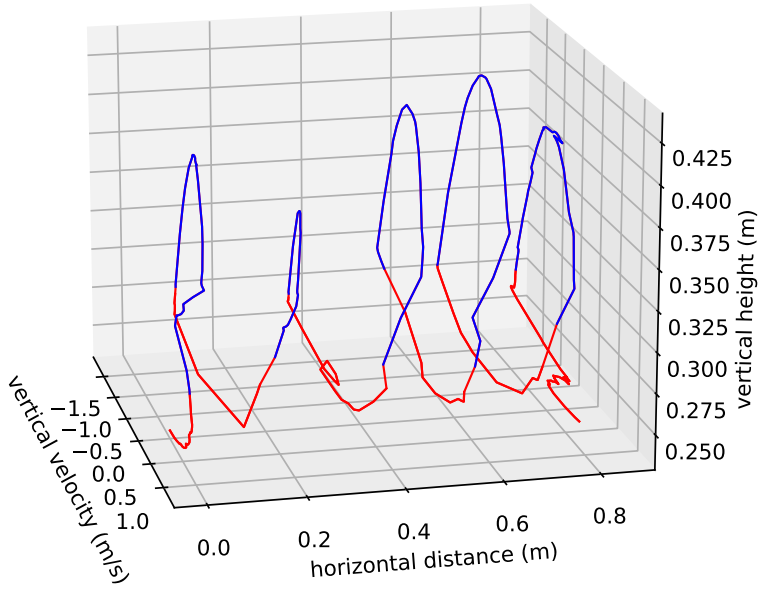

**Figure 7.7:** Figure showing the body angle of the free body robot for both the rigid and rough surfaces.

Figure 7.8 indicates the limit cycle of the free body robot on both surfaces, as well as the limit cycle achieved by the optimiser. The limit cycle indicated that the robot reached a point of stability along its path, and it can be seen that even with the added disturbance, the robot still managed to achieve a limit cycle.

The results obtained in this section indicated that the controller designed for the free body robot was successful in controlling the platform, even with the increased difficulty of the extra degree of freedom introduced into the system. Even though the disturbance introduced by the wooden blocks had an impact on the robot's body causing it to rotate further, it could still complete the long-time-horizon task, by achieving a lesser final distance.




**(a)** Physical System.




**(b)** Optimiser.

**Figure 7.8:** Graph showing the result of the limit cycle of the free body robot along the limit cycle obtained from the model used in the trajectory optimisation, the red line indicates when the robot was on the ground and blue when it was in the air. The solid line indicated the data for the robot on a rigid surface, and the dotted line indicates the rough surface data.

# Chapter 8

# Conclusion

Legged robotics is a continuously growing field and even with outstanding platforms like the Boston Dynamics Atlas [5] and Honda's Asimo [4], legged robots are still far away from becoming a larger part of everyday life. Even with the astonishing advances in legged robotics, many research groups struggle to create robotic platforms that are capable of robust movement outside the laboratory environment. For a robot to successfully perform outside of the laboratory environment, it needs to be capable of robust movement.

The goal of this research was to use trajectory optimisation to inspire controller design for a low-cost pneumatic electrical bipedal robot capable of robust and explosive movement. To achieve this, trajectories had to be generated that matched the robot that was designed, therefore a mathematical model of the robot was created and placed within the optimiser to be solved. The optimiser was left to handle contacts using through-contact methods, which were enforced by using complementarity constraints. This allowed the optimiser to choose its own contact order so that a natural gait pattern for the system could emerge. Two different robot models were used in this project, the one had a fixed body and the other had a free body allowing its body to freely rotate, increasing the complexity of the system.

A method was needed to confirm whether the trajectories generated were feasible, a low-cost bipedal robot was designed and built which consisted of two servo motors at the hips and pneumatic cylinders acting as legs. The robot was constrained to the sagittal plane through the use of a boom arm attached to a support rig. The support rig was fitted with the necessary power delivery for the actuators and sensors. The sensors captured the appropriate data needed in order to control the robot, as well as compare the results to the generated trajectories. The sensors included optical encoders which were used to capture the height, distance travelled as well as the body angle of the robot. A Teensy 4.0 was used to read the sensor data, whereafter it was transmitted to a Raspberry Pi, which handled the actuator control. ROS was used to send the sensor data to the PC as it was responsible for the control of the robot, the actuator data required was sent back

to the Raspberry Pi using the established ROS connection.

The generated trajectories were analysed to inspire a controller design for the system. For this, common trends between three different trajectories at different average velocities were studied and noted down. For the fixed body, it was found that the robot touched down with a leg, swept that foot back and lifted off with the same leg, as was the case for the free body. The control for the robot was split into two parts, the first part was the pneumatic control which controlled when the cylinders would fire or retract and the second part was the servo control which ensured that the leg reached the correct touchdown and lift-off angles. These controllers also compensated for the rotation of the boom and in the case of the free body, the angle of the body, by adding these angles to the required position of the legs. The controller for the fixed body robot had to be adapted as it was found that with both feet touching the ground the robot could not take a step forward, this was fixed by adjusting when the legs retracted or extended to allow for the robot to take a step without both feet touching the ground. While the controller for the free body had to be adjusted to stop the robot from falling over, this was achieved by placing a leg in front of the body and then using the back leg to propel the robot forward.

To first validate if the controllers that were designed were capable of controlling the robot, a simulation was set up by using Gazebo, a simulation program packed within the ROS software. The robot designed was first imported into Fusion 360, whereafter a plugin was used to export the robot model into a supported file format used by Gazebo. The data obtained from the testing done on the simulated robot model was compared to the generated trajectories, and it was found that the controllers could control the robot in the simulation environment. Further testing was done on the physical system to see if it led to the same results.

The testing on the physical system was first done on the fixed body robot with a rigid surface. After successful testing, wooden blocks were placed in the robot's path to investigate what effects these hindrances would have on the robot's motion. The results of the testing on the rigid surface were compared to the data obtained from the trajectory optimisation, and the test from the rough surface was compared to the rigid surface. It was found that the controller executed the trends observed in the trajectories in a satisfactory way.

After testing on the fixed body robot was concluded, the robotic platform was modified to allow for the robot's body to freely rotate. The tests were then performed on the free body robot and the results were compared between the rigid surface and optimiser and between the rough surface and rigid surface. It was found that even with the addition of a rotating body, the controller was able to handle the rigid surface as well as the disturbances added by the rough surface. The results from the testing confirmed that trajectory optimisation could inspire controllers capable of robust movement even over terrain that was not solved

for by the model. In the end, it was concluded that trajectory optimisation could lead to controller design for a low-cost robotic platform and showed that a robot could be controlled robustly without the need for expensive computing systems.

## 8.1 Future Work

This research had shown that trajectory optimisation could successfully lead to controller design for a bipedal robot. Future work could explore turning with a pneumatic bipedal robot by extending the degrees of freedom of the legs. The versatility of the robot could further be increased by equipping it with onboard power and a compressed gas canister. With the turning and onboard systems, the robot could be taken into a 3-dimensional environment, for this reason, the ROS connection between the system can be switched to communicate over Wi-Fi, allowing the robot to travel further distances. Trajectories could be generated for different rough terrain (such as stairs) extending the capabilities of the robot. Trajectories tend to be stitched together this also allows for the possibility of different trajectories with different movements to be generated and stitched together, allowing for a trajectory library to be created and swapped between as the need arises. The bipedal robot could be extended to a pneumatic quadruped, increasing its stability, and making it easier to extend the robot into a 3-dimensional space.

# Appendix A

# Equations of Motion

Each of the two robots were modelled using rigid links. Each link had a mass, moment of inertia (MoI) and centre of mass (CoM) describing it. The body of the robot was connected to the legs using torque actuators, while the legs were pneumatic actuators modelled as prismatic actuators with additional constraints applied to it. To generate the EoM the generalised coordinates, $\boldsymbol{q}$, were first defined as well as the velocity, $\dot{\boldsymbol{q}}$, and acceleration, $\ddot{\boldsymbol{q}}$.

The position vector of the CoM, $P_{CoM}$, of each rigid link was calculated using the generalised coordinates. The velocity of the CoM was calculated using the Jacobian matrix:

$$\dot{P}_{CoM} = jacobian(P_{CoM}, \boldsymbol{q})\dot{\boldsymbol{q}} \tag{A.1}$$

From the position and velocities calculated, the kinetic, $T$, and potential, $V$, energy were calculated for each link. The potential and kinetic energy were summed to generate the total potential and kinetic energy present in the system.

$$\begin{aligned} T_{linear} &= \frac{1}{2} m \times \dot{P}_{CoM} \times \dot{P}'_{CoM} \\ T_{rotational} &= \frac{1}{2} \omega' \times J_{MoI} \times \omega \\ V &= m \times g \times P_{CoM,z} \end{aligned} \tag{A.2}$$

where $g$ was the gravitational constant, $m$ the mass of the link and $P_{CoM,z}$ the vertical height of the CoM. The angular rate vector was $\omega$ and $J_{MoI}$ the moment of inertia of the link.

The EoM were generated using the manipulator equation:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{B}\boldsymbol{\tau} + \boldsymbol{J}'_{foot}\boldsymbol{\lambda} \tag{A.3}$$

where $\boldsymbol{M}$ was the mass matrix, $\boldsymbol{C}$ the Coriolis matrix, $\boldsymbol{G}$ the gravitational potential matrix, $\boldsymbol{B}$ maps the applied forces on the system and $\boldsymbol{J}'_{foot}\boldsymbol{\lambda}$ maps the ground reaction forces presents at the foot to the generalised coordinates. The following formula was used to calculate the mass matrix:

$$\boldsymbol{M} = jacobian(jacobian(T_{total}, \dot{\boldsymbol{q}})', \dot{\boldsymbol{q}}) \tag{A.4}$$

where $T_{total}$ is the total kinetic energy in the system. The gravitational potential matrix was calculated using partial differentiation, as follows:

```
for i = 1:length(q)
    G(i) = diff(V_total, q(i));
end
```

where $V_{total}$ is the total potential energy in the system. The Coriolis matrix was calculated as follows:

```
for i = 1:length(q)
 for j = 1:length(q)
  for m = 1:length(q)
   temp = 0.5(diff(M(i,j),q(m))) + diff(M(i,m),q(j))
          - diff(M(j,m),q(i)))q̇(m);
    C(i,j) =  C(i,j) + temp;
  end
 end
end
```

The Jacobian of the foot position was calculated in order to map the GRF to the generalised coordinates, this was done as follows:

$$\boldsymbol{J}_{foot} = jacobian(\boldsymbol{P}_{foot}, \boldsymbol{q}) \tag{A.5}$$

where $\boldsymbol{P}_{foot}$ is the position of the foot. Lastly, the $\boldsymbol{B}$ was calculated. This matrix maps the forces (including torques) to the appropriate generalised coordinates. The $\boldsymbol{B}$ matrix changed depending on the model that was used, an example of the matrix be as follows:

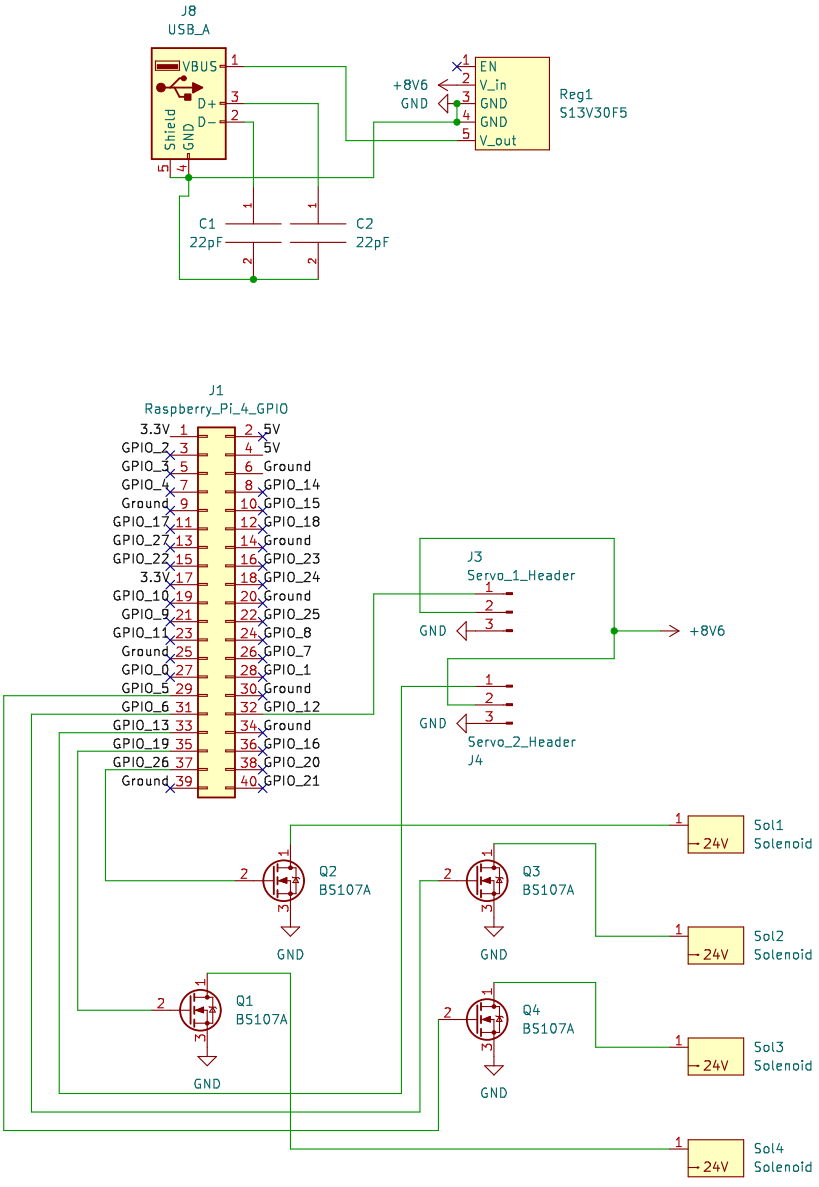
$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

and $\boldsymbol{\tau}$ would be:

$$\boldsymbol{\tau} = \begin{bmatrix} \textit{damping in } x\textit{-direction} \\ 0 \\ \textit{torque actuator left hip} \\ \textit{pneumatic actuator left leg} \\ \textit{torque actuator right hip} \\ \textit{pneumatic actuator right leg} \end{bmatrix} \tag{A.7}$$

# Appendix B

# Electrical Schematic



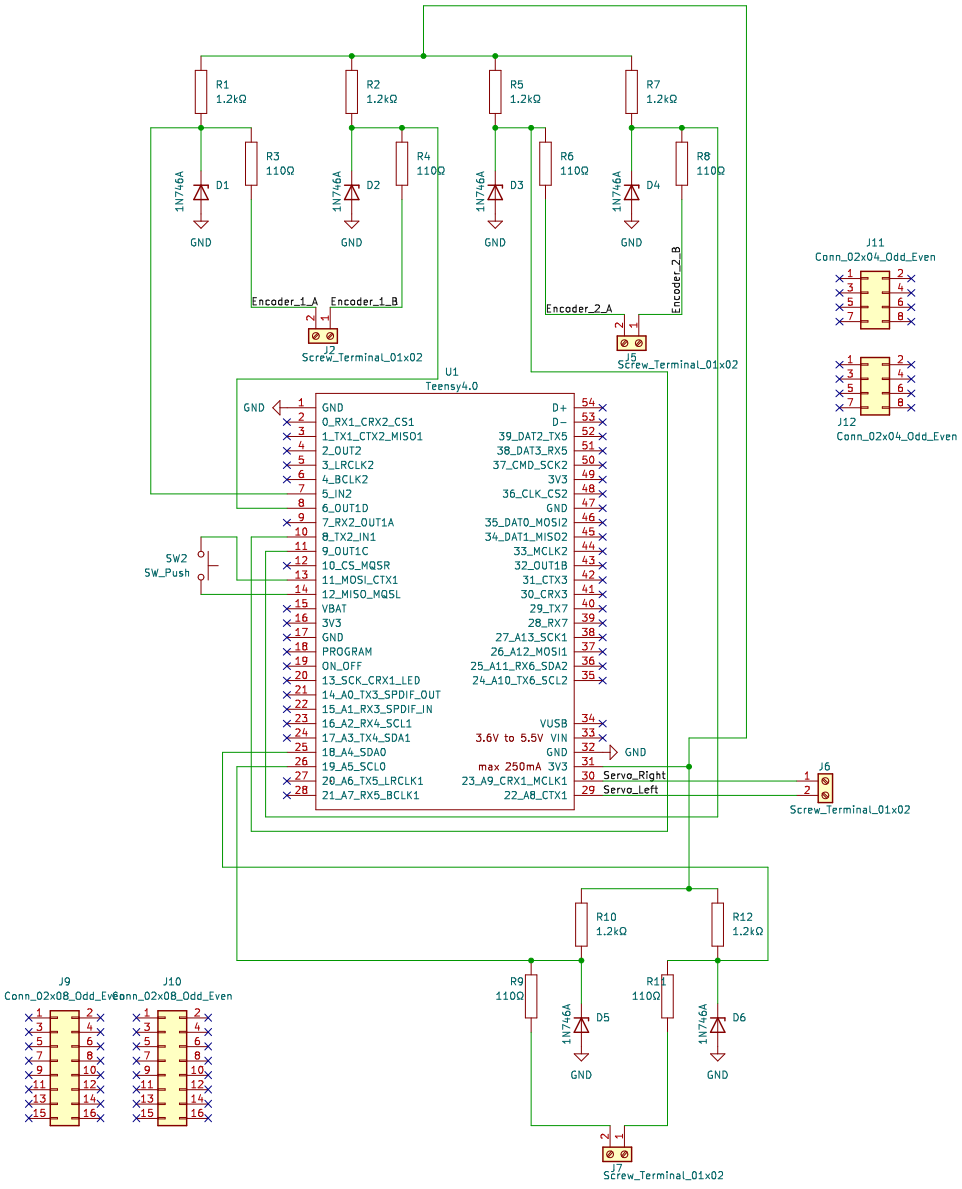



**Figure B.1:** The electrical schematic of the system.

# Bibliography

[1] Park, H., Wensing, P.M. and Kim, S.: High-speed bounding with the mit cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, vol. 36, no. 2, 2017.

[2] Zucker, M., Bagnell, J.A., Atkeson, C.G. and Kuffner, J.: An optimization approach to rough terrain locomotion. *IEEE International Conference on Robotics and Automation*, 2010.

[3] Fankhauser, P., Bjelonic, M., Bellicoso, C.D., Miki, T. and Hutter, M.: Robust rough-terrain locomotion with a quadrupedal robot. *IEEE International Conference on Robotics and Automation*, 2018.

[4] Mogg, T.: 7 times honda's iconic asimo robot blew us away (and 1 side-splitting fail), July 2018. Digitaltrends (Emerging Tech).
Available at: `https://www.digitaltrends.com/cool-tech/7-times-hondas-asimo-bot-showed-off-its-skills`

[5] Guizzo, E.: How boston dynamics is redefining robot agility, November 2019. IEEE Spectrum.
Available at: `https://spectrum.ieee.org/how-boston-dynamics-is-redefining-robot-agility`

[6] Fisher, C.: *Trajectory Optimisation Inspired Design for Legged Robotics*. PhD, University of Cape Town, 2020.

[7] Fisher, C., Blom, A. and Patel, A.: Baleka: A bipedal robot for studying rapid maneuverability. *Frontiers in Mechanical Engineering*, vol. 6, no. 54, 2020.

[8] Beck, C., Miró, J.V. and Dissanayake, G.: Trajectory optimisation for increased stability of mobile robots operating in uneven terrains. *2009 IEEE International Conference on Control and Automation*, 2009.

[9] Raibert, M.H.: *Legged Robots That Balance*. The MIT Press, 1986.

[10] Spot - the agile robot. Online available, 2022. Boston Dynamics.
Available at: `https://www.bostondynamics.com/products/spot`

[11] Meyer, J.: *Trajectory Optimization Inspired Pneumatic Locomotion on Compliant Terrains*. Masters, University of Stellenbosch, 2021.

[12] Kamimura, K.T.T., Aoi, S. and Matsuno, F.: Body flexibility effects on foot loading in quadruped bounding based on a simple analytical model. *IEEE Robotics and Automation Letters*, 2018.

[13] Gan, Z.J.Z. and Remy, C.: On the dynamic similarity between bipeds and quadrupeds: A case study on bounding. *IEEE Robotics and Automation Letters*, 2018.

[14] Xi, W., Yesilevskiy, Y. and Remy, C.: Selecting gaits for economical locomotion of legged robots. *The International Journal of Robotics Research*, vol. 35, no. 9, 2016.

[15] Reher, J., Ma, W. and Ames, A.D.: Dynamic walking with compliance on a cassie bipedal robot. In: *In 2019 18th European Control Conference (ECC)*, pp. 2589–2595. 1998.

[16] Posa, M., Cantu, C. and Tedrake, R.: A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–87, 2014.

[17] Patel, A., Shield, S., Johnson, A. and Biegler, L.: Contact-implicit trajectory optimization using orthogonal collocation. *Robotics and Automation Letters*, vol. 4, no. 2, p. 2242–2249, 2019.

[18] Luo, J., Su, Y., Ruan, L., Zhao, Y., Kim, D., Sentis, L. and Fu, C.: Robust bipedal locomotion based on a hierarchical control structure. *Cambridge University Press*, vol. 37, 2019.

[19] Mosher, R.: Test and evaluation of a versatile walking truck. *Proceedings of Off-Road Mobility Research Symposium*, pp. 359–379, 1968.

[20] McGhee, R. and Frank, A.: On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, pp. 331–351, 1968.

[21] Raibert, M.H., Brown, H.B. and Chepponis, M.: Experiments in balance with a 3d one-legged hopping machine. *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[22] Playter, R.R. and Raibert, M.H.: Control of a biped somersault in 3d. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 582–589. 1992.

[23] Zhao, M. and Qiu, Y.: Event-based control for pneumatic single-legged hopping robot. In: *2012 IEEE International Conference on Mechatronics and Automation*, pp. 297–302. 2012.

[24] Van Zyl, J.: *Rapid acceleration of legged robots: a pneumatic approach.* Masters, University of Cape Town, 2021.

[25] Raibert, M.H. and Sutherland, I.E.: Machines that walk. *Scientific American*, vol. 248, no. 1, pp. 44–53, 1983.

[26] Graichen, K., Hentzelt, S., Hildebrandt, A., Kärcher, N., Gaißert, N. and Knubben, E.: Control design for a bionic kangaroo. *Control Engineering Practice*, vol. 42, pp. 106–117, 2015. ISSN 0967-0661.

[27] Siciliano, B. and Khatib, O.: *Springer Handbook of Robotics*. 2nd edn. Springer Science & Business Media, 2016.

[28] Pratt, J. and Pratt, G.: Intuitive control of a planar bipedal walking robot. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 3, pp. 2014–2021 vol.3. 1998.

[29] Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C.D., Tsounis, V., Hwangbo, J., Bodie, K., Fankhauser, P., Bloesch, M., Diethelm, R., Bachmann, S., Melzer, A. and Hoepflinger, M.: Anymal - a highly mobile and dynamic quadrupedal robot. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 38–44. 2016.

[30] Lynch, K.M., Marchuk, N. and Elwin, M.: *Embedded Computing and Mechatronics with the PIC32 Microcontroller*. 1st edn. Elsevier, 2015.

[31] Kenneally, G., De, A. and Koditschek, D.E.: Design principles for a family of direct-drive legged robots. *IEEE ROBOTICS AND AUTOMATION LETTERS*, vol. 1, no. 2, 2016.

[32] Kaminaga, H., Yamamoto, T., Ono, J., Katayama, Y. and Shimoyama, Y.: The development of actuators with backdrivability, 2009. Copyright©2009, Nakamura & Takano Laboratory. All Rights Reserved.
Available at: `http://webpark1757.sakura.ne.jp/research/backdrive/index.php#bibliography`

[33] Wang, A. and Kim, S.: Directional efficiency in geared transmissions: Characterization of backdrivability towards improved proprioceptive control. *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[34] De, A. and Koditschek, D.E.: The penn jerboa: A platform for exploring parallel composition of templates. *arXiv:1502.05347*, 2015.

[35] Blackman, D.J., Nicholson, J.V., Ordonez, C., Miller, B.D. and Clark, J.E.: Gait development on a direct drive, quadrupedal robot. *SPIE Defense + Security*, 2016.

[36] Pratt, A.G. and Williamson, M.M.: Series elastic actuators. *MIT Artificial Intelligence Laboratory and Laboratory for Computer Science*, 1995.

[37] Fukuoka, A. and Kimura, H.: Dynamic locomotion of a biomorphic quadruped 'tekken' robot using various gaits: walk, trot, free-gait and bound. *Applied Bionics and Biomechanics*, vol. 6, pp. 63–71, 2009.

[38] Park, H., W., P.M. and K., S.: Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. *Robotics: Science and Systems Conference*, pp. 1–9, 2015.

[39] Park, H.-W., Wensing, P.M. and Kim, S.: Jumping over obstacles with mit cheetah 2. *Robotics and Autonomous Systems*, vol. 136, p. 103703, 2021. ISSN 0921-8890.
Available at: `https://www.sciencedirect.com/science/article/pii/S0921889020305431`

[40] Yao, L., Yu, H. and Zongxing, L.: Learning a contact-adaptive controller for robust, efficient legged locomotion. *Conference on Robot Learning*, 2020.

[41] Yao, L., Yu, H. and Zongxing, L.: Design and driving model for the quadruped robot: An elucidating draft. *Advances in Mechanical Engineering*, vol. 13, p. 168781402110090, 04 2021.

[42] Gealy, D.V., McKinley, S., Yi, B., Wu, P., Downey, P.R., Balke, G., Zhao, A., Guo, M., Thomasson, R., Sinclair, A., Cuellar, P., McCarthy, Z. and Abbeel, P.: Quasi-direct drive for low-cost compliant robotic manipulation. *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[43] Kau, N., Schultz, A., Ferrante, N. and Slade, P.: Stanford doggo: An open-source, quasi-direct-drive quadruped. *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[44] Ding, Y. and Park, H.: Design and experimental implementation of a quasi-direct-drive leg for optimized jumping. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[45] Pneumatic cylinder working principle. Online by ATO.com, 2019.
Available at: `https://www.ato.com/pneumatic-cylinder-working-principle`

[46] King, R.: Hydraulics vs pneumatics. Online available at:, 2020.
Available at: `https://www.rowse.co.uk/blog/post/hydraulics-vs-pneumatics`

[47] Raibert, M.H., Blankespoor, K., Nelson, G. and Playter, R.: Bigdog, the rough-terrain quadruped robot. *IFAC World Congress*, vol. 6, no. 11, 2008.

[48] solenoid valve cpe10-m1bh-5l-qs-6,. FESTO, 2018.

[49] Vadiati, A., Bagheri, A., Mahjoob, M. and Sadigh, M.J.: Design, control, and prototyping of a series elastic actuator for an active knee orthosis. *Engineering Solid Mechanics*, vol. 6, pp. 241–252, 01 2018.

[50] Elery, T., Rezazadeh, S., Nesler, C. and Gregg, R.: Design and validation of a powered knee-ankle prosthesis with high-torque, low-impedance actuators. *IEEE Trans Robotics*, 2020.

[51] Liou, H.-H. and Ho, M.-T.: Hopping control of a pneumatic single-legged robot using sliding mode control. In: *2019 International Automatic Control Conference (CACS)*, pp. 1–6. 2019.

[52] Binnard, M.B.: *Design of a small pneumatic walking robot.* Masters, Massachusetts Institute of Technology, 1995.

[53] Tedrake, R.: Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation. Course Notes for MIT 6.832, 2022.
Available at: `http://underactuated.mit.edu`

[54] Kelly, M.: An introduction to trajectory optimization: How to do your own direct collocation. *Society for Industrial and Applied Mathematics*, vol. 59, no. 4, pp. 849–904, 2017.

[55] Betts, J.T.: *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming.* Society for Industrial and Applied Mathematics, 2010.

[56] Wächter, A. and Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Subject Classification*, 2005.

[57] Ralph, D. and Wright, S.J.: Some properties of regularization and penalization schemes for mpecs. *Optimization Methods and Software*, vol. 19, no. 5, pp. 527–556, 2004.

[58] Fletcher, R. and Leyffer, S.: Solving mathematical programs with complementarity constraints as nonlinear programs. *Optimization Methods and Software*, vol. 19, no. 1, pp. 15–40, 2004.

[59] Hoheisel, T., Kanzow, C. and Schwartz, A.: Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints. *Mathematical Programming*, vol. 137, 2013.

[60] Carlo, J.D., Wensing, P.M., Katz, B., Bledt, G. and Kim, S.: Dynamic locomotion in the mit cheetah 3 trough convex model-predictive control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[61] Grimes, J.A. and Hurst, J.W.: The design of atrias 1.0 a unique monopod, hopping robot. *Adaptive Mobile Robotics*, 2012.

[62] Kurtz, V., Wensing, P., Lemmon, M. and Lin, H.: Approximate simulation for template-based whole-body control. *arXiv*, 2020.

[63] Kurtz, V., Da Silva, R., Wensing, P. and Lin, H.: Formal connections between template and anchor models via approximate simulation. *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019.

[64] Haynes, G. and Rizzi, A.: Gaits and gait transitions for legged robots. *IEEE International Conference on Robotics and Automation*, 2006.

[65] Azab, A.: Ros documentation. Available online, 2022.
Available at: `http://wiki.ros.org/Documentation`

[66] Arubai, N.: Ros concepts. Available online, 2022.
Available at: `http://wiki.ros.org/ROS/Concepts`

[67] Gazebo - physics parameters. Available online, 2014. Robotics Foundation.
Available at: `https://classic.gazebosim.org/tutorials?tut=physics_params&cat=physics`

[68] Smith, R.: Open dynamics engine. Available online, 2006.
Available at: `http://www.ode.org/ode-latest-userguide.html#sec_3_8_0`

[69] Urdf in gazebo. Available online, 2014. Robotics Foundation.
Available at: `https://classic.gazebosim.org/tutorials?tut=ros_urdf&cat=connect_ros`

[70] Siciliano, B., Sciavicco, L., Villani, L. and Oriolo, G.: *Robotics*. Springer Science & Business Media, 2010.

[71] Hubicki, C., Jones, M., Daley, M. and Hurst, J.: Do limit cycles matter in the long run? stable orbits and sliding-mass dynamics emerge in task-optimal locomotion. *IEEE International Conference on Robotics and Automation*, vol. 4, no. 15, 2005.

[72] Hart, E.H., Laird, C.D., Watson, J., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L. and Siirola, J.D.: *Pyomo - Optimization Modeling in Python*. 2nd edn. Springer Nature, 2017.

[73] Scholtes, S.: Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. *Society for Industrial and Applied Mathematics*, vol. 11, no. 4, p. 918–936, 2001.

[74] Festo pneumatic cylinder - 19233, 16mm bore, 125mm stroke, dsnu series, double acting. Online Available, 2021.
Available at: `https://docs.rs-online.com/f3fc/0900766b814f06de.pdf`

[75] Kitamura, T.: fusion2urdf. Repository On GitHub, 2021.
Available at: `https://github.com/syuntoku14/fusion2urdf`