Faculty of Engineering and the Built Environment
Department of Electrical Engineering

Hopping Control of a Single Leg Robot

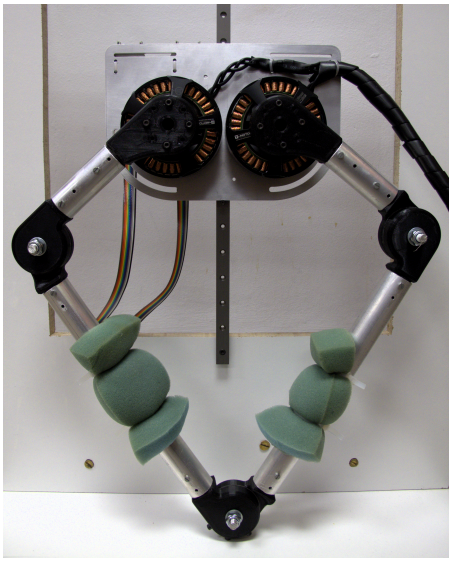
Prepared for Dr. Amir Patel.
Submitted to the Department of Electrical Engineering
at the University of Cape Town in partial fulfilment
of the academic requirements
for a Bachelor of Science (Eng.) degree in Mechatronics.

Benjamin Scholtz

November 14, 2016

Keywords: robotics, virtual model, compliance control, force control, mechatronics

To all the people that helped me jump!



Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this final year project report from the work(s) of other people, has been attributed and has been cited and referenced.
3. This final year project report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Name: Benjamin Scholtz
Signature:

Date: November 14, 2016

Abstract

A vertically constrained direct drive robotic leg platform was modelled, simulated, designed, built, and tested in order to better understand rapid acceleration control. The research was performed to investigate the following questions: **Is a virtual model a suitable replacement for accurate dynamic modelling in complex robotic topologies? Can high fidelity force control be effectively implemented without using force feedback? Is a virtual compliance control system effective in handling high speed impacts and executing rapid acceleration manoeuvres?** The dynamic model of the robot is complex, instead a virtual model uses simulations of components placed on the body of the robot to generate the desired end effector force response. The end effector was virtually modelled in the polar coordinate system as a radial and torsional series spring-damper. The desired virtual model motor torques were generated using the Jacobian kinematic mapping. Proprioceptive force control was possible due to the transparent coupling between the direct drive actuator and end effector. An iterative hardware and software design process was used to enable effective robotic testing - both an embedded communication and control system, and a GUI, were developed for the platform. Experiments were performed in virtual model spring-damping, impact absorption, trajectory tracking, force control, and current control. Jump tests were performed investigating robustness, repeatability, and rapid acceleration control. Force control and virtual model fidelity were verified by critically analysing both theoretical simulated responses and practical data. The robot generated an energy of $3.9\ J/kg$ with a maximum hopping height of $0.4\ m$, comparing well to the current state of the art. Robust hopping control was achieved with an 8.57% mean time shift and a negligible mean peak force deviation over 7 consecutive jumps. A robust robotic platform was successfully developed that enabled high fidelity force control using a virtual compliance model. The research contributed a platform and control framework that can be effectively used in future rapid acceleration research in the UCT Mechatronics Lab.

Acknowledgements

I'd like to thank my friends in the lab, some of whom were responsible for convincing me to take on this daunting project! Craig Burden, Gareth Callanan, Roberto Aldera, Munnawar Tayob, Michael Evans and Brad Stocks.

All the post graduate researchers in the lab (from fourth year number... 3?) – thank you for the endless support and missions to the UCT pub! I'd specifically like to thank Callen Fisher for the assistance, sharp mind, and sometimes harsh words to set us on track without hesitation!

To the mechatronics lab squash team – I hope to be able to play against you all again, I restrung my racquet just for that chance... Callen (still to beat), Stacey, Neil, Arnold, Givs, Tinashe (most improved and serious comeback!), Robyn.

Ben Bingham and Luke Bell – your contribution to the project was a great kick-start that helped avoid a lot of trouble! Thank you for being quick to help.

Brendan Daniels and Justin Pead – you were there day in and out, dealing with that pesky laser cutter, providing handfuls of components, defusing near blown LiPos, and providing your time to 3D print parts for Baleka.

My family – for all the tea ferried up to my room, dealing with my hermit like habits in the final stretch, and the care!

Last and certainly not least, my supervisor, Dr. Amir Patel – your guidance was strong, and your words and endless enthusiasm gave confidence when needed most! The project was a daunting task and you helped define my research vision.

Terms of Reference

Description

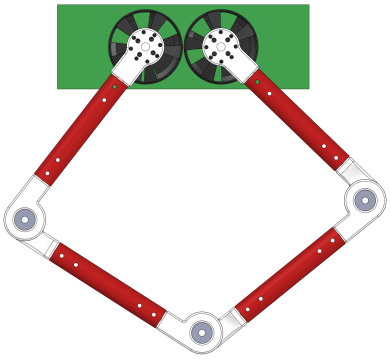


Figure 1: Version 1 of Baleka leg platform (Ben Bingham, 2016).

The Mechatronics Lab has recently developed a single leg direct drive robot, Baleka, to investigate modelling and control of rapid accelerations. This project will involve the design of a control system to perform stable hopping with the robot. Various controller algorithms will be investigated and compared (eg. PID, MPC, etc.). The project will also involve developing a test rig for the robot.

Deliverables

- Mathematical model of the hopping robot must be developed in Simulink/Matlab
- Hopping controller design
- Mechanical design of the test rig
- Experimental testing of the robot

Skills/Requirements

- Mathematical Modelling
- Mechatronics Design
- Control Systems
- Embedded Systems
- Strong Practical and Mathematical skills required

ELO3: Engineering Design

Perform creative, procedural and non-procedural design and synthesis of components, systems, engineering works, products or processes.

The student is expected to design:

- Robot feedback control system
- Rig for testing of hopping motion

Area of Research

- Bio-inspired robotics
- Control systems

Extra Information

`http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5648972` `http://kodlab.seas.upenn.edu/uploads/Avik/compositionTR_sc.pdf`

Contents

List of Figures

List of Tables

List of Source Codes

1 Introduction

> *"Begin at the beginning," the King said, gravely, "and go on till you come to an end; then stop."*
>
> — Lewis Carroll, *Alice in Wonderland*

With a hop, skip, and a jump – the journey begins!

1.1 Background

The robotics leg, affectionately named Baleka, is one small step in a greater project - the UCT Mechatronics Lab Cheetah project. Baleka means *run away*, in Xhosa. The leg linkage system was initially designed and built in the mechatronics lab. After completion of the undergraduate thesis project the robotic leg will be further developed by postgraduate researchers in the laboratory, using the same geometric leg configuration.

Baleka will be used to investigate modelling and control of rapid accelerations, with a controller developed to perform stable consecutive hopping with the robot.

1.2 Objectives of the Study

1.2.1 Purpose of the Study

The purpose of this study is to develop a robust robotic leg platform and testing rig capable of rapid acceleration and high fidelity force control experimentation.

1.2.2 Problems to be Investigated

- High speed embedded system communication and packet processing.
- Virtual model control for accurate end effector force output.
- Effective high speed kinematic control.

- Effective use of motor drivers to achieve rapid accelerations with a direct drive BLDC motor.
- Development of a platform suitable for further use in dynamic legged motion research.

1.2.3 Research Questions

- Is a virtual model a suitable replacement for accurate dynamic modelling in complex robotic topologies?
- Can high fidelity force control be effectively implemented without using force feedback?
- Is a virtual compliance control system effective in handling high speed impacts and executing rapid acceleration manoeuvres?

1.3 Scope and Limitations

The scope of this project covers:

- Development of a mechanical robotic leg platform using available resources.
- Development of a control system capable of high fidelity force control for rapid acceleration hopping.
- Development of a GUI for robot configuration, data logging and live plotting.
- Development of basic robotic models.
- Effective configuration of motor drivers based on motor model.

The limitations of this project include:

- Time frame of one semester to complete the research.
- Motor drivers with limited communication speeds and peak current capability.
- Limited budget for extended mechanical development.

1.4 Plan of Development

This study will initially investigate the current state of the art based on pre-existing legged robotic platforms and control techniques applicable to this project.

The project plan and methodology will develop an outline of the structure of the investigation and scientific method.

The leg kinematics and dynamic model will be developed and simulated, with a focus on the control system to be implemented.

The mechanical and electrical construction and design of the leg will be covered extensively followed by the development of the communication protocol and graphic user interface.

The control system will first be developed and simulated, before implementation on the robotic platform to iteratively improve the design.

Experimental testing will be performed in the following section which will include spring-damper tests, drop tests, trajectory tracking tests, force control tests, and finally hopping tests.

The final design and implementation of the robot will be critically validated before the study ends with conclusions and recommendation for future work based on the initial objectives of the study.

In the appendix code listings and other miscellaneous material will be included for extended study.

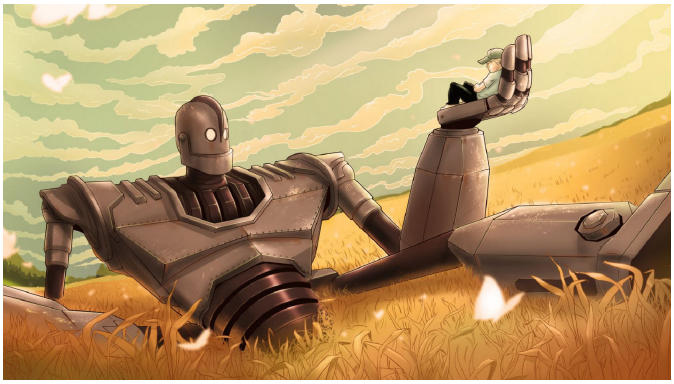
2 Literature Review

2.1 Introduction

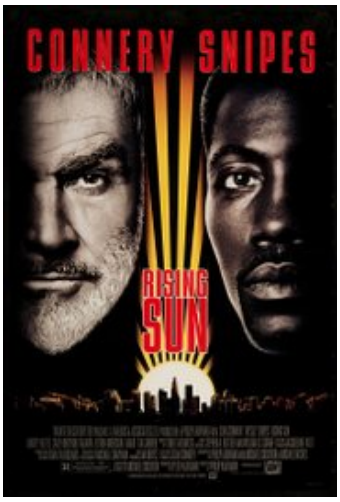
Would be robotics engineers take their inspiration from popular culture with The Iron Giant (fig. 2.1a) and B.E.N. (fig. 2.1c) fresh in mind. The Rising Sun included robots developed by Marc Raibert (fig. 2.1b), founder of the CMU (now MIT) Leg Laboratory, who pioneered self-balancing dynamic control of hopping robots.

Robotics turned towards pairing complex dynamic responses with simple control systems when Marc Raibert developed the dynamically stable Planar One-Leg Hopper in 1980. Development continued until the Atlas humanoid robot (2013) of Boston Dynamics achieved human like static stability and the Cheetah Robot V2 (2015) of MIT Biomimetics achieved lifelike legged manoeuvres.

The literature covered in this review will lead us from the natural locomotion of Kangaroos to the supernatural motion of complex legged machines. From one-legged hoppers to Baleka like forms, these robots will each be critically discussed to draw parallels between the current state of the art and Baleka as well as discovering the contribution that Baleka can make to current research. The art of force control will be touched on before the applications of such control will be seen in industry. Modelling, control, and actuation methods will be investigated and considered for use on Baleka.



(a) The Iron Giant (1999).



(b) Rising Sun (1993).

(c) Treasure Planet (2002).

Figure 2.1: Humanoid robots in popular culture.

2.2 Legged Locomotion in Nature

The study in [1] modelled the human leg as a spring loaded inverted pendulum, as seen in fig. 2.2, and found that it described the mechanics of biological running well. The study concluded that the biological leg spring has a higher spring constant for higher stride frequencies.[1] A stiffer spring is able to absorb and impart more energy over the course of a stride, thus enabling a higher stride frequency. The Baleka leg is limited to movement in a single vertical axis, but the virtual spring stiffness should have the same effect on energy transfer as found in this study – a transfer of energy from spring energy to peak potential energy at the highest point of the jump.

When animals run, they use a hopping movement. This hopping movement is defined by limited contact with the ground with phases of mid-air motion. Cavagna et al found that they use musculoskeletal springs to store and return elastic energy.[1] This storage and release of elastic energy is the principal on which Baleka's control system will be developed - by controlling the theoretical transfer of energy between jump phases one should be able to achieve height control and impact absorption, as discussed in chapter 5.

Farley et al determined that the leg spring stiffness of animals remains the same at all speeds, this shows a clear natural design element that serves a purpose in energy transfer and control during running.[1]

The study in [10] found that the hopping motion of a Kangaroo is split into two phases, a contact phase and a floating phase. The phases were investigated using kinetic and potential energy, where it was found that at high speed there were more fluctations in kinetic energy and less in potential energy - this means that work is being done and energy is being added to the system through actuation of the Kangaroo's tendons.[10] In a simple hopping motion on the other hand, as is the case with Baleka, the majority of the kinetic energy should be efficiently transferred to potential energy.

[10] also mathematically derives the energy relation of launch and impact on the Kangaroo's forward motion. It was found that the line of action of the force exerted by a Kangaroo on the ground always passes through the center of mass.[10] This is useful theory in the case of Baleka being modelled as a spring-damper mass system, as it confirms the assumption that the force input to the system can be simplified as a gravitational force acting on the center of mass.

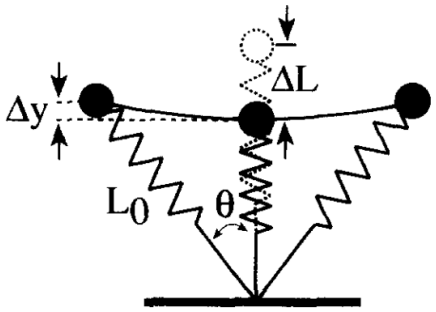


Figure 2.2: Model of a human leg as an inverted spring mass system - Farley and Gonzalez (1996).[1]

2.3 State of the Art in Legged Robotics

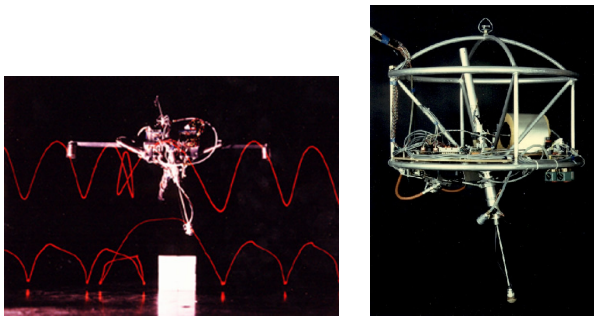
The development of dynamic hopping control in robots started under Marc Raibert's guidance at the CMU Leg Laboratory in 1980. He then moved the lab to the MIT Leg Laboratory and founded Boston Dynamics in 1992. Marc Raibert, the namesake of Raibert control, developed the first truly simple and beautiful self-balancing hopping controllers which will form the basis for the research being completed on Baleka.[11]

The study of the state of the art of legged robotics will develop an understanding of leg control algorithms and methodologies, as well as determine where Baleka fits in and contributes to the current research.

2.3.1 Monoped Robots

Using the Planar One-Leg Hopper in fig. 2.3a it was determined that given constraints of balance and controlled travel, a hopping motion emerges naturally.[12] These natural dynamics are considered in the development of Baleka, where a virtual model is used while allowing the natural dynamics to take place, and non-conservative forces are effectively ignored. The techniques for planar one-leg hopping were generalized for the 3D case and used in the 3D One-Leg Hopper.

The 3D One-Leg Hopper seen in fig. 2.3b was built using hydraulic and pneumatic actuators for hip angle and leg extension control. A single legged robot was used as it simplifies the study of self balancing control algorithms, with any more than one leg the coupling between these legs needs to be considered.[13] This is essentially why Baleka was developed as a single leg, single axis machine - the simplicity in control allows a



(a) Planar One-Leg Hopper - MIT Leg Laboratory (1980-1982).

(b) 3D One-Leg Hopper - MIT Leg Laboratory (1983-1984).

Figure 2.3: Monoped robots.

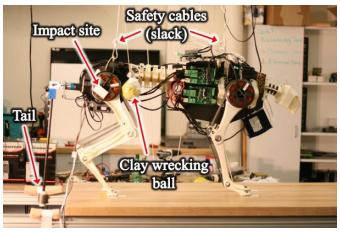
more comprehensive study of the hopping motion dynamics without coupling interfering.

The 3D One-Leg Hopper was one of the first robots built by Raibert where it was found that a simple control algorithm, for isolated behaviours - forward running, attitude control, and hopping height - could be used to effectively control the hopping of a robot in 3 dimensions, treating any coupling between these behaviours as disturbances to the isolated control algorithms.[13] Baleka uses a similar principal implementing control of phase energy with a basic virtual model control system as developed in chapter 5.

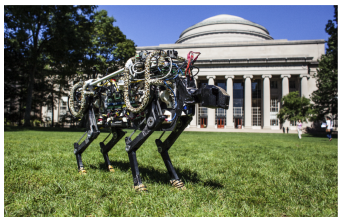
2.3.2 Quadruped Robots

The quadruped robots in fig. 2.4 all pose the unique challenge of achieving incredibly high torque from their actuators to account for the large mass of the robots. In the case of the study of [9] seen in fig. 2.4c the leg was designed for high force proprioceptive control, but was never implemented in a full robot. Both the MIT Cheetah robots were successfully built and used.

The high torque required was achieved, as further discussed in section 2.6, by using a highly efficient and mechanically transparent planetary gear system. This allowed proprioceptive force control which is further investigated in section 2.5.



(a) Cheetah Robot - MIT Biomimetics (2012).



(b) Cheetah Robot V2 - MIT Biomimetics (2015).

(c) GOAT 3-DOF Leg Topology - (Kalouche, 2016).

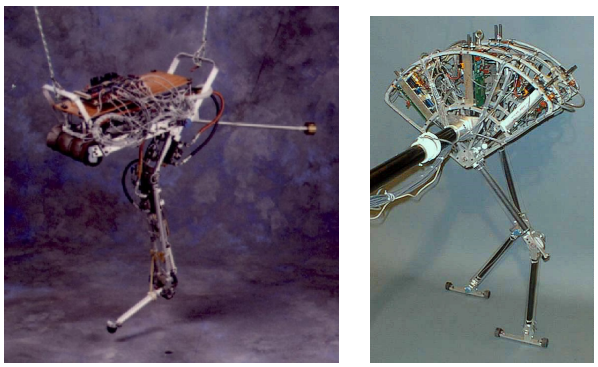
Figure 2.4: Quadruped robots.

2.3.3 Bio-inspired Legged Robotics

Animals make use of tendons and muscles to both store and release energy to assist with walking, and legged robots can learn a lot from them. By using various virtually and mechanically compliant components in place of tendons and muscles, the following robots seen in fig. 2.5 are bio-inspired.

The Spring Flamingo was created by Jerry Pratt from 1996-2000.[14] It is a unique combination of mechanical spring-damper components (or Series Elastic Actuation - S.E.A.) as well as virtual model control.[14] Baleka makes use of the virtual model control method developed by Pratt et al during these initial experiments. The Spring Flamingo, being just under a meter tall and with an awkwardly positioned center of mass as seen in fig. 2.5b, required many actuated joints to stay up right. The robot had actuators at the hip, knee and ankle to implement virtual compliance control along with various rotary and linear potentiometers were needed to measure compression and movement.[14]

The Spring Flamingo research specifically investigated how actuated joints can help with bipedal walking and creating a robust and shock resistant robot.[14] Shortly after the research, a paper was produced by Pratt et al in [15] detailing the virtual model control methodology - the work on Spring Flamingo as well as virtual model control formed the basis for the spring-damper mass model of Baleka. Baleka, in its various



(a) Uniroo - MIT Leg Laboratory (1991-1993).

(b) Spring Flamingo - MIT Leg Laboratory (1996-2000).

Figure 2.5: Bio-inspired legged robots.

topologies, essentially had virtual compliance in its hip, its joints and in compression as developed in chapter 5.

2.3.4 Closed Kinematic Chain Leg

The use of a closed kinematic chain can lead to singularities in kinematic control - multiple acutator positions for a single position, as well as complex kinematic mappings, can exist. The study [16] proved this in its derivation of the kinematic model for a 5 bar linkage and was seen in the case of Baleka even for a simplified kinematic model in section 4.4.

Two robots that have successfully used the 5 bar linkage design are [2] and [3], seen in fig. 2.7. [3] used the assumption that the motors were co-linear, and [2] actually implemented co-linear motors in order to simplify the kinematic derivation. Baleka used the kinematic equations first developed in [2] and later adapted for [3].

Figure 2.6 shows the various kinematic chain configurations investigated by [2]. Baleka used a simple kinematic workspace plot, section 4.4, to determine what the best configuration would be - only using symmetric linkage designs.

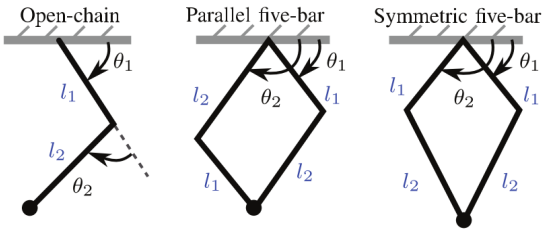


Figure 2.6: Kinematic chain linkage leg designs as implemented by Kenneally and Koditschek.[2]

Figure 2.7: Closed Kinematic Chain Leg using Raibert's Scissor Algorithm (Duperret, Koditschek, 2016).[3]

2.4 Dynamic Modelling

Dynamic modelling in this context involves taking the kinematic robot model and describing its dynamic behaviour in such a way that allows it to be actuated, usually to implement force control. A few of these methodologies will be investigated as implemented in past research, and considered for use in the Baleka case.

2.4.1 Virtual Model

A virtual model is an imaginary construct of various virtual components on an existing system. Instead of cancelling the dynamics of the existing system and imparting its own on the system it attaches the virtual components to the geometric constraints of the model and applies the necessary actuated joint torques to impart the response of the system, with its natural dynamics, had the components actually been present.[15]

[9] uses a virtual model to describe the robotic leg seen in fig. 2.4c. This allows control of the robot using a model that would have otherwise been incredibly complex. As further described in section 2.5, there are design choices necessary to ensure high fidelity force control using a virtual model.

A few of the benefits of a virtual model include:[15]

- Endless configuration options.
- Potentially simple computation.
- Allows compact mechanical design with complex robot dynamics.
- A high level controller can be used that simply changes the virtual model as the environment dictates.

Baleka, although not as complex as [9], would benefit from the design choices made in the study in order to control the leg without the use of a more complex model or force feedback.

2.4.2 Lagrangian Model

In the study [16] a Lagrangian model and control system was developed for a hybrid machine system (constant speed motor with servo motor) with a five bar linkage end

effector. The robot was not intended for dynamic movements, but rather for use in a known factory environment.

In the case of developing a Lagrangian model and working with generalized coordinates, an accurate kinematic map is needed. The kinematic model derived in [16] was not trivial and the associated Lagrangian model neither. Ideally these models would be able to be used on Baleka, but geometrically it is an inverted form of Baleka that uses different general coordinates in a Cartesian system, whereas Baleka used a polar coordinate system.

To adapt the Lagrangian model used in [16] would have been out of the scope of this project, where the development of a platform for testing dynamic control algorithms was the primary aim, and the actual modelling of the leg secondary to that.

2.4.3 Raibert Model

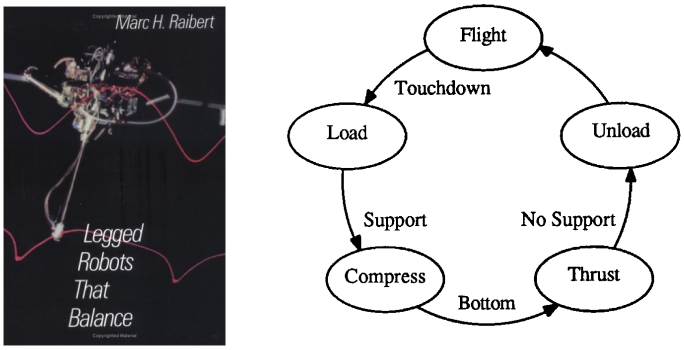
Marc Raibert investigated hopping in the two dimensional one-legged case in 1984 in the study [13]. This was the beginning of what is today known as Raibert control. Raibert's control methodology, as applied to the robots he helped develop, was extensively documented in the book seen in fig. 2.8a.

Raibert control has three simple core principles extracted from [13]:

1. Hopping can be decomposed into separate control algorithms maintaining forward velocity, attitude, and height.
2. The dynamic hopping motion can be considered in phases with defined kinetic and potential energy relations.
3. The natural dynamics of the robot are allowed to take place and any coupling of the control algorithms is treated as a disturbance.

Raibert recognized that running and hopping are simply phases of intermittent support with a flight phase in between. The support phase can be further broken down into stance and launch phases. This methodology is clearly shown in fig. 2.8b.

Raibert's use of energy conservation assumes that energy is efficiently transferred from one phase to the other and using these energy transfers things like hopping height can be controlled, any errors in these calculations can be fixed with a height sensor and integral error term. Although height control was theoretically implemented in Baleka with relative accuracy, no feedback for error correction was used.



(a) Legged Robots That Balance - Marc H. Raibert (1986).

(b) Raibert control state machine.

Figure 2.8: Legged Robots That Balance cover page and exert.[4]

The beauty of Raibert control is the simplicity of the control algorithms. These algorithms don't require an accurate model, but rather allow natural dynamics to take place. In the case of Baleka, a dynamic model was not used, yet dynamic control was still achieved.

2.5 Force Control

Force control comes in many forms, the most important characteristic being that force is the primary controller output - in some cases combined force and position control is implemented, or uncontrollable mechanical compliance components are integrated into the robot, or controllable virtual components are used. These topologies will be discussed relating to the Baleka use case.

2.5.1 Mechanical Compliance

Mechanical compliance can be built into a robot passively, or in some cases actively as is the case with the tunable stiffness leg seen in fig. 2.9 from the study [5].

As was shown in section 2.2 animals benefit in stability and energy output from having springy or elastic legs. In the case of unknown terrain it will be useful to be able to

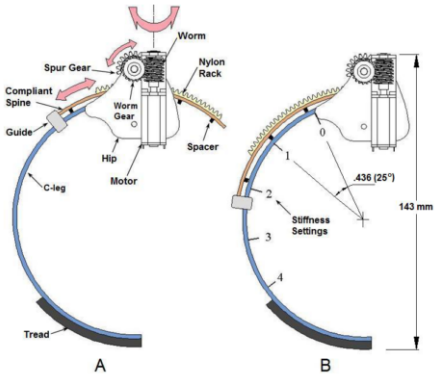


Figure 2.9: Tunable stiffness leg for dynamic locomotion.[5]

tune this mechanical compliance component. Figure 2.9 shows a structurally controlled variable stiffness leg that allows the leg to be set to a number of compliant set points. This is a compromise between passive mechanical compliance and full virtual compliance control.

Series Elastic Actuators

[17] developed an eletromechanical robot design that made use of Series Elastic Actuators (S.E.A.). The aim of the study was to design a dynamic machine that could manipulate itself as well as its environment while conserving energy - potential energy should be effectively transferred to kinetic energy and kinetic impact energy stored for later use.[17] A similar methodology was used with Baleka to achieve jumping control.

In the case of [17], a S.E.A. was implemented as a linear spring of set stiffness in parallel with the motor. By using the S.E.A. in parallel with the motor it means the motor is put under less strain trying to keep the necesarry reaction forces to maintain compression of the spring and store energy.[17]

Series elastic actuation, as with animal tendon and muscle usage, provides the following benefits as described in [17]:

- Decreased reflected motor inertia: Elastic actuation ensures actuation can happen immediately when stored elastic energy exists, rather than having to deal with motor inertia.
- Stable force control: Sudden impacts are effectively absorbed by the S.E.A. instead of the body, and force interaction with the environment is managed more effectively.

- Elastic energy storage: much like the Kangaroo tendons store energy to enable incredibly efficient jumping during extended periods with little extra work.

[17] mentions that the cost of stable force control is a decrease in sensing and actuation bandwidth. This is the case with passive compliance, but variable compliance allows wider actuation bandwidth by being able to reconfigure the robot in different environments. Baleka uses a similar set up to maintain a high actuation bandwidth, but using purely virtual compliance components.

2.5.2 Virtual Compliance

In [15] a virtual model control framework was developped. This was developped in 2001 after extensive testing of these frameworks had already been performed with the Spring Flamingo. The various robots that have implemented virtual model control have already been discussed, now an outline of virtual model control as related to compliance control will be discussed using [15] as the basis.

Virtual model control uses virtual components to provide the leg or robot with some form of compliance;[15] whether that be for handling fine objects, or storing and releasing energy as is the case with Baleka. These virtual components can be springs and dampers as used in [9] and Baleka, or any other component that alters the dynamic response of a robot in a simply defined way. The virtual component force response is mapped to the motor torques and this creates the illusion that the components are actually attached to the robot, as defined in [15].

Virtual model control allows a robot to be compact and computationally simple.[15] It is also highly reconfigurable, as developed in chapter 8 where the Baleka leg had characteristics of the leg changed at phase transitions.

Virtual model control can either replace, or complement, series elastic actuation as seen in the Spring Flamingo case.[15] In the case of Baleka series elastic actuation was omitted due to the added complexity and cost, but in other cases it allows more energy to be stored and released quickly as in [3].

Both [3] and the Spring Flamingo in fig. 2.5b use mechanical compliance in the form of joint springs as well as virtual compliance - the two concepts covered above.

2.5.3 Proprioceptive Force Control

Proprioceptive force control is the ability to estimate the force experienced at an end effector without actually having force sensor feedback - it relies on the fact that the motor is directly coupled to the leg system and/or has minimal non-conservative forces and impedance.[6]

In [6] transparency is stressed as the primary design goal needed to achieve proprioceptive force control. Transparency means that any torque exerted on the end effector is the same torque experienced by the motors after having a kinematic mapping applied - where kinematics implies a positional relation and does not take into account leg dynamics.[6] The choice of actuator is also important, as a clearly defined relationship between actuator torque and control input needs to exist, otherwise it adds another uncertainty that reduces transparency. In the case of Baleka, a current command will be used for force control and thus the torque constant, K_t, needs to be properly callibrated as it was in subsection 6.6.3. [6] also points out that using a force sensor could lead to limit cycles and instability if not properly calibrated - this was seen in the case of Baleka where instability existed if the virtual model was not configured correctly and thus the estimated forces were incorrect.

2.6 Actuator Design

Industrial robots are typically high power and highly geared systems - they can afford to be. The advent of drones and bio-inspired robotics research inspired the development of high torque and low inertia motors available off-the-shelf. The T-Motor brushless DC motors used by Baleka have been used in a number of robots previously, and for good reason - they enable low speed high torque direct drive control. The following discussion outlines some of the research in direct drive robotics.

The study [2], titled "Design Principles for a Family of Direct-Drive Legged Robots", quantified the pros and cons of direct drive mechanisms and investigated the effect of various kinematic and control design choices on direct drive performance.[2] The study specifically investigated a 5 bar linkage design as further developed in [3] and used in the design of the Baleka topology as seen in chapter 4.

A number of advantages and disadvantages of direct drive robotics were discussed in the study [2], a few that are important to the design of the Baleka leg are summarised:

Advantages

- **Transparency:** By directly coupling the motor to the leg, mechanical impedance is reduced. A higher fidelity virtual model is able to be achieved as torque is more efficiently transferred to the end effector - this avoids the use of force sensing.
- **Mechanical performance:** A less complex system ultimately leads to more robust and efficient operation. This simplicity can also help in the design of and control of dynamic motion models.

Disadvantages

- **No torque amplification:** In a gearless system there is no speed and torque amplification. The robots inherently need a lot of torque. This means the direct drive motors must operate in the high torque low speed regions of the motor design, which was confirmed later on in this study in subsection 6.6.2 and shown in fig. 6.13. The T-Motor U10 Plus motors are rated for maximum efficiency and power well out of the low speed high torque range that Baleka will operate at.

[2] investigated the leg Jacobian singular values and motor thermal cost for various kinematic positions. Baleka contributed to this research with the simulation and description of the effects of kinematic position on motor current draw, torque output, and end effector force output in section 8.5 and subsection 6.6.1.

[2] found that near stall conditions a direct drive system can be energetically expensive. Two performance measures exist for motor selection, namely instantaneous performance and steady performance - instantaneous being the impulsive torque that can be achieved and steady the thermal torque that is possible.[2] The idea of measuring thermal torque should be investigated - in the Baleka design, problems were encountered with steady performance heat dissipation, as discussed in subsection 6.2.1 where an aluminium plate was used to increase the possible thermal torque.

The robotic study in [6], pictured in fig. 2.10, developed a planetary geared robotic leg for use in the MIT Cheetah project using a custom made actuator. A similar planetary gear system was later developed and used by [9], pictured in fig. 2.4c, this time with the same off-the-shelf T-Motor brushless DC motor used in Baleka. Although these gear systems bring the same benefits of direct drive, with the torque amplification of gearing, they are incredibly complex and expensive to develop. In the case of Baleka, the torque capabilities of the motors were at their limits. Baleka is a simple one leg hopper with a relatively high torque to mass ratio - the planetary geared systems used in [6] and [9] were necessary because of their full body robotic design which requires more torque to

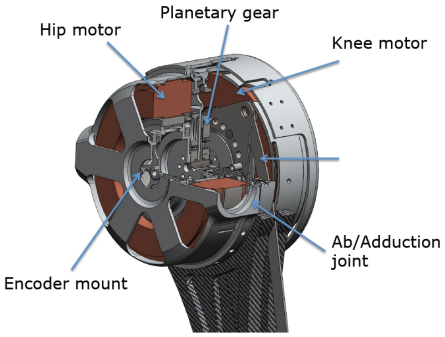


Figure 2.10: Planetary geared robotic leg.[6].

compensate for added mass.

2.7 Applications in Industry

2.7.1 Bose Active Suspension

Active and semi-active suspension feature in many of today's high end cars. The purpose of active suspension is primarily for ride comfort, but also:[18]

- Maintaining vehicle posture during manoeuvres and external disturbance.
- Road handling and vehicle agility.
- Avoiding abrupt movements.

Active suspension is effectively a form of compliance control like that seen in [15]. Baleka will use a virtual compliance model with BLDC actuators in order to implement "active suspension", whereas vehicles are forced to use a more heavy duty active suspension system.

A common method to implement variable suspension damping in vehicles is to use magneto-rheological fluids. The flow characterisitics of these fluids can be changed electronically allowing the compliance of fluid shocks to change quickly.[18] The Bose active suspension system, seen in fig. 2.11, used the magneto-rheological fluids method of actuation.

It was noted in [18] that active suspension leads to higher energy consumption, cost, complexity, and operational requirements. In comparison the form of active compliance control implemented in Baleka is intended to decrease mechanical complexity and cost

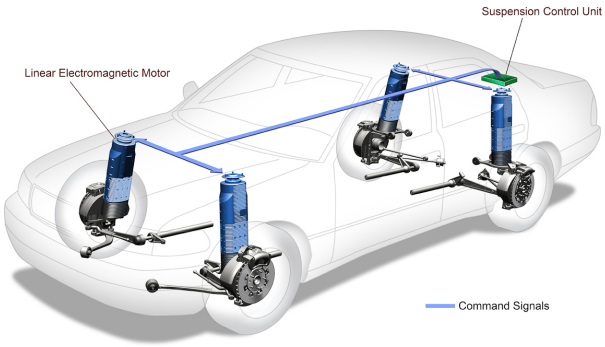


Figure 2.11: Bose Active Suspension (Bose Corporation, 1980s)[7].

compared to using real spring-damper components. Energy consumption and operational requirements should be investigated in the case of Baleka.

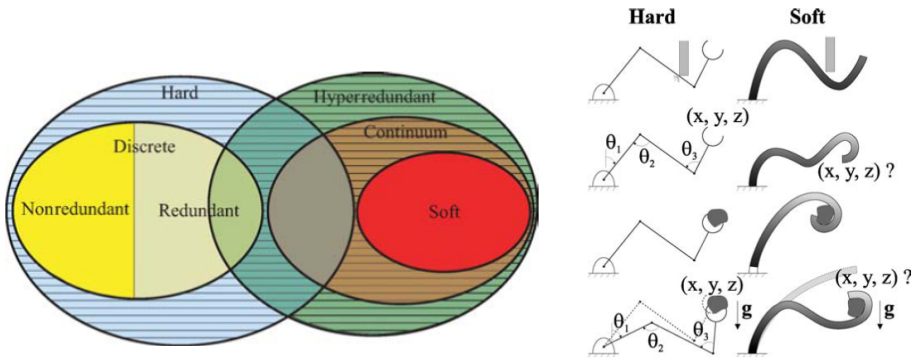
Active suspension control systems use optimal control and H2 robust control methods with performance being measured by RMS values of sprung mass acceleration and tyre deflection.[18] In the case of Baleka a more simplistic force controller is used with an ideal spring damper mass motion in mind.

2.7.2 Soft-robotics

The industrial revolution brought the widespread use of machinery to perform repetitive tasks, and with it acts that protect factory floor workers. Injuries from human machine interactions were common, modern safety standards tend to prevent accidents like this from happening. Human labour is still needed for quality control and certain complex awkward tasks in factories.

The development of soft robotics, and hard robotics with compliance control, has made human machine interaction more natural and safe.

Traditional robots with rigid structures have limited predefined workspaces and ways in which they can interact with the environment.[8] Soft robotics enables a robot and its manipulator to operate with multiple degrees of freedom in unstructured environments.[8] A good example of an unstructured environment is in robotic surgery, a war-zone, or exploring unknown terrain. All of these situations would benefit from a robot that can adapt in the way it interacts or manoeuvres in its environment.



(a) Classification of robots based on materials and degrees of freedom.

(b) Capabilities of hard and soft robots.

Figure 2.12: Soft robotic classification and capabilities.[8]

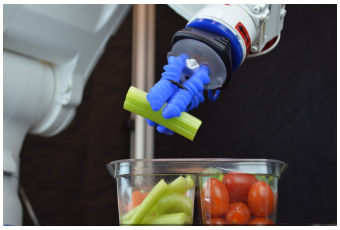


Figure 2.13: Compliant soft robotic handling (Forbes, 2016).

Robots are not only defined as hard or soft based on the compliance of their materials as stated in [8], but also on their ability to comply with their environment. In the case of Baleka active compliance will be virtually implemented. Baleka would still be defined as a hard robot, as seen in the definition in fig. 2.12, but another category is maybe needed for environmental interaction and compliance. Baleka has rigidly defined kinematics, but in its interactions and responses will have properties common in soft robotics.

Another industrial application of soft robotics and even compliant robotics is the handling of compliant products, as seen in fig. 2.13, and delicate products. Compliant products exist in farming - fruit and vegetables need to be handled with care to prevent unnecessary forces being applied during packaging. Manufacturing of delicate products such as pottery and porcelain require precise force limits, using soft robotics or compliant robotics we can better handle these products.

3 Project Plan and Methodology

"You know my methods, Watson."
— Sherlock Holmes in Arthur Conan Doyle, *The Crooked Man*

The project methodology is summarised in fig. 3.1.

Further analysis of the necessary modelling, simulation and design elements are presented.

A plan is presented for experimentation to both verify design choices and answer research questions.

3.1 Modelling, Simulation & Design

The geometric design of the leg, having already been chosen, will be further developed in theory and kinematic workspace simulations will be performed to confirm the leg linkage lengths for optimal performance. The kinematic mappings of the leg will be developed as the final step in leg design.

A virtual compliance model needs to be developed to achieve: launch energy control, impact energy control, and steady state performance. First the low level spring-damper mass motion will be considered, deriving the necessary equations to design a suitable virtual compliance model.

Both the dynamic and kinematic models will be simulated and iteratively corrected if necessary.

The design of the leg will be split into hardware and software elements - where hardware includes mechanical and electrical design, and software includes embedded system and GUI design. Throughout the hardware design of the leg each element will be critically considered by modelling, iterative design, and theoretical design, as the case dictates.

A kinematic workspace model will be developed to present predictions of current usage, motor torques, and end effector forces - this will help validate hardware and software design choices with the research questions in mind.

The software design will be developed after the hardware design is at a usable level. This will enable testing of the system as it is developed and addition of necessary functions. The core of the software development will involve the design of a communication system for all elements of the electrical hardware design.

A controller will be developed through theory and simulation, before being implemented and tested in experimentation. This will be an iterative process where hardware, software, and the underlying control system will be adapted to meet the necessary specifications.

3.2 Testing & Verification

The aim of this project is to develop a robust robotic platform capable of performing high acceleration jumps using a virtual model - the experiments conducted will reflect that and develop the control system to such a point that it meets these design specifications.

At each stage of experimentation the scientific method will be introduced, where aims of the experiment are outlined, before the experimental data is analysed and a summary is presented clarifying whether the aims have been met. Verification of the experiments will result from comparison of the experimental data to previous simulated and theoretical data as well as critical analysis.

Data will be logged including control system data, virtual model forces, motor signals and kinematic positions. Along with this data all experiments will be recorded for later analysis. The leg will be mounted on the robotic guide platform.

Tests will be performed to investigate various virtual model spring-damper topologies, and the fidelity of the response achieved. These experiments will be presented as a comparison between practical data and theoretical design. If the theoretical design is not shown in the practical data then the necessary steps will be taken to correct that if possible - this is the iterative design component of the experiment.

Drop tests will be performed to determine the best virtual model configuration for leg impact, and to investigate the robustness of the leg design.

The spring-damper tests will inform the configuration of the virtual model for jump tests, where the spring-damper constants will be chosen from experimental analysis. The jump tests will investigate all phases of jumping individually.

The design specifications say the leg needs to be robust. In order to test this an

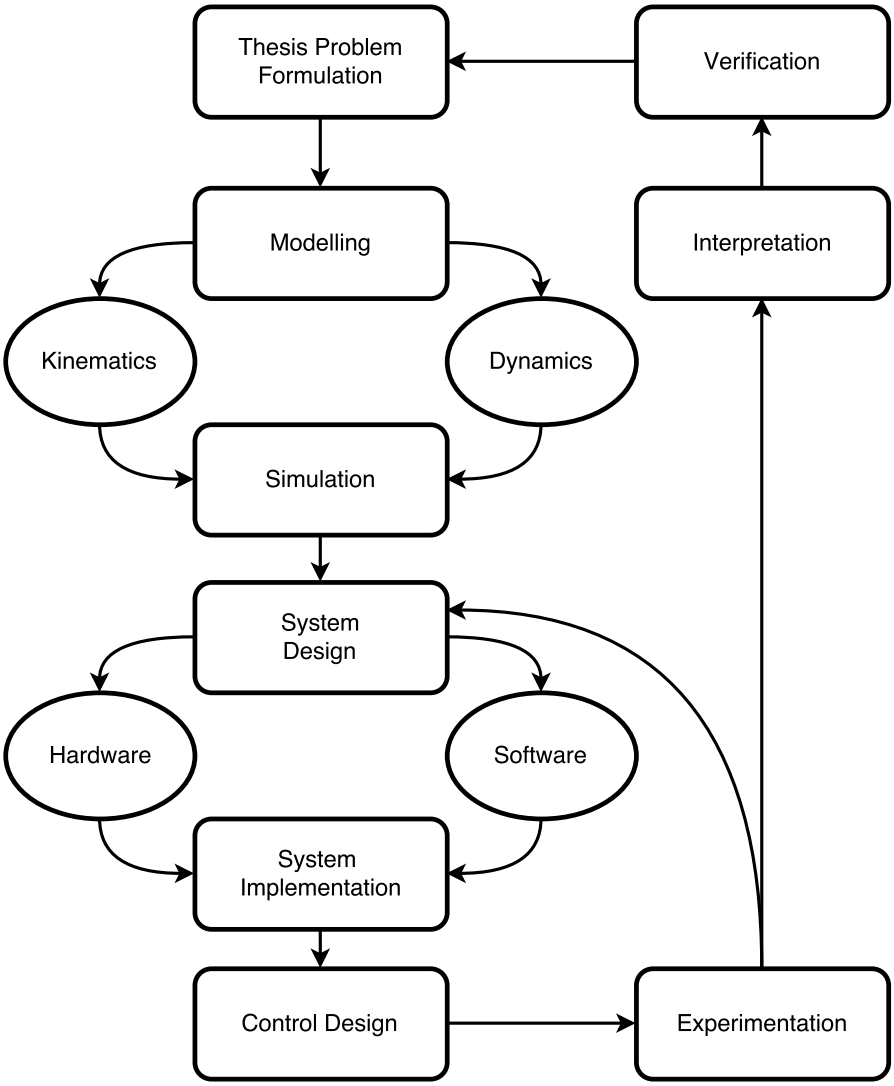


Figure 3.1: Baleka thesis project methodology flow chart.

experiment will be performed to determine the repeatability of jumps, investigating force control fidelity and disturbance rejection.

The motor and motor driver combination will be experimentally tested for correct operation - in the form of response time, set-point tracking, and steady state error.

The final phase of experimentation will be to perform basic trajectory tracking. This will test kinematic fidelity and dynamic performance likely to be experienced in further legged locomotion study.

4 Kinematics

4.1 Geometry

The geometry of the leg was first found in the study in 2006 [16] that investigated a hybrid machine system based on a 5-bar linkage design. This study derived a set of complex kinematic equations to solve for the end effector position given the motor angles. This robot was more of a study in kinematic derivation, and although the same geometry was used, complex kinematic equations were derived that would be difficult to implement on a micro-controller system. The kinematic equations also needed further mapping to the polar coordinate system used in Baleka.

In 2015 a single legged jumping robot was developed in [3] that used the same geometric leg topology - the Baleka design was based off of this robot and the kinematic equations derived in this study were used to implement the virtual model control system for Baleka.

In 2016 a similar leg topology was used in [9] with a three motor 3 DOF geometric design. The virtual model control implemented in this study was adapted and improved for use on Baleka.

The leg linkage length choices that were originally implemented by Ben Bingham based off the design in [3] were validated using a kinematic workspace simulation in this study in section 4.4. The leg linkage lengths of $l_1 = 0.15\ m$ and $l_2 = 0.3\ m$ were chosen.

The foot of Baleka will be referred to often in the design. The foot is the intersection between the lower linkages, l_2, and is the end effector of the robot.

In the design of leg dynamics, a radial set-point is often spoken of rather than a radial position or command. This set-point is a radial distance in the virtual model, further developed in chapter 5, that indicates the length of the leg spring-damper system at rest, measured from the midpoint between the two motors to the foot. A radial offset indicates the difference between the virtual model radial set-point and the actual end effector radial position. With a virtual model control system, you don't command a position, but rather define the virtual model set-points for a certain natural response.

When facing the leg, the rotational set-point of the leg might be specifically spoken of as either the angular set-point (θ) or the arc-length set-point (s), measured anti-clockwise

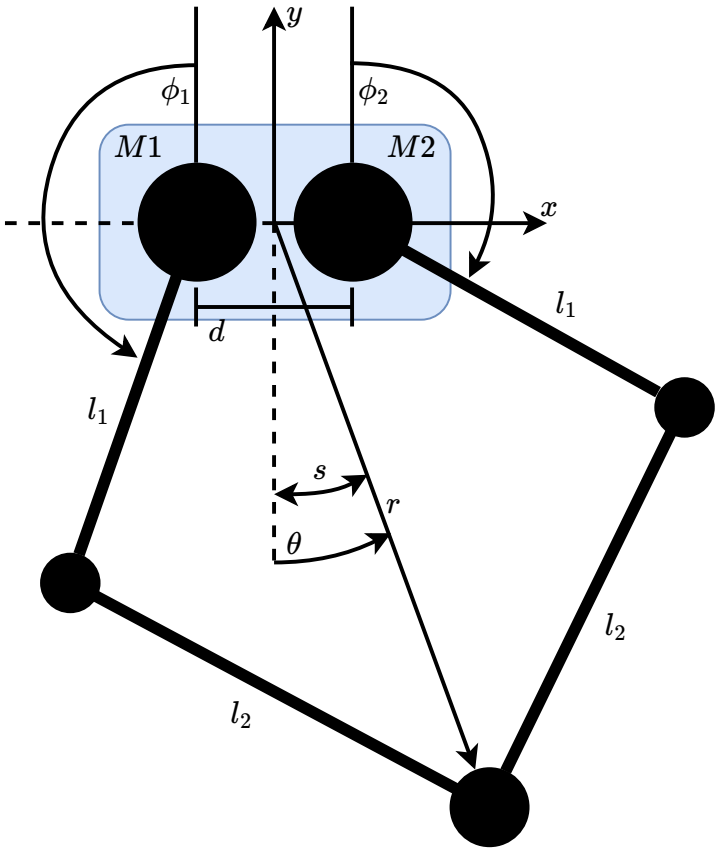


Figure 4.1: Geometric view of leg.

from the negative y-axis.

Both of the motor angles, ϕ_1 and ϕ_2, are measured from the positive y-axis to the first linkage l_1, as shown in fig. 4.1 - this measurement is calibrated by placing the end effector vertically down with linkage pairs parallel and setting the encoder positions to zero counts. The microcontroller then calculates the appropriate motor angles using the motor driver feedback count values.

4.2 Kinematic Equations

The geometry of the leg is fairly complex and the derivation of the kinematic equations equally so. J.M. Duperret and D.E. Koditschek derived the kinematic equations eqs. (4.1) and (4.2) in the study [3].

In this study the assumption was made that the distance d, as seen in fig. 4.1, is zero. This simplifies the derivation of forward and reverse kinematic equations of the leg design by making the leg a 4-bar linkage. These kinematic equations are more easily calculated on board a microcontroller[3], leaving more processing power for other control tasks if needed.

The ease of calculation makes the loss in accuracy acceptable - in practise the simplified kinematic equations worked well with an insignificant calculation time made possible by the STM32F4's on-board floating point unit, and insignificant kinematic distortion.

Forward Kinematic Map

$$f(\phi_1, \phi_2) = \begin{pmatrix} \sqrt{{l_2}^2 - {l_1}^2 \sin\left(\frac{\phi_1}{2} + \frac{\phi_2}{2}\right)^2} - l_1 \cos\left(\frac{\phi_1}{2} + \frac{\phi_2}{2}\right) \\ \frac{\phi_1}{2} - \frac{\phi_2}{2} \end{pmatrix} \tag{4.1}$$

Inverse Kinematic Map

$$g(r, \theta) = \begin{pmatrix} \pi - acos(\frac{r^2 + l_1^2 - l_2^2}{2rl_1}) + \theta \\ \pi - acos(\frac{r^2 + l_1^2 - l_2^2}{2rl_1}) - \theta \end{pmatrix} \tag{4.2}$$

4.3 The Jacobian

The forward kinematic Jacobian is formed by taking partial derivatives of the forward kinematic equation eq. (4.1) as shown in eq. (4.3). In some cases the inverse kinematic Jacobian will be used to implement simulations.

It is used as a mapping from the joint angles ϕ_1 and ϕ_2 to the end effector generalized coordinates r and θ. The Jacobian can be applied in robotic kinematic control to determine joint velocities and forces to achieve a desired force or velocity at the end effector, in this case the leg foot.

Forward Kinematic Jacobian

$$J(\phi_1, \phi_2) = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{X}}\right] \tag{4.3}$$

where $\mathbf{X} = [\phi_1 \ \phi_2]$ and $\mathbf{f} = f(\phi_1, \phi_2)$.

Inverse Kinematic Jacobian

$$J(r,\theta) = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{X}}\right] \tag{4.4}$$

where $\mathbf{X} = [\mathrm{r}\ \theta]$ and $\mathbf{f} = g(r,\theta)$.

4.3.1 Velocity Mapping

The Jacobian can be used to map the motor rotational velocities to the full leg model polar velocity, as in eq. (4.5):

$$v(\dot{r},\dot{\theta}) = Jw(\dot{\phi_1},\dot{\phi_2}) \tag{4.5}$$

4.3.2 Summary

The Jacobian was used for kinematic mapping in three cases:

1. **The mapping from polar foot force to motor torques:** Force Control in section 8.5 using the transpose of forward kinematic Jacobian.
2. **The mapping from motor velocities to polar velocities:** subsection 4.3.1 and section 9.1 using the forward kinematic Jacobian.
3. **The mapping from motor torques to polar foot force for simulation purposes:** Controller Development in chapter 8 using the transpose of inverse kinematic Jacobian.

4.4 Simulation

A kinematic workspace is a visual representation of possible end effector positions in a specific coordinate system. In the case of Baleka, a polar coordinate system was used with the foot of the robot being the end effector. The leg model with relevant coordinates can be seen in fig. 4.1 and will be referred to further in the discussion below.

The kinematic workspace generated by various linkage length combinations was compared in order to properly choose an appropriate configuration.

In order to perform a geometric simulation of all possible foot positions Matlab was used, the code in mention can be seen in listing 8 in appendix D.

A grid of all ϕ_1 and ϕ_2 motor angles was generated. It was assumed that the motor angles would be limited to a range of between 20^o and 180^o. Using the forward kinematic equation from chapter 4 all possible r and θ pairs were generated and subsequently plotted to produce the kinematic workspaces in figs. 4.2 to 4.5.

4.4.1 Data Analysis

The plots were generated with a motor angle resolution of $0.125\ radians$. In reality the motor encoders used had a 500 count resolution and could therefore accurately measure at $0.0125\ radian$ steps. For better visual representation the lower resolution was used.

It can be clearly seen that at the extremes of the radial range the points plotted are more dense. This indicates that a higher resolution movement can be achieved at these points. This is expected due to the more acute joint angles when at these positions. Realistically for a system with mechanical slack this has little useful effect, but theoretically the higher resolution at extended foot positions would be of use for fine tuned launch and landing control were the leg would either be extended or compressed.

There are three points of interest on the kinematic workspace plots:

1. Radial position of greatest angular movement.
2. Minimum achievable radius.
3. Maximum achievable radius.

$l_1 = 5\ cm$ **and** $l_2 = 30\ cm$

For leg linkage lengths of $l_1 = 5\ cm$ and $l_2 = 30\ cm$ in fig. 4.2 the radial position of greatest angular movement is $r = 0.3\ m$. The radius varies from $0.35\ m$ at its greatest down to $0.25\ m$.

The position of greatest angular movement is ideally positioned in the middle of the radial range providing a well spaced workspace.

The radial range of only $10\ cm$ is limiting and not ideal for jumping which requires maximum radial range to ensure the foot is in contact with the ground for as long possible when launching.

$l_1 = 15\ cm$ **and** $l_2 = 30\ cm$

For leg linkage lengths of $l_1 = 15\ cm$ and $l_2 = 30\ cm$ in fig. 4.3 the radial position of greatest angular movement is just above $r = 0.3\ m$. The radius varies from $0.45\ m$ at its greatest down to $0.15\ m$.

The position of greatest angular movement is ideally positioned in the middle of the radial range providing a well spaced workspace.

The radial range of $30\ cm$ is adequate for single leg hopping, but would maybe need to be increased for full body robotic movement.

$l_1 = 30\ cm$ **and** $l_2 = 30\ cm$

For leg linkage lengths of $l_1 = 30\ cm$ and $l_2 = 30\ cm$ in fig. 4.4 the radial position of greatest angular movement is near to zero. The radius varies from $0.6\ m$ at its greatest down to $0\ m$.

The position of greatest angular movement is positioned at near zero which is of little benefit to robotic hopping movement where maximum angular movement is needed when the leg is extended.

The radial range of $60\ cm$ is large, but with the drawbacks mentioned above.

$l_1 > l_2$

As soon as l_1 is greater than l_2 the forward kinematic mapping from chapter 4 produces imaginary values of radius. This can be seen in fig. 4.5 where complex values were not plotted leaving gaps in the kinematic workspace.

In practise it is difficult to properly implement complex calculations on an embedded system and it is better avoided all together.

4.4.2 Summary

The simulation of $l_1 = 15\ cm$ and $l_2 = 30\ cm$ provided the best combination of maximum radial length and range while placing the radial position of greatest angular movement midway between the two extreme limits.

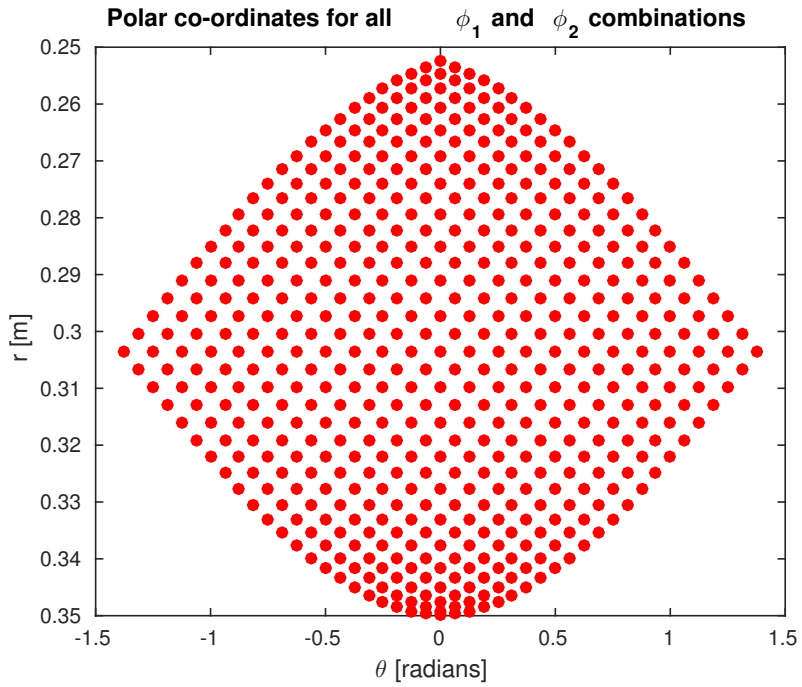


Figure 4.2: Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 5cm$ $l_2 = 30cm$.

A linkage ratio of $\frac{1}{2}$ should be used, with linkage length being decided upon based on motor current requirements where a longer leg produces more torque at the extreme foot positions and will therefore use a higher current.

The motor current simulation with a linkage length of $l_1 = 15\ cm$ and $l_2 = 30\ cm$ was performed in fig. 6.11 - this combination of linkage lengths was found to be ideal for the BLDC motors and motor drivers available.

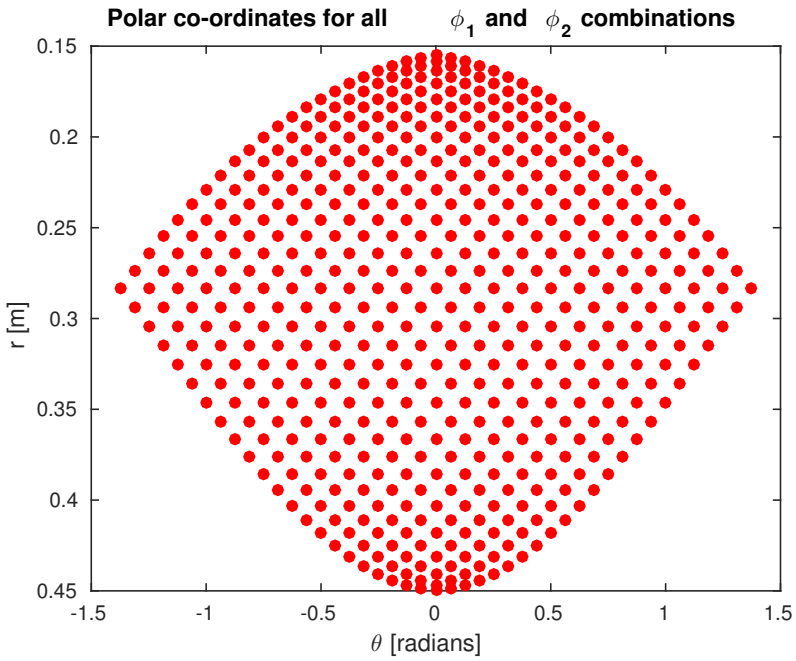


Figure 4.3: Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 15cm$ $l_2 = 30cm$.

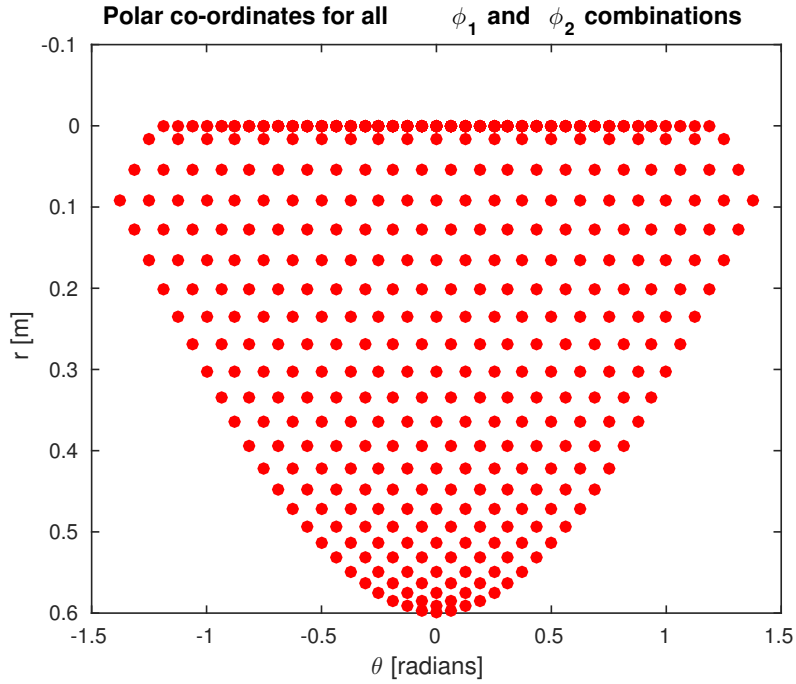


Figure 4.4: Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 30cm$ $l_2 = 30cm$.

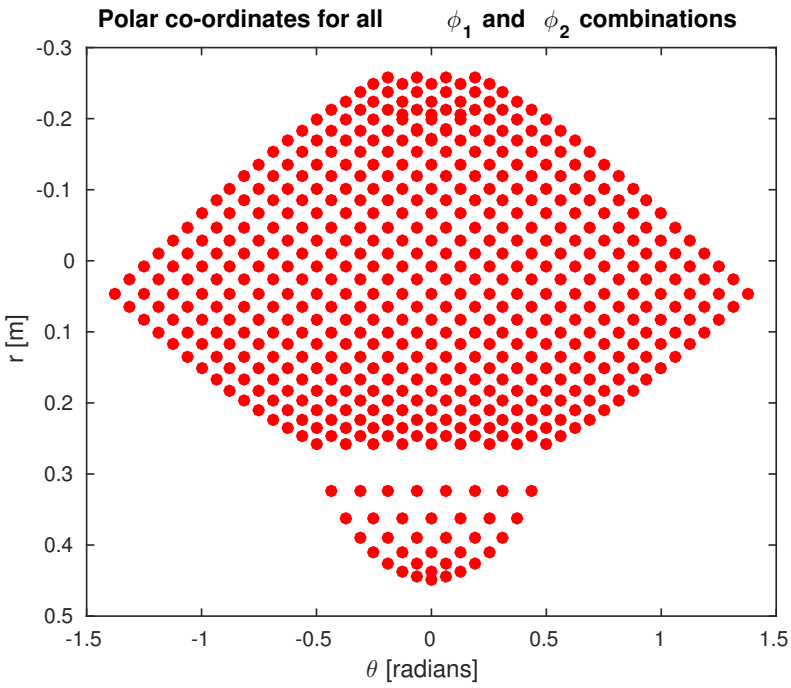


Figure 4.5: Polar co-ordinates generated for all ϕ_1 and ϕ_2 combinations using forward kinematics: $l_1 = 30cm$ $l_2 = 15cm$.

5 Dynamic Modelling

5.1 Virtual Compliance Model

Virtual model control is a control method that uses virtual components to create virtual forces when they interact with a real kinematic robotic model.[15] The virtual components take the form of springs, dampers, and any other mechanical device that alters the dynamics of a robots movement in some way, or any combination of the above.

The virtual model topology is defined in the mathematical sense on a very high level and the expected motion is related to the kinematic model and applied to the robotic model using whatever means of actuation exist.

In the case of Baleka two virtual model topologies were designed and implemented, the full-leg spring-damper virtual model in fig. 5.1 and the joint spring-damper model in fig. 5.2. These two topologies were adapted for the Baleka kinematics from [9].

Two adaptations to the virtual model system in [9] were made:

1. A polar coordinate system was used for the virtual model topology rather than a Cartesian system.
2. A virtual torsional spring was developed for the full-leg case.

Assuming a full-leg virtual model where the foot force is defined around the polar coordinate system, the resulting foot force vector would be decomposed into r and θ components:

$$F = [f_r \ f_\theta]^T \tag{5.1}$$

By using the arc-length radian measure relation, $s = r\theta$, we can decompose the foot force into r and s components:

$$F = [f_r \ f_s]^T \tag{5.2}$$

Defining a generic spring-damper compliance model as derived in [9], we can theoretically place the virtual component on any axis or measurement we wish, provided that it can be determined using the existing kinematic equations and Jacobian. The generic compliance model used in Baleka is as follows:

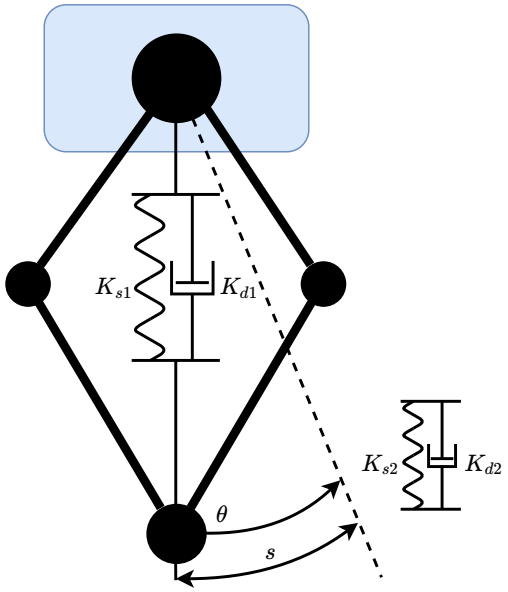


Figure 5.1: Leg spring-damper virtual model.

$$f_a = k_s(a_{fbk} - a_{cmd}) + k_d(\dot{a}_{fbk} - \dot{a}_{cmd}) \tag{5.3}$$

where K_s is the spring constant, K_d is the damping constant, and a and its derivatives are the axis or measurement in question.

For full leg spring-damper control, the virtual component is placed on the polar coordinate of the end effector with respect to the robot body, as shown in fig. 5.1.

Another possibility is to implement two torsional spring-dampers virtually defined around the motor positions. This will be referred to as the joint spring-damper virtual model as shown in fig. 5.2.

It must be noted that before using the unit of radians alongside meters, as in polar coordinates, the radian measure should be normalised to not overpower the radial spring-damper force component. This is further investigated in section 8.7.

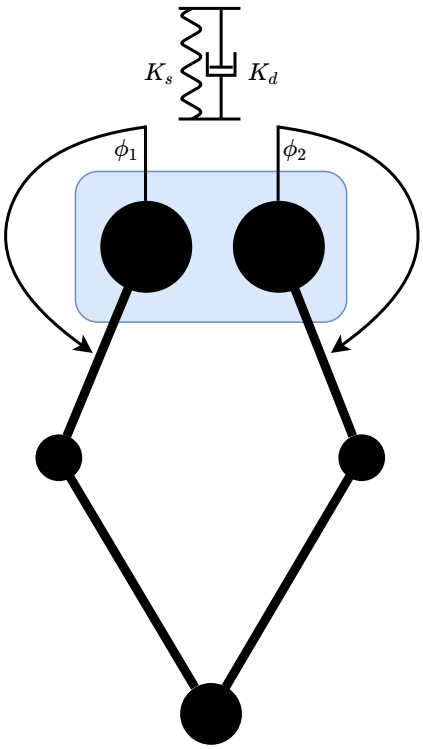


Figure 5.2: Joint spring-damper virtual model.

5.2 Spring-damper Mass Motion

After determining the topology of the spring-damper virtual model, it is necessary to define and develop the commonly used spring-damper mass motion model and apply it to the specific case of the Baleka leg.

Using Newton's second law, the downward force of a mass under acceleration is:

$$F_m = ma = m\ddot{x} \tag{5.4}$$

A spring with spring constant k and a damper with damping constant c have restoring forces as follows:

$$\begin{aligned} F_s &= kx \\ F_d &= c\dot{x} \end{aligned} \tag{5.5}$$

Given the spring-damper mass system in fig. 5.3, free to oscillate, the equation of motion below is derived:

$$\begin{aligned} -m\ddot{x} &= kx + c\dot{x} \\ m\ddot{x} + kx + c\dot{x} &= 0 \end{aligned} \tag{5.6}$$

By setting $x(t) = Ae^{\lambda t}$, the roots of the ODE above are:

$$\lambda_{1,2} = \frac{-c \pm \sqrt{c^2 - 4mk}}{2m} \tag{5.7}$$

Critical Damping

In eq. (5.7) the zero origin crossing defines a critically damped system, and can be defined as follows:

$$\begin{aligned} c^2 - 4mk &= 0 \\ c_{critical} &= 2\sqrt{mk} \end{aligned} \tag{5.8}$$

Damping Ratio

The damping ratio is the ratio of the implemented damping constant to the critical damping constant, and can be defined as follows:

$$\zeta = \frac{c}{c_{critical}} \tag{5.9}$$

The equation eq. (5.7) can be restated in terms of the damping ratio as follows:

$$\lambda_{1,2} = (-\zeta \pm \sqrt{\zeta^2 - 1})\omega_0 \tag{5.10}$$

Under, Over and Critical Damping

For the spring-damper system to be under, over or critically damped, the following conditions must be met:

- Under: $\zeta < 1$ with imaginary roots
- Over: $\zeta > 1$
- Critical: $\zeta = 1$

For the robotic leg, ideally we want an under damped system when landing - this ensures some of the shock of landing is dissipated by the spring damper oscillations. If the system is critically damped the leg mechanics will experience a lot of stress and the motor drivers are unlikely to be able to supply the current needed.

Experimental Damping

Experimentally the damping ratio ζ can be determined using the logarithmic decrement calculation. The logarithmic decrement is found by taking the natural logarithm of the ratio of amplitudes of two waveform peaks - this is then equated as follows:

$$\begin{aligned} \delta_n &= ln(\frac{x_a}{x_b}) \\ \zeta &= \frac{\delta_n}{\sqrt{(2\pi n)^2 + \delta_n^2}} \end{aligned} \tag{5.11}$$

where $n = b - a$ determines the number of peaks between measurements.

This equation was used in testing, as in section 9.3, to determine the damping ratio of the system to validate the virtual model.

Natural Frequency

The natural frequency, ω_0, is the frequency the undamped system will oscillate at if given an initial x offset and left to freely oscillate:

$$\omega_0 = \sqrt{\frac{k}{m}}\ [rad/s] \tag{5.12}$$

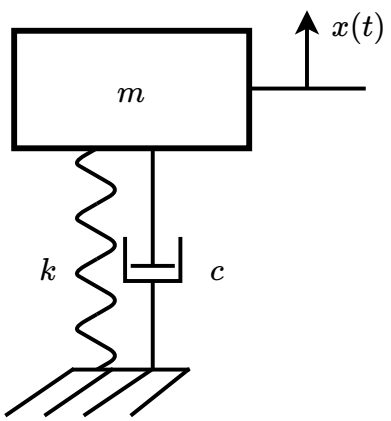


Figure 5.3: Spring-damper mass model.

The damped natural frequency is the same as above, but with the damping constant c not equal to zero:

$$\omega_d = \sqrt{1-\zeta^2}\omega_0 \; [rad/s] \tag{5.13}$$

5.3 Leg Spring-damper Model

The leg, when viewed as a spring-damper mass system in free space, has a mass of approximately $0.5\ kg$ which is the mass of the leg linkages.

On impact and launching the leg spring-damper mass system is in contact with the ground and therefore the mass of the plate and motors acts on the entire system with a mass of approximately $2.2\ kg$.

5.3.1 Launch Energy

In order to launch the leg a set height of $0.4\ m$, the spring potential energy needs to be efficiently transferred into kinetic energy. This kinetic energy will launch the leg to a height with a corresponding potential energy. To calculate the spring constant needed to complete this jump assuming $80\ \%$ mechanical efficiency, equation eq. (5.14) is used. This is based on the principal of conservation of energy.

From eq. (5.16), $E_{ps} = \frac{1}{2}k\Delta x^2$. Using an initial radial set-point of $0.3\ m$ and decompressing the spring to a radial set-point of $0.4\ m$ we get a δx value of $-0.1\ m$.

$$\begin{aligned} E_p &= E_{ps} \\ mgh &= \frac{1}{2}k\Delta x^2 \\ k &= \frac{2mgh}{\Delta x^2} = \frac{2 \times 2.2 \times 9.81 \times 0.4}{0.1^2} = 1726.6\ N/m \end{aligned} \tag{5.14}$$

The calculated spring constant of $1726.6\ N/m$ was used during the jump tests.

5.3.2 Impact Energy

When the leg is dropped on the linear guide from its maximum height of $0.5\ m$ the potential energy that needs to be absorbed by the spring-damper is $10.791\ J$ as calculated in eq. (5.15).

$$\begin{aligned} E_p &= mgh \\ E_p &= 2.2 \times 9.81 \times 0.5 = 10.791\ J \end{aligned} \tag{5.15}$$

The spring-damper system should absorb all this energy with the leg depressed to a radial set-point of at most $0.3\ m$, to insure the body does not impact the ground. The majority of the impact energy will be absorbed as spring potential energy with the dynamics being changed slightly by the damper kinetic energy as seen in the spring-damper drop tests in fig. 9.4. The spring potential energy and damper kinetic energy are shown in eq. (5.16).

$$\begin{aligned} E_{ps} &= \frac{1}{2}kx^2 \\ E_{kd} &= \frac{1}{2}c\dot{x}^2 \end{aligned} \tag{5.16}$$

The velocity of the leg in free fall is approximately $2\ m/s$ found by performing drop tests and determining the number of video frames that it took for the leg to drop $0.4\ m$, which was approximately 5. Using eq. (5.17) the velocity was found.

$$v_{final} = \frac{height}{frames \times \frac{1}{fps}} = \frac{0.4}{5 \times 0.04} = 2\ m/s \tag{5.17}$$

A theoretical value for the spring constant k can be found by using conservation of energy as seen in eq. (5.18). A damping constant of $5\ N/(m/s)$ was assumed as

determined through experimentation in fig. 9.4.

$$\begin{aligned} E_p &= E_{ps} + E_{kd} \\ mgh &= \frac{1}{2}kx^2 + \frac{1}{2}c\dot{x}^2 \\ k &= \frac{2(mgh - \frac{1}{2}c\dot{x}^2)}{x^2} = \frac{2(10.791 - \frac{1}{2} \times 5 \times 2^2)}{(0.35 - 0.3)^2} = 632.8\ N/m \end{aligned} \tag{5.18}$$

This spring-constant choice was practically confirmed during the spring-damper drop tests of section 9.4.

6 Hardware Design

6.1 Original Leg Design

6.1.1 Leg Hip

The original leg 'hip' was designed by Ben Bingham in 2016 in completion of his undergraduate vacation work as seen in fig. 6.1.

The 'hip' was constructed of 6 mm perspex sheet in a box design with metal L connectors to join the sheets securely. The design of the 'hip' allowed the motor drivers as well as the microcontroller to be mounted on the body, with space provided for an extra leg for future two-legged movement.

6.1.2 Leg Guide

The guiding system consisted of two parallel steel rods with ball bearings mounted on the 'hip'. The ball bearings were mounted using a 3D printed holder, with one for each circular bearing.

6.1.3 Limitations

The leg mounting plate went through three design iterations after the original 'hip' design before the final design was created, as seen in fig. 6.4c.

The original 'hip' design had the following mechanical design flaws:

1. The 6 mm perspex box construction with on-board microncontroller and motor drivers was too heavy for efficient jumping action when compared to similar designs like [3] (1.3 kg), [9] (2.5 kg), [6] (4.2 kg) where there is a high leg torque to mass ratio.
2. The ball bearings are particularly heavy.

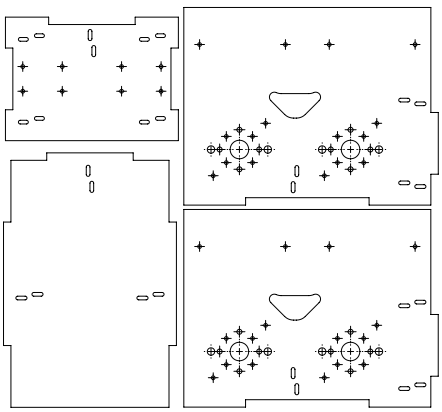


Figure 6.1: Original 'hip' design by Ben Bingham, 2016.

3. The mounting of the leg guide places a significant torque in all three cartesian coordinates being off-center from the center of mass.
4. The design of the leg guide requires the two steel rods to be perfectly parallel to remove resistance to movement, which is difficult to achieve practically.

These problems were accounted for by replacing the original 'hip' with a rigid aluminium mounting plate with off-board microcontroller and motor drivers. The leg guide consisting of parallel steel rods and ball bearings was replaced with a linear guide as seen in fig. 6.7.

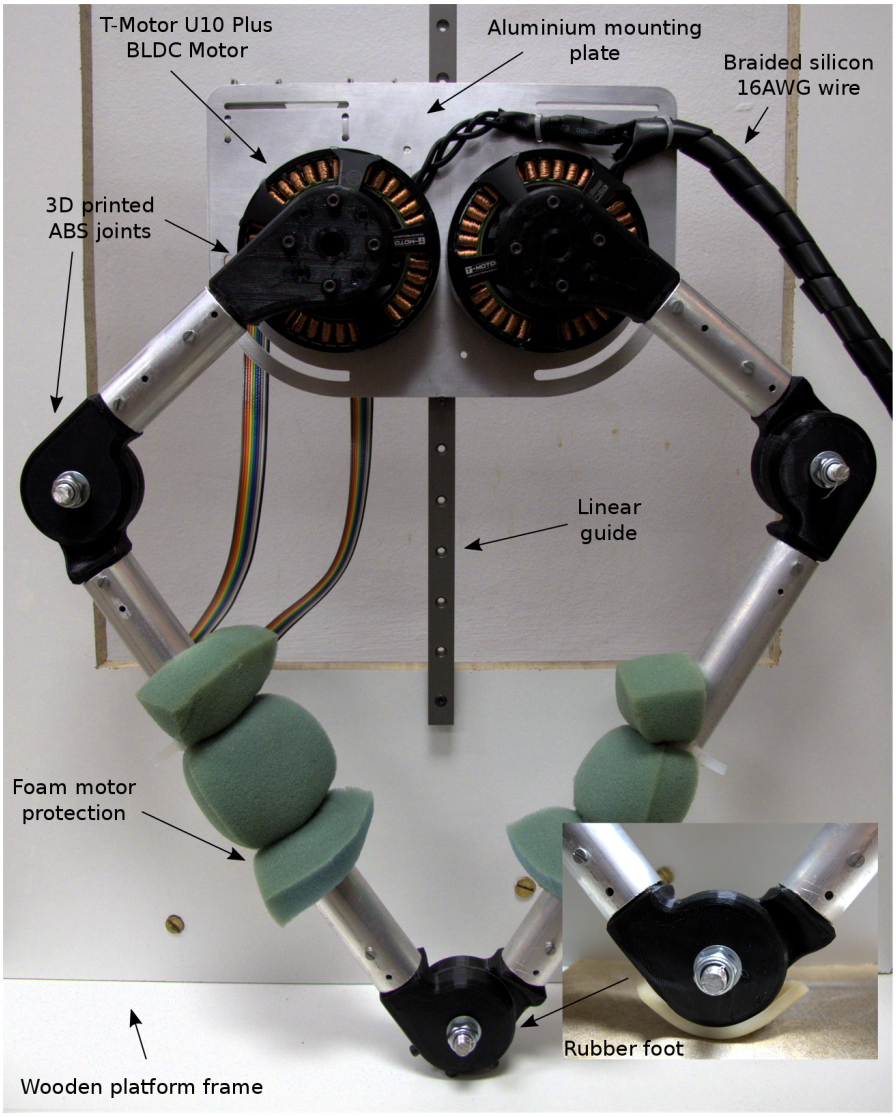


Figure 6.2: Final leg design mounted to platform and linear guide: front.

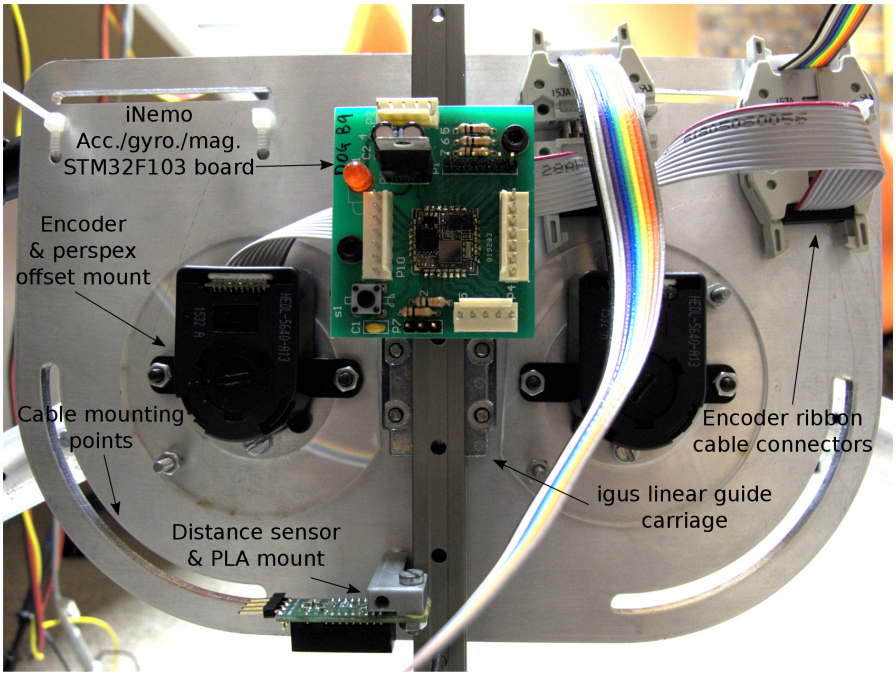


Figure 6.3: Final leg design mounted to platform and linear guide: back.

6.2 Mechanics and Construction

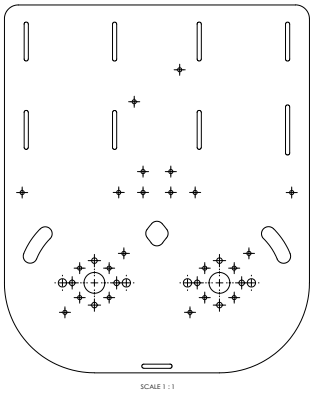
6.2.1 Aluminium Mounting Plate Design

The original mounting box described in section 6.1 was replaced with a single mounting plate.

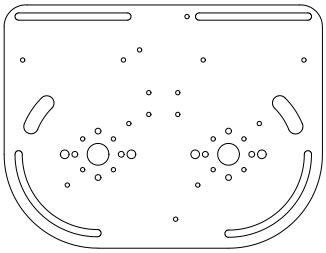
In order to reduce the weight of the platform and to prevent damage to components during jump tests, the motor drivers and microcontroller were mounted off-board on a separate mounting plate - the final design of which can be seen in fig. 6.5.

The plate that the motors and encoders mount to went through several iterations before the final design was used. A few major factors were considered in the design process, both for the motor mount and motor driver mount:

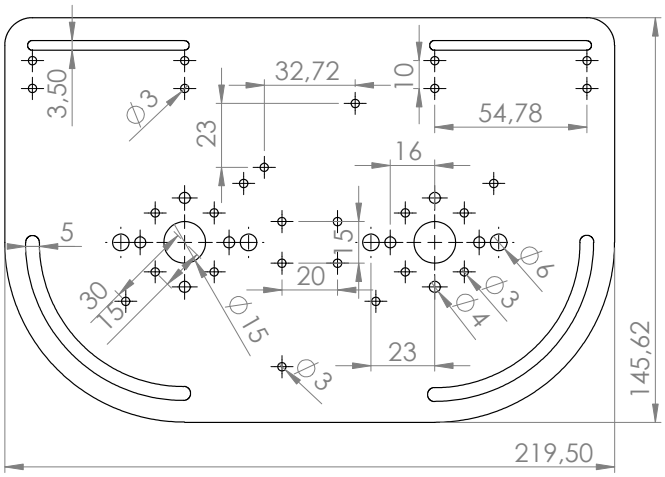
1. Perspex material was too flexible, especially considering the close tolerances of the motor mounts - aluminium was used instead of perspex as a more rigid material.
2. The motors when used in a high torque relatively static environment reach temperatures of close to 100^oC - this heat was noticeably better dissipated by aluminium.
3. The motor drivers perform better and are less likely to thermally cut-out if mounted



(a) CAD mounting plate V1.



(b) CAD mounting plate V3.1.3.



(c) CAD mounting plate final design.

Figure 6.4: Leg mounting plate iterations.

on aluminium.

4. The center of mass of the robotic leg body was calculated and the linear guide mount was placed as close as possible to this point. This ensured as little torque as possible was placed on the linear guide which would lead to frictional losses and ware.

5. The iNemo accelerometer, gyroscope, magnetometer combination was placed as close as possible to the center of mass to ensure proper readings were achieved which accurately represented the robot dynamics.

6. Excess weight was cut wherever possible, most noticeably so above the motor mounting points.

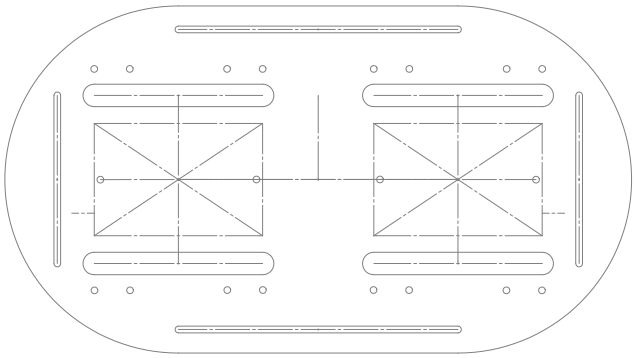


Figure 6.5: Motor driver interface mounting plate.

6.2.2 Leg Linkage and Foot Design

The 4-bar linkage design of the leg was originally constructed by Ben Bingham in 2016 for completion of his undergraduate engineering vacation work in the mechatronics lab. The leg was constructed as follows:

- Three sets of rotational joints make up the linkage system. The joints were 3D printed using ABS plastic and used 3 mm screws to connect to the aluminium leg sections.
- 8 mm loctite nut, bolt and washer combinations connected the joint components with perspex discs to reduce friction between the joints.
- The aluminium leg sections were constructed of 25 mm diameter tubing to form a leg of 0.15 m and 0.3 m sections including the joints.

The leg had a number of design flaws that will be improved upon in the implementation of the Mechatronics Lab Cheetah project with the leg being redesigned by Callen Fisher. These issues are listed below:

1. The joints provide significant friction when under torque outside of the two degrees of freedom of the leg.
2. The resistance to movement provided by the joints in normal operation is directly related to the amount of torque tightening on the nut and bolt.
3. The nuts and bolts slowly work themselves loose under normal operation due to vibrations and impact.

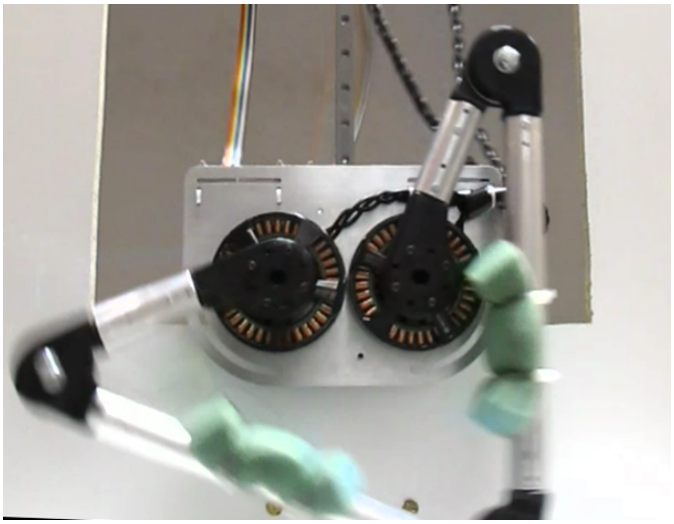


Figure 6.6: Leg foot lateral slipping.

4. The motor joints, being off-set from the motor shafts, place significant torque on the motor shafts under impact.

Without the proper material to provide friction during foot placement on the ground, lateral slipping occurs as seen in fig. 6.6. To improve the grip of the foot, the following design choices were made:

1. 80 to 120 grit sandpaper was placed on the platform to emulate the material encountered by a Cheetah in normal operation, namely dirt and gravel.
2. A rubber mould was made, by mixing and setting rubber components, before being attached to the foot using 3 mm hex screws.

The combination of sandpaper and rubber foot stopped lateral slipping from occurring.

6.2.3 Testing Platform

The testing platform was designed to be limited to a vertical axis of movement. This allows robust testing of the hopping capabilities of Baleka. In future experimentation a rotational hinged platform can be used to test forward hopping trajectories.

The original platform design used two parallel $16\ mm$ tool steel rods with ball bearings attached to the leg platform. This not only added extra weight, but also made it difficult to properly mount the rods parallel to avoid friction.

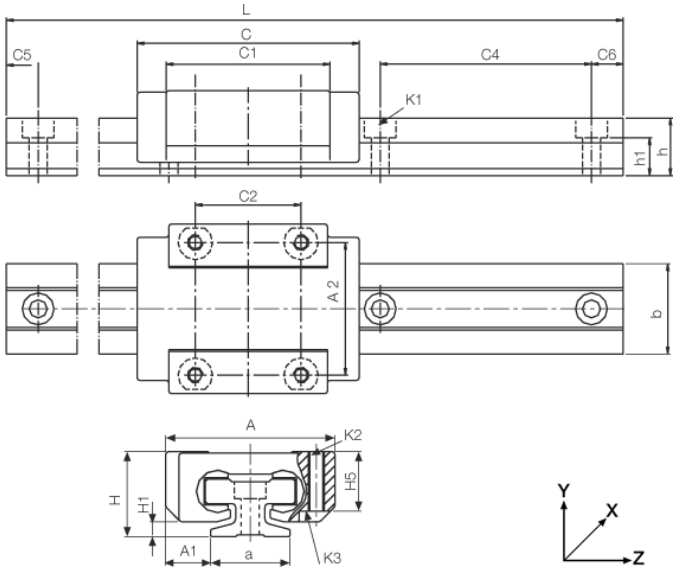


Figure 6.7: igus DryLin T - Low-profile linear guide.

The original setup was replaced with an igus linear guide as seen in fig. 6.7, specifically the TW-04-12 DryLin T miniature slide carriage along with the appropriate rail.

A rail of $0.6\ m$ was used which resulted in a potential hopping height of $0.4\ m$. This height proved to be adequate given the motor driver's $60\ A$ current limit which in practise resulted in a maximum hopping height of just under $0.4\ m$. Further hopping experiments performed with the linear guide platform can be seen in chapter 9.

The linear guide rail was mounted on a wooden frame with a heavy $3\ kg$ wooden counterweight on the base. The choice of wood for the frame was to reduce vibrations and for ease of construction. The heavy wooden base ensured the platform was stable during jump experimentation.

Despite specifications of the rail not requiring lubrication, a basic oil based lubricant was used and significantly reduced friction caused by forward rotational torque of the carriage on the linear guide rail.

To simulate the environment and platform used in the hopping experiments a CAD assembly was generated, as seen in fig. 6.8, as well as a virtual world Matlab model. This allowed the leg to be manipulated and dropped on the platform to see how it would behave. The resulting tests ensured the testing platform would have a good chance of performing well in real life experiments. Given that the linear guide system was a significant capital investment this was necessary.

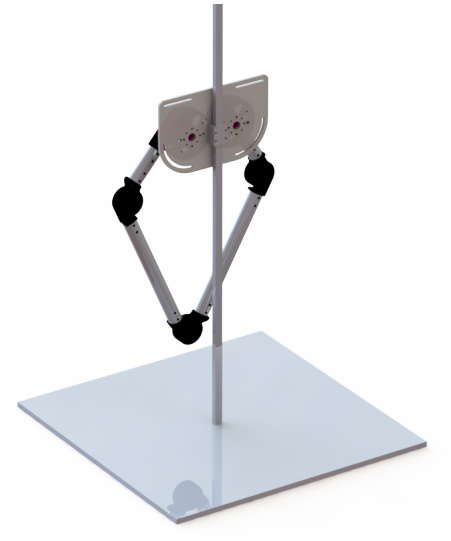


Figure 6.8: Linear guide mounted leg model (CAD Solidworks assembly).

6.3 Mass Distribution

The calculation of the mass and center of mass (COM) was critical for making the following design choices:

- Placement of the iNemo sensor board as close to the COM as physically possible (seen in fig. 6.3).
- Mounting of the linear guide carriage as close to the COM to minimize torque and stress placed on guide rail during jumping, as well as interference to jump dynamics (seen in fig. 6.3).
- Calculation of jump dynamics as in chapter 5.

The mass of individual leg components was first calculated manually using a scale, the log of which can be seen in table 6.1. The total mass of $2.2\ kg$ from this log was used in all dynamics, modelling and simulation calculations.

The iNemo and linear guide carriage are of minimal mass and were not included in the manual mass calculation.

The center of mass was simulated in Solidworks. The material properties of each component were first configured appropriately.

For simulation the leg radial set-point was set to $0.3\ m$ - this is the value of the leg radius during flight, freefall, impact, and compliant landing phases of jumping as seen in

Item	Mass (g)	No.	Total Mass (g)
T-Motor U10 Plus	500	2	1000
Servo drive	123.9	2	247.8
Servo drive mounting card	50.7	2	101.4
Leg	500	1	500
Plate	350	1	350
Total			**2199.2**

Table 6.1: Leg component mass.

section 9.5. This ensures the COM calculation is accurate for the majority of the jump.

In reality the center of mass (COM) moves negligibly with the foot position. The reason for this can be seen in fig. 6.9 where the mass distribution simulation shows the majority of the mass is concentrated around the COM, in red. The colour legend in table 6.2 explains the colour code and shows the simulated mass of each component.

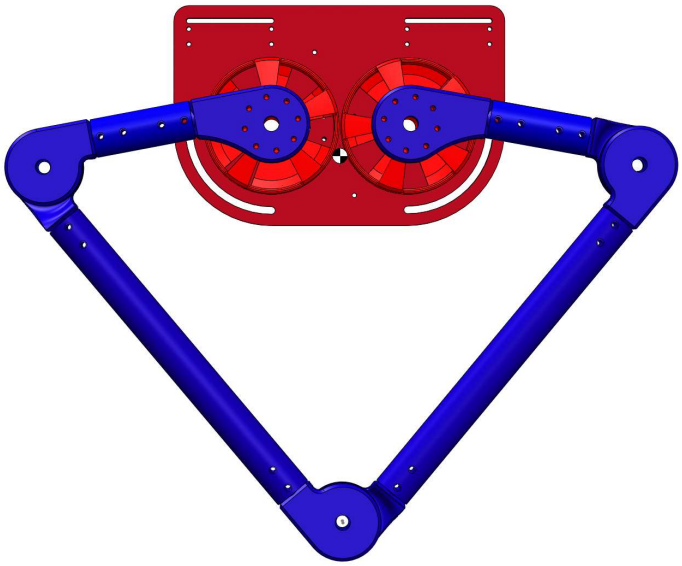
In fig. 6.9 the COM is clearly shown as a black and white checker board circle with the distance from critical points shown on the second figure. The linear guide carriage is mounted just above this point along with the iNemo mounted with spacers above the carriage - it was not possible, due to the motor mounts, to place these components physically closer.

6.3.1 Limitations

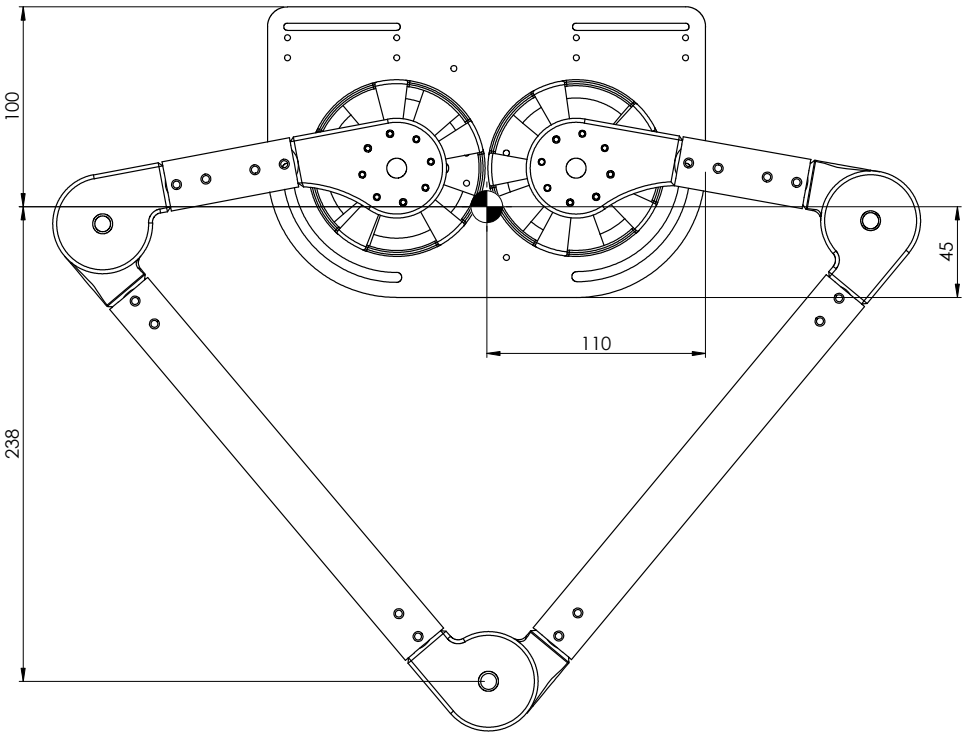
The COM was only calculated in the two dimensional case. A significant torque was found to exist in the third dimension during experimentation that caused the leg to twist forward on the mount. This provided significant friction on the linear guide rail. In future designs the COM should be calculated in the third dimension and the motors possibly mounted coaxially on either side of the aluminium plate - this would have minimized the forward torque as the motors are the components with the most mass.

6.3.2 Actuator to Body Mass Ratio

In the study [2] it is stated that as large a proportion of the mass budget as possible should be dedicated to the actuator, in this case the T-Motor U10 Plus BLDC motor. Table 10.1 shows that the Baleka robotic leg has the second highest actuator to body mass ratio,



(a) Mass distribution simulation.



(b) Center of mass of leg assembly.

Figure 6.9: Mass distribution of leg assembly.

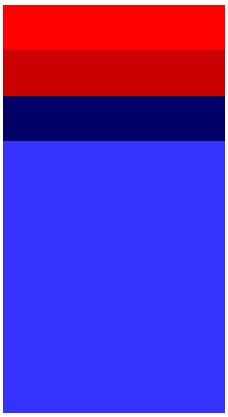
Colour legend	Component	No.	Mass (g)
	T-Motor U10 Plus	2	424.56
	Mounting Plate	1	378.86
	Linear Guide Carriage	1	100.37
	ABS Motor Joint	2	50.55
	Long Linkage	2	46.40
	Joint	5	39.83
	Foot Joint	1	39.63
	Short Linkage	2	12.81
	Washer Bearing	3	2.41
Total:			**1793.88**

Table 6.2: Solidworks leg assembly mass distribution.

second only to the GOAT leg. This performance rating is perhaps biased as the Baleka leg does not have the mass of on-board motor drivers and control.

6.4 Mechanical Impedance

6.4.1 Linear Guide

In section 9.5 a set of jump-tests were performed. By extracting the acceleration data from the video frames of a jump a plot of acceleration vs. time was found as seen in fig. 9.15. During the free-fall phase of the jump the mean acceleration was calculated as $-9.51\ m/s^2$. Using this acceleration and the known acceleration due to gravity, the coefficient of kinetic friction for the linear guide, μ_k, can be calculated as in eq. (6.1).

$$\begin{aligned} F_k &= F_n \mu_k \\ \mu_k &= \frac{F_k}{F_n} \\ \mu_k &= \frac{|m\ddot{x} - mg|}{|mg|} \\ \mu_k &= \frac{|2.2 \times -9.51 - 2.2 \times -9.81|}{|2.2 \times -9.81|} \\ \mu_k &= 0.031 \end{aligned} \tag{6.1}$$

This value for kinetic friction is minimal and in practise was assumed to have an

insignificant effect on jump dynamics. If a more precise mechanical system was developed and jump height control was implemented, then the kinetic friction could be taken into account. In experimentation the jump height was not consistently controllable due to mechanical joint slack.

6.4.2 Leg and Joints

The friction, and to a limited extent the inertial load, of the leg and joints was accounted for during the calibration process for the torque constant K_t in subsection 6.6.3. By using force control and the calibration process the joint frictional force that needs to be overcome is indirectly included in the control model.

Due to mechanical joint slack and an imprecise mechanical system, as discussed previously, the inertial and precise frictional load of the leg is difficult to accurately calculate - because of this it is better to experimentally account for all these factors during calibration.

6.5 Electronics and Communication

6.5.1 Accelerometer and Gyroscope

The iNemo board designed and built by Callen Fisher during his masters studies, consisting of a STM32F1 microcontroller with on-board accelerometer, gyroscope and magnetometer, was to be used to measure acceleration data for force measurements and jump phase control.

Due to time constraints acceleration data was not used or found to be necessary for basic hopping control. Provision was made for mounting the board as close to the center of gravity of the robot as was physically possible, as can be seen in fig. 6.3 where the final mounting setup is shown.

The iNemo board was mounted approximately $4\ cm$ from the center of gravity directly over the linear guide. This was achieved by using nylon spacers and the included $3\ mm$ mounting points on the board.

The necessary peripheral configuration and data processing can be easily integrated into the embedded system communication protocol developed in chapter 7 in future. The

GitHub page for the iNemo development board can be seen at `https://github.com/Callen-Fisher/INEMO-development-board`.

6.5.2 Distance Sensor

A distance sensor was mounted to the base of the mounting plate, as seen in fig. 6.3. This provided feedback of the height of the leg's center of mass above the ground.

The leg height was used for height control as well as flight phase determination.

An infra-red distance sensor was chosen with a narrow beam width - this ensures there is minimal reflection off surrounding objects that could interfere with distance readings. The beam reflects off the surface of the ground and a time-of-flight calculation is used to determine distance. Infra-red is open to possible interference from surrounding fluorescent light sources, and this can be accounted for by using it in an area out of direct line of site of light sources. Infra-red was chosen because it is cheaper than an equivalent laser distance sensor.

The Pololu Carrier with Sharp GP2Y0A60SZLF Analog Distance Sensor was used. It requires a $3\ V$ voltage source which can be supplied directly from the micrcontroller which runs off the same voltage.

The distance sensor outputs an analog signal which is related to the height. This analog signal is fed into the ADC of the microcontroller and using a linear relationship is mapped to the distance.

The Pololu distance sensor has a range of $10 - 150\ cm$, which is more than enough for the intended hopping height of $20 - 50\ cm$ due to the testing rig limits.

The height sensor was calibrated by setting it at $0.1\ m$ from a surface and using a scaling factor to adjust the linear relation between distance and voltage until an accurate measurement was achieved.

A 3D printed carrier, a render of which can be seen in fig. 6.10, was designed for the distance sensor which offset the sensor from the mounting plate and placed it in a central location above the linear guide.

6.5.3 Microcontroller

The microcontroller had to meet the following specifications:

Figure 6.10: 3D printed PLA distance sensor mount.

- 4 x UART Ports
- 1 x ADC
- 1 x Floating point unit
- DMA Capabilities
- USB Debugging
- 5V tolerant UART ports

Being familiar with the STM32 series of microcontrollers, a STM32F4 board was chosen and met the above specifications with two additional USART ports for future peripheral needs.

6.6 Motors and Drivers

6.6.1 Driver Selection

The most important factor when choosing a motor driver for dynamic hopping control is the peak current specification. During the launch phase a current impulse will be used to transfer maximum energy to the flight phase.

In fig. 6.11 the kinematic workspace developed in section 4.4 was combined with a virtual compliance control simulation to determine what the theoretical maximum current requirement would be.

A heat map for motor 1 and motor 2 current draw can be seen, with a maximum current magnitude of $52.3\ A$. This simulation was performed with a nominal virtual spring configuration of:

- $K_{s1} = 300\ N/m$
- $K_{s2} = 30\ N/m$

In order to account for the extra current draw when using damping, $60\ A$ was chosen as the motor driver specification.

The peak current specification chosen was adequate for the platform in use, but for achieving higher and more fine tuned jump control a higher peak current is needed. This is further investigated in section 9.5.

To achieve a controller sampling frequency of $200\ Hz$ a high speed and robust communication protocol is needed. RS-485 was chosen as the preferred protocol for the following reasons:

1. Robust to noise interference from BLDC motors.
2. Can support high speed data rates up to $1\ MBaud$.
3. Easy to implement and readily available on off-the-shelf microcontrollers.

In practise the motor drivers caused a communication bottle neck due to the time taken for the controllers to respond to control packets sent - ideally the motor driver should be able to receive, process and reply to control packets without a significant delay. This is further investigated in chapter 7.

The specifications determined above were met by the AMC servo drive and mounting card seen in fig. 6.12.

6.6.2 Motor Selection

A brushless DC motor (BLDC), by definition, has few parts such as brushes that will degrade over time or provide resistance to movement. This is advantageous in robotic systems were repeated high torque operations will be performed.

Due to the recent popularity of BLDC motors in use on quadcopter and other hobby platforms, high performance motors have become more accessible.

In the case of a direct drive robotic platform, performance is measured by the amount of torque the motor can supply for a given current under load.

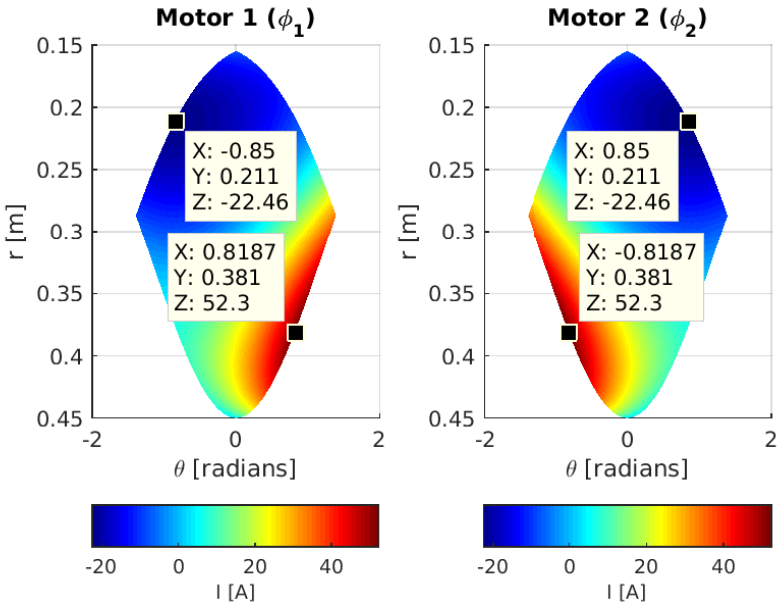
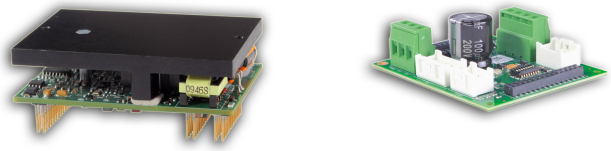


Figure 6.11: Active compliance motor current requirements.



(a) AMC DigiFlex Performance Servo Drive.

(b) AMC DigiFlex Performance Servo Drive mounting card.

Figure 6.12: AMC Servo Drive and Mounting Card.

In the study by [9] various COTS (commercial off the shelf) motors were compared using the thermal specific torque as a performance measure. The T-Motor U10 Plus was found to have the highest thermal specific torque at $0.42\frac{Nm}{kgC^o}$ at $r_{gap} = 40mm$ [9]. When compared to the custom made MIT Cheetah motors at $0.71\frac{Nm}{kgC^o}$ at $r_{gap} = 49mm$ found in [6] they perform favourably.

Figure 6.13 visually shows the relationship between torque, current and speed and places the requirements of the Baleka leg on the plot in blue and the specifications of the T-Motor U10 Plus in red. The T-Motor U10 Plus has the following specifications:

- KV rating: 80
- Shaft diameter: 15 mm
- Weight: 500 g
- No. of Cells (Lipo): 6-14s
- Max Continuous current(A): 33 A
- Max Continuous Power(W): 1500 W
- Internal resistance: $95\ m\Omega$

The KV rating of 80 indicates an unloaded speed of $80\ rpm$ per volt. This is on the low end of the scale and generally means the motor can achieve a higher torque at low speed than a higher KV rated motor.

The ideal motor for the robotic platform would achieve high torque, low current draw and a high torque for low speed operation as shown in blue in fig. 6.13.

The T-Motor U10 Plus sits somewhere in the middle of the performance plot, allowing relatively high speed operation using a 10s battery with adequate torque and current draw specification. Two 5s LiPos in series were used to supply the motors, providing approximately $40\ V$ and a theoretical maximum speed of $3200\ rpm$.

The T-Motor U10 Plus was chosen for the reasons above, as well as the fact that it is a well known and used motor in direct drive robotic projects, being featured in [3] and [9] among others. This is beneficial as they are well documented and it allows us to compare the relative performance between these robotic leg platforms.

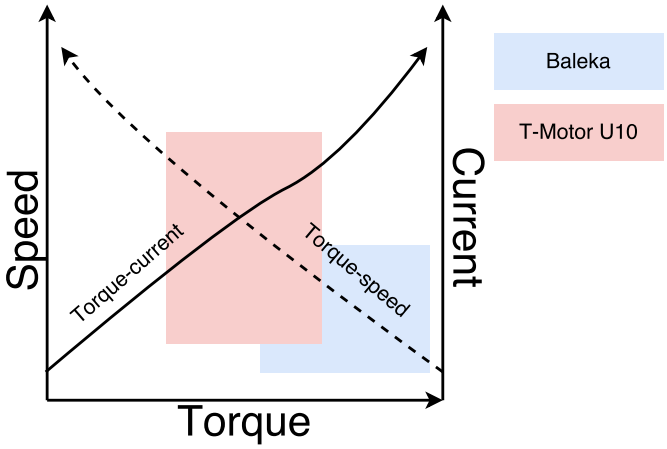


Figure 6.13: Motor performance requirements.

Figure 6.14: T-Motor U10 Plus Brushless DC Motor.

6.6.3 Motor Model Calculations

Experimental Calculation of K_t and K_e

The motor torque constant, K_t, was calculated using the torque current relation $\tau = K_t I$. The leg was modelled as a virtual spring-damper system, as seen in fig. 5.1.

The spring constant, K_{s1}, was set to $200\ [N/m]$, and the damping and torsional spring-damping constants were set to zero. K_t was tuned until the theoretical foot force matched the practical foot force measured via a load cell. The leg was fixed at a set height imposing a radial offset on the virtual spring-damper system.

For a spring constant of $200\ [N/m]$ and a radial offset of $0.15\ m$ a theoretical foot force of $K_{s1}\Delta r = 30\ N$ was expected. A mass of approximately $3\ kg$ was measured with $K_t = 0.08\ [Nm/A]$ set in the virtual leg model controller, resulting in a foot force of $3\ kg \times 9.81\ m/s^2 = 29.43\ [N]$.

The study in [9], using the same T-Motor U10 Plus motors, calculated a torque constant of $K_t = 0.072\ [Nm/A]$. This confirms the experimental results obtained above.

For an ideal motor at a constant operating point, K_e will equal K_t, as shown in eq. (6.2).

$$
\begin{aligned}
&V_t = K_e\omega_m + IR_m \\
&\tau_m = K_t I \\
&P_{elec.} = V_t I = K_e\omega_m I + I^2 R \\
&P_{mech.} = \tau_m\omega_m = K_t I\omega_m \\
&P_{loss.} = I^2 R_m \\
&P_{elec.} = P_{mech.} + P_{loss.} \\
&\therefore K_e\ [V/rad/s] = K_t\ [Nm/A] = 0.08
\end{aligned}
\tag{6.2}
$$

Further force calibration and fidelity tests were performed in section 9.2.

Calculation of R_m and L_m

The resistance and inductance of the 3 phase windings of the motor were calculated using a lab multimeter to be $R_m = 47.5\ m\Omega$ and $L_m = 35\ \mu H$ respectively.

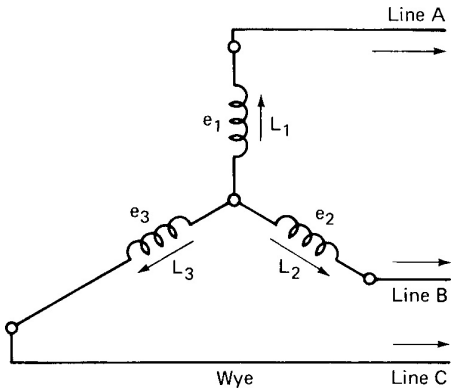


Figure 6.15: WYE connected BLDC motor windings.

Brushless DC motor windings are usually connected in WYE formation, as seen in fig. 6.15. This means the measured values for resistance and inductance were line-to-line values and had to be divided by two to get the per phase values above.

Calculation of J_m

In order to calculate the moment of inertia of the motor, J_m, the ratio of acceleration torque to acceleration to steady state needs to be found. By commanding a DC equivalent current input of $1\ A$ and measuring the time taken to reach a steady state velocity, eq. (6.3) can be used to calculate J_m. The velocity vs. time plot used can be seen in fig. 6.16.

$$\begin{aligned} J_m &= \frac{T_{acc.}[N/m]}{a[m/s^2]} \\ &= \frac{IK_t}{a} \\ &= \frac{IK_t}{\frac{\Delta v}{\Delta t}}\ [kg/m^2] \end{aligned} \tag{6.3}$$

where $I = 1\ A$, $K_t = 0.08\ Nm/A$, $\Delta v = 1313.906 \times \frac{2\pi}{60}\ rad/s$ and $\Delta t = 588.889 \times 10^{-3}\ s$.

This results in a motor moment of inertia of $J_m = 3.424 \times 10^{-4}\ [kg/m^2]$.

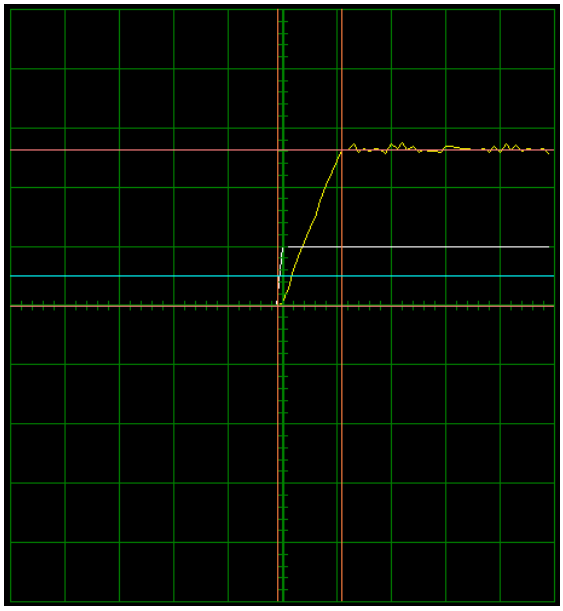


Figure 6.16: Velocity vs. time plot for 1A equivalent DC command.
(500 rpm; 500 ms/div)

Calculation of B_m

The motor damping or viscous friction, B_m, was assumed to be negligible. Brushless DC motors have near zero damping and will have little effect on the simulated motor model.

Calculation of τ_e and τ_m

The electrical and mechanical time constants of the motor, τ_e and τ_m respectively, can be used to plot a root-locus plot with poles at $-\tau_e$ and $-\tau_m$ as can be seen in fig. 6.17. This is useful when designing a current controller for the system. τ_e and τ_m can be calculated using eq. (6.4).

$$\begin{aligned} K_m &= \frac{1}{B_m} \\ \tau_m &= \frac{J_m}{B_m} \\ K_e &= \frac{1}{R_m} \\ \tau_e &= \frac{L_m}{R_m} \end{aligned} \tag{6.4}$$

From eq. (6.4) and using the previously calculated motor constants, $\tau_e = 7.368 \times 10^{-4}$

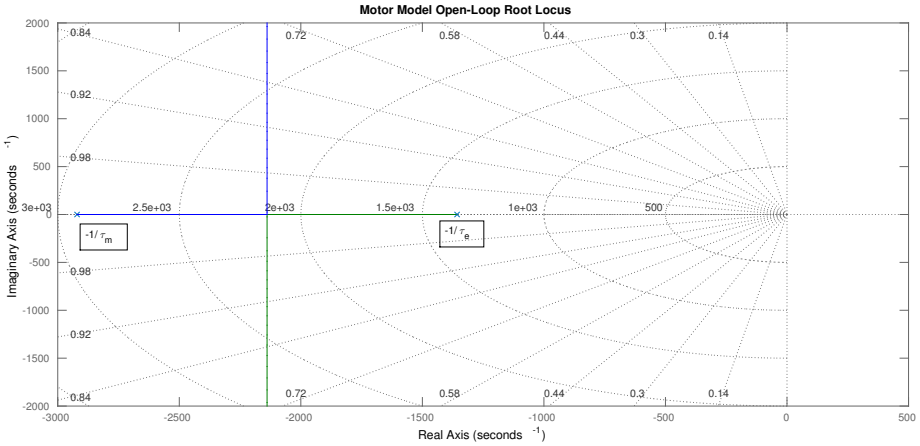


Figure 6.17: Motor model open loop root-locus plot.

and $\tau_m = 3.424 \times 10^{-4}$. This is assuming the motor viscous friction B_m is insignificant which is usually the case in mechanically well made BLDC motors.

The resulting motor model open loop root-locus plot can be seen in fig. 6.17. As expected the system has only negative real roots and will be stable in open loop.

6.6.4 Driver Configuration

The AMC drivers allow extensive customisation. After the motor, encoder, and general communication control parameters are configured, the PID control loops of the drivers can be configured, as seen in fig. 6.18.

The motor drivers were initially configured with both on-board PID current and position control loops enabled. This allowed initial modelling of the motors, configuring of the motor encoders, and determining of the position limits (in counts). For control of the leg, the position control loop was finally implemented on the STM32F4 microcontroller, while using the existing current control loop of the motor drivers. The custom position control loop was indirectly implemented using the spring-damper virtual model and force control by setting the polar coordinate set-points, in comparison to the AMC driver position loop which used a PID controller specifically controlling position.

The AMC drivers were configured using the AMC Driveware configuration software, which provided an oscilloscope to measure the relevant motor responses as seen in figs. 6.16, 6.19 and 6.20.

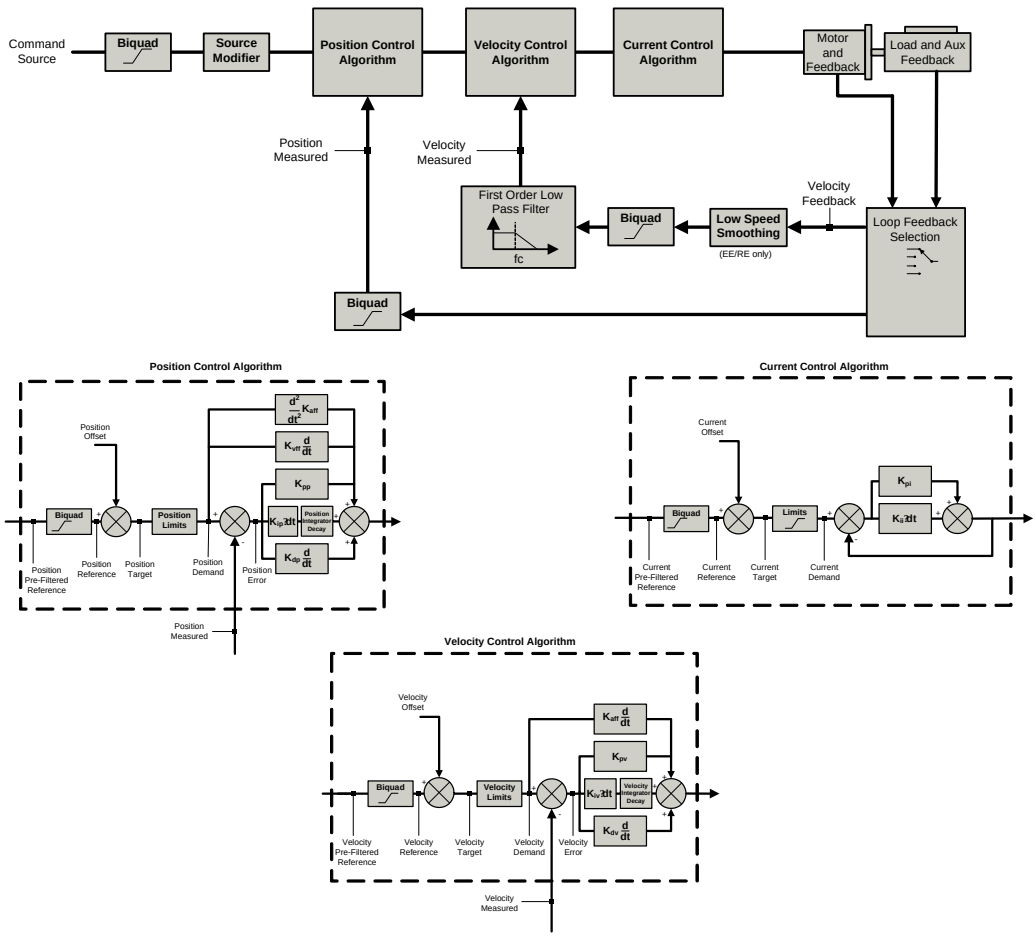
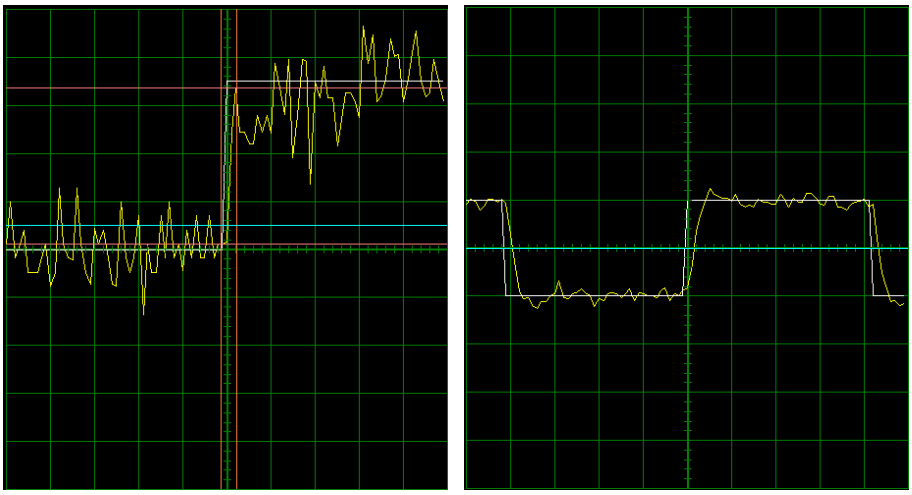


Figure 6.18: AMC DigiFlex Performance Servo Drive control loops (AMC, 2014).



(a) Motor driver current loop pre-tuning. (200 mA; 5 ms/div)

(b) Motor driver current loop tuning - 1A 120Hz square wave command. (1 A; 1 ms/div)

Figure 6.19: Motor driver current loop tuning plots.

Current Control Loop

By using a 1 A 120Hz square wave current command the current PI control loop was tuned, as seen in fig. 6.19. Initially both the proportional gain, K_p, and the integral gain, K_i, were set to zero. K_p was slowly increased until the final amplitude of the current output just started to overshoot. K_i was then set to minimize the steady state error.

Values of $K_p = 0.277$ and $K_i = 0.262$ were obtained. Both motors were found to operate optimally with the same PI gain values.

Position Control Loop

The on-board AMC motor driver control loop was set up to test the encoder configuration. The encoder and relevant position limits can be seen in subsection 6.6.5.

A 1Hz sinusoid was used to tune the PID control loop gains. Values of $K_p = 0.0005793$, $K_i = 0.0006052$ and $K_d = 2.769e^{-9}$ were found to achieve optimal set-point tracking as seen in fig. 6.20. The sinusoidal set-point can be seen in white and the position feedback in yellow. A 10-30 ms lag time can be seen due to the inertial load. This lag time causes a dead-band which should be considered when implementing a controller.

These tests were performed with the leg attached - the inertial load provided by the

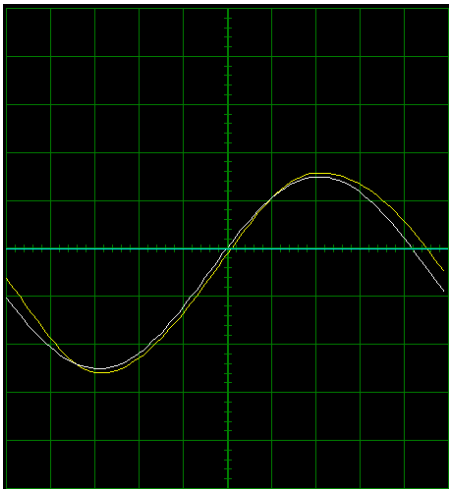


Figure 6.20: Motor driver position loop tuning - (-350:200) count 1Hz sinusoid command with 300 count offset.
(100 ct; 100 ms/div)

leg made PID control loop tuning possible, whereas without any inertial load the BLDC motors overshot their set-point.

6.6.5 Motor Encoders

The Avago Technologies HEDL-5640-A13 rotary encoder was used for feedback of encoder position to the motor drivers which then calculate the motor position and velocity relative to a starting encoder count position.

The encoder has the following specifications:

- Optical sensing.
- Incremental counting.
- 500 counts per revolution.

The Avago encoder was chosen for its light, easily mountable frame along with the relatively high resolution position feedback of $0.72 deg./count$.

A shaft was designed to mount to the rear of the BLDC motor and interface to the encoder. The shaft was designed in OpenSCAD using programmatic CAD and can be seen in fig. 6.21. It was 3D printed by Justin Pead in White Lab using PLA plastic.

The shaft was designed to be mounted using three hex screws to the provided mounting

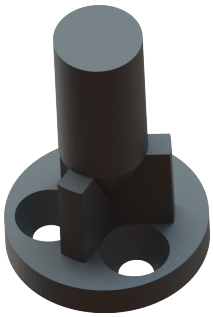


Figure 6.21: 3D printed PLA motor encoder shaft.

point on the rear of the motors. A star configuration was used for the interface between the shaft mounting point and the cylindrical encoder shaft so that these hex screws could be accessed easily.

Ideally the shaft should be milled using metal for better heat dissipation. Initially, before the aluminium mounting plate was used, the encoder shafts warped slightly while in operation and had to be bent back into shape. 3D printing was used for rapid prototyping and due to cost considerations.

7 Software Development

The development of two core project deliverables will be discussed:

1. Functional configuration, control and logging platform.
2. Reliable embedded system control system and motor driver communication protocol.

The following software development environments were considered: C/C++ (Qt), Python, Processing (dedicated graphic programming), and Matlab (serial toolbox).

The Qt C/C++ GUI development environment was chosen for the following primary reasons:

- Low over-head programming language.
- Open source software.
- Low level computer hardware access results in near real-time serial packet processing and accurate timing.
- Compatibility with C packed struct packet encoding and decoding code (as developed in section 7.2).

The design, operation and commissioning of the software system will be covered in detail.

7.1 RTOS Communication Protocol

Ideally FreeRTOS tasks should operate with as much isolation as possible, with each task performing a specific function with different priority levels. In the case of a communication protocol, the overall task is inherently sequential so this leads to FreeRTOS being used in a sequential way with all tasks operating at real-time priority level.

FreeRTOS was chosen because of the following benefits:

1. Segmentation of code into specific tasks provides readable code for future users.
2. FreeRTOS task configuration provides a framework for future embedded robotic control systems as well as easily reconfigurable code.

3. Semaphores and queues lend themselves to sequential reception and processing of packets at predefined intervals very well.
4. Isolation of tasks provides easy debugging of communication issues.

7.1.1 Heartbeat Task

The *Heartbeat* task's primary function is to synchronise the communication protocol. It is the only task running with a 5 ms non-blocking delay. Every 5 ms it completes the following functions:

1. Compiles a read command packet for current, position and velocity.
2. Appends these three packets to the two motor driver transmit queues.
3. Gives a binary semaphore to the motor 1 and motor 2 transmit tasks.
4. Starts a 5 ms non-blocking delay to allow the two motor transmit tasks to complete their transmission and for the motor driver replies to be received and decoded.
5. Gives a binary semaphore to the PC transmit task to compile and send the newly received motor data for logging.

7.1.2 PC TX Task

The *TXPC* task is used purely for logging of data over serial on the Baleka C++ application, as seen in section 7.4.

Data is received in a queue from both the motor *RXMotor1* and *RXMotor2* tasks as well as the *Controller* task which is compiled into a packet along with status bits indicating various events and conditions.

Status bits in the packet indicate to the PC logging software whether specific data has been successfully received by the *PCTX* task in the queue. A CRC check is added for transmission based error detection.

The task is run every $5\ ms$ once the motor data has been requested, received and processed, and the *Controller* task has instructed the motor drivers to perform a specific command.

7.1.3 PC RX Task

The *RXPC* task is the only task that is required to perform non-synchronous reception of packets. The DMA reception is initialised and then the task waits until the full PC packet has been received. The RX complete interrupt handler then gives the *PCRX* task a semaphore to continue decoding of the data. The use of non-blocking DMA UART reception and semaphores ensures other tasks are able to run during the reception process.

The *RXPC* task checks how much data has been received before searching for the start-bytes of the packet. Once a valid packet has been found it is copied to a packed struct and the CRC is calculated and compared to the CRC member of the struct. If it is valid then the *RX_DATA_VALID* flag is set, this indicates to other tasks that they can safely use the packet data received.

Every packet has an op-code that indicates what data is present in the packet. A switch statement is used to process the data appropriately and perform the necessary functions. These functions range from configuring the motor drivers, killing the drivers, calibrating motor positions and triggering various on-board events such as jumping and leg trajectories. All these functions operate by setting flags or compiling packets to be sent to the necessary queues for transmission.

An example of the packet processing code can be seen in listing 5.

7.1.4 TX Motor Task

The *TXMotor* task functions as an interface between the rest of the tasks and the motor driver. The task is periodically run when a semaphore is received.

When the task starts a DMA reception from the motor drivers is initialised, which will only be halted once control is handed over to the *RXMotor* task.

Every time the task runs, at least three command packets are sent to the motor driver from the *TransmitMQ* queue. This queue is filled by the *Heartbeat* task which synchronises all communication protocol tasks. The three command packets request current, position, and velocity data from the motor drivers which is processed by the *RXMotor* task and used in the control loop every $5\ ms$.

The queue runs in a loop until transmission is complete, with a number of timing conditions being completed to ensure that a reply is received from the motor driver before

the next request is transmitted - if this is not done properly then data can be lost.

The other three queues are used for current commands, position commands, and for general motor driver management commands such as changing gain sets or initialising the motors. An important task for the general driver management queue, *CommandMQueue*, is to disable the motor drivers when position and/or current limits are reached.

7.1.5 RX Motor Task

The *RXMotor* task is used to process data received from the motor driver in a reply to a command. The task waits to receive a semaphore from the *TXMotor* task, after which the DMA reception is halted.

The task then begins processing the data by:

1. checking how much data has been received,
2. searching the buffer and finding the index of all start bytes,
3. finding unique user programmable op-codes based on location of start bytes,
4. using a switch statement to appropriately decode data using a packed struct based on the op-code,
5. and adding this data to a buffer before moving on to the next start byte index for processing.

This data buffer is then sent to the *Controller* and *TXPC* task queue before waiting for a semaphore to be received again.

7.1.6 Controller Task

The controller task's core function is to reliably implement the control loop design seen in fig. 8.4 and developed in chapter 8. This means the controller task runs at a consistent $200\ Hz$ frequency.

The control loop waits for new data to be received from each motor on a queue from the *RXMotor* task - this is expected to happen every $5\ ms$. If an error occurs or the data received from the drivers is invalid then the control loop will not run for safety.

Once the data has been received, if a valid *RXPC* packet is available it is checked for virtual model configuration data which is then used to set up the spring-damper model.

A *TRIGGER* flag and *START* flag are both checked before functions are run or the controller is allowed to operate. The *TRIGGER* flag is used to start jump and trajectory sequences.

Multiple calculations are performed to implement a control model:

1. Cumulative integral error term.
2. Jacobian mapping for velocity and position.
3. Spring-damper force vector which is mapped using to motor torques using Jacobian.
4. Torque current command mapping using torque constant as derived in subsection 6.6.3.

If there are invalid kinematic or current set-points a kill function is implemented that both stops the control loop from operating and kills the motor driver bridge.

If a current command is successfully calculated, a motor driver command is compiled and sent to the *TXMotor* task queue.

7.1.7 FreeRTOS Timing

The default 1000 Hz tick rate was overridden with a tick rate of 5000 Hz to enable more fine tuned packet timing and delays. The timing configuration can be seen in listing 2.

In order to achieve a control loop rate of 200 Hz, a sampling time of 5 ms or 25 ticks was used.

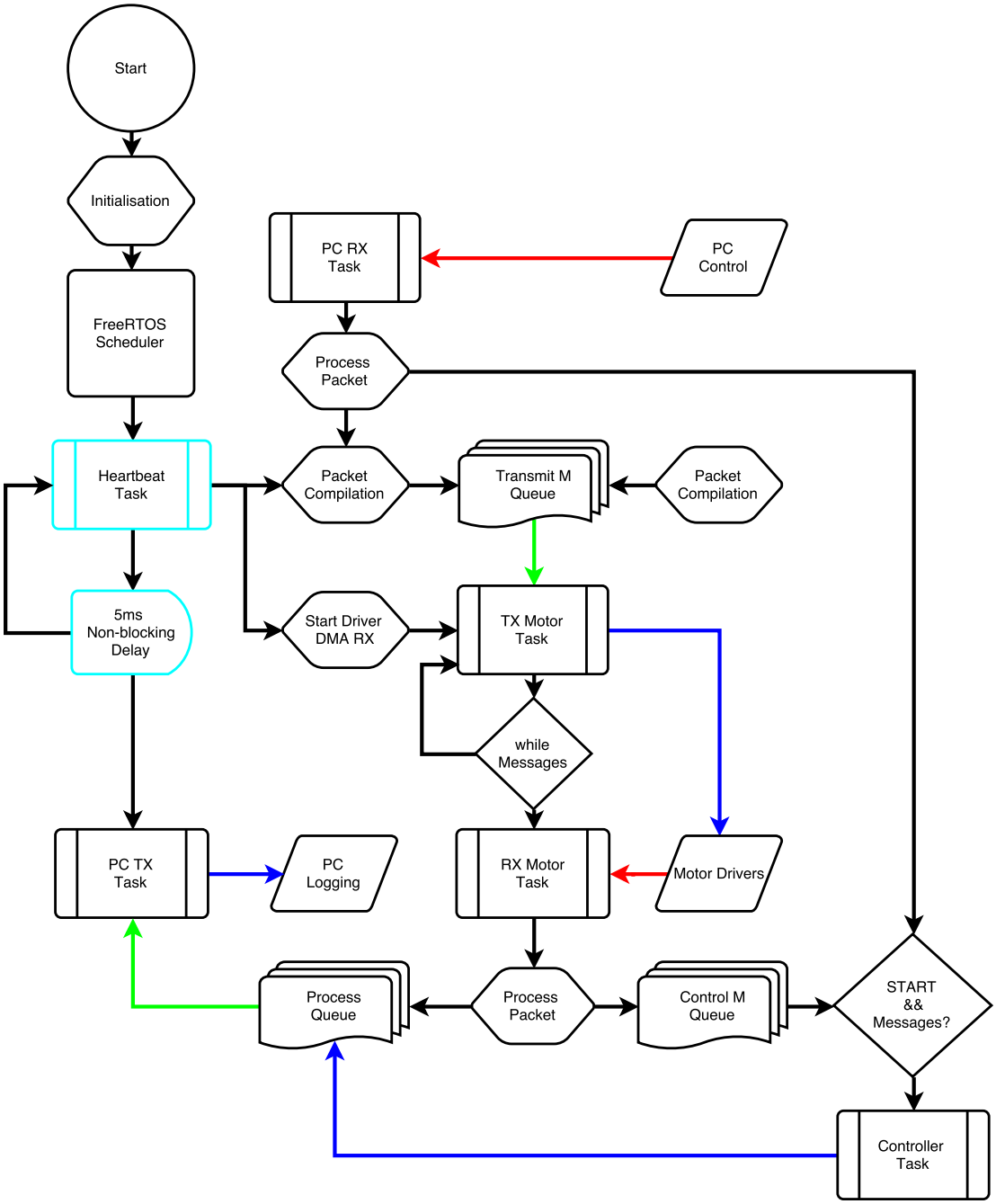


Figure 7.1: FreeRTOS communication protocol flow diagram.

7.2 Packet Transmission

7.2.1 Structuring

Usually when a C compiler stores data in memory it does so in a manner that optimizes memory access speed. In the case of a struct this isn't intuitively in successive memory addresses, but with padding.

Packets are compiled and transferred in bytes, the smallest variable size on any architecture apart from bit fields. A simple and elegant solution to packet structuring and decoding is to use a struct where the individual bytes are consecutive and byte aligned, much like an array, but with the benefit of accessing by struct member too.

This method has a few major benefits:

- To decode packets the data can be copied from a receive buffer directly over a struct and each of the struct members will contain the appropriate "decoded" data.
- To structure and compile a packet, all the relevant data can be assigned to a struct member and then when the packet is ready to be transmitted it is as simple as passing a pointer to the first value of the struct along with the packet length.
- The CRC check can be performed on a sequence of struct members with the guarantee that all members are consecutive and byte aligned.
- The packet length is the size of the struct!

The way to ensure that bytes are consecutive and byte aligned is to use the __attribute__((packed)) C compiler type attribute. This is a compile time flag that forces the struct to be byte aligned with no padding. An example of a packed struct from the embedded software can be seen in listing 3 in appendix B.

7.2.2 Integrity Checking

A Cyclic Redundancy Check was implemented for all communication. Polynomial division is performed on the data contents of the packet, this results in a remainder that constitutes the CRC that is appended to the packet. If this CRC does not match the calculated CRC on the receiver side then measures are taken to deal with the corrupt packet.

Device	Protocol	Width	Polynomial	Init	Refin	Refout	XORout	Check
iNemo/PC	CCITT-FALSE	16	0x1021	0xffff	FALSE	FALSE	0x0000	0x29b1
Motor Driver	XMODEM	16	0x1021	0x0000	FALSE	FALSE	0x0000	0x31c3

Table 7.1: CRC protocol configuration.

The CRC ensures the communication system is robust to interference and corruption, which can come from a number of sources:

- 3 phase alternating current flowing through motor phase wires.
- Motor EMF generated.
- RTOS and embedded system errors due to synchronisation.

Two CRC protocols were used. The CRC calculation function developed by James Gowans in December 2012 was customized to work with the motor driver two byte CRC protocol.

The iNemo and PC packets used the CRC-16/CCITT-FALSE protocol and the motor drivers are hard-coded to use the XMODEM protocol, with the configuration seen in table 7.1 and originally found in [19].

In order to reduce CPU time used for CRC calculations, two CRC lookup tables are generated before the RTOS scheduler takes control.

7.2.3 Compilation

Motor Driver Packet

A function was created to compile motor driver command packets based on the AMC motor driver packet protocol. The *BaseCommandCompile* function prototype can be seen in listing 1.

All motor driver packets either request data from a memory address or contain data to be written to a memory address. The command bits *ComBits* of the function are written to the packet, where 0x02 tells the motor driver to set data and 0x01 tells the motor driver to read data. *INDOFF1* and *INDOFF2* indicate the motor controller address and offset for reading or writing of data.

The *SeqBits* are a unique sequence of 4 bits that function as an ID for the packet - the

motor driver will reply with the same ID to indicate which command it is responding to. The response sequence bits are then used as an op-code for the packet processing switch statement used in the *RXMotor* RTOS task.

Some of the function inputs set up the structuring of the packets and identify the packets within the embedded system. *DATA* is a pointer to the data to be written to the address, *LEN* indicates how many bytes of data must be read from the motor driver address and *SNIP* is a parameter that is a workaround for proper packet compilation indicating how many bytes must be removed from the packet if the packet is for requesting data rather than writing data.

The final function input, *n*, is an index for the packet. This is an index for the array of structs that is used to compile the packets. This way when a packet command is to be sent to a function on one of the RTOS queues, just the index can be sent.

An example of an array of structs can be seen in listing 6 and an example usage of the *BaseCommandCompile* function can be seen in listing 7.

All of the parameters, op-codes and other information relating to motor command packet encoding can be found in table C.1 in appendix C.

```
1 void BaseCommandCompile(uint8_t n, uint8_t SeqBits, uint8_t ComBits,
2 uint8_t INDOFF1, uint8_t INDOFF2, uint8_t *DATA,
3 uint8_t LEN, uint8_t SNIP);
```

Listing 1: Motor packet compilation function.

PC Packet

The structure of the packets both received and transmitted by the PC can be seen in fig. 7.2 and fig. 7.3 respectively.

The typical start and stop byte (one) were replaced with start and stop bytes (two), being *[* and *]* respectively. This removed the need to search for and remove any start or stop byte in the packet data before transmitting. The chance of there being a series of two start or stop bytes in the data is 1 in $16 \times 16 \times 16 \times 16 = 65536$.

In fig. 7.6 the transmission duty cycle for packets transmitted from the embedded system to the PC, the larger of the two packets, is approximately 20% - this leaves plenty of time for additional data to be transmitted. The PC TX packet is 75 bytes and the PC RX packet is 59 bytes. As additional data is appended to the packets via the packed struct,

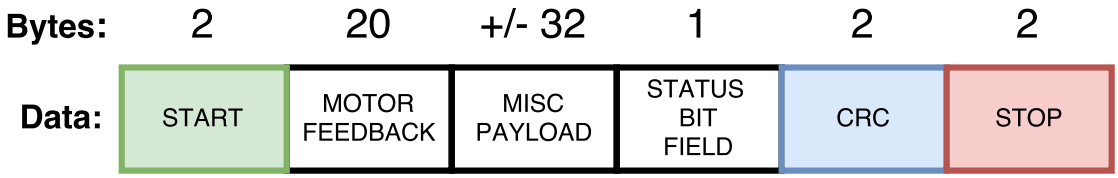


Figure 7.2: PC RX packet structure.

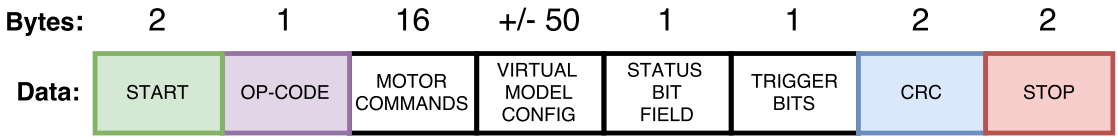


Figure 7.3: PC TX packet structure.

the protocol allows for quickly reconfiguring transmission and reception processes to accommodate this extra data for logging and live plotting.

The status bit field of the packet allows individual bits to be used as variables - this is a special C variable type that is useful for creating a sort of binary flag in the packet.

The trigger bits of the *PC TX* packet allows various events and sequences to be started in the embedded system control loop.

7.3 Peripheral Configuration

A basic peripheral configuration diagram can be seen in fig. 7.4 showing which UART and GPIO ports are routed to which pins.

7.3.1 Protocol

A number of serial protocols were used in the communication system. To reduce noise interference from motor EMI the RS-485 protocol was used. RS-485 uses differential signalling with a $200\ mV$ binary threshold. This protocol is designed for long distance low noise data transmission at data rates up to $10\ MBits/s$ at short to medium range.

The VD30 and VD485 RS-485 driver ICs were used to couple the TTL serial logic of the microcontroller to the RS-485 transmission lines. A RS-485 to TTL serial USB converter was used to interface with the PC. The iNemo microcontroller was directly connected via

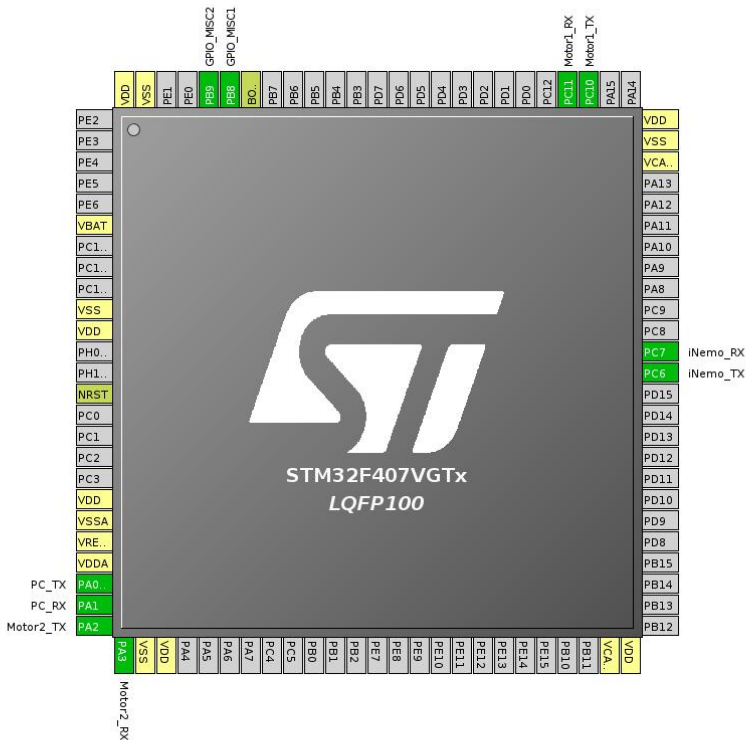


Figure 7.4: STM32F4 microcontroller peripheral configuration.

TTL level serial to the microcontroller. The communication protocol configuration can be seen in fig. 7.5.

7.3.2 Data Rates

Two data rates were selected for the motor driver interface, and the iNemo and PC interface.

The maximum communication speed of the motor controllers is 921600 baud over RS-485. Due to the high speed control loop frequency, and in order to achieve the high density packet transmission and reception as seen in fig. 7.6, the 921600 baud rate was selected.

In fig. 7.6 the motor 1, M1, command and reply sequence can be seen. Every time a command is sent the corresponding reply has to be received before the next command can be sent. This is an on-board motor driver problem, when commands are sent before the previous reply the motor driver fails to perform the command. This bottle neck greatly reduced possible control loop frequency. Without this limit it would be possible

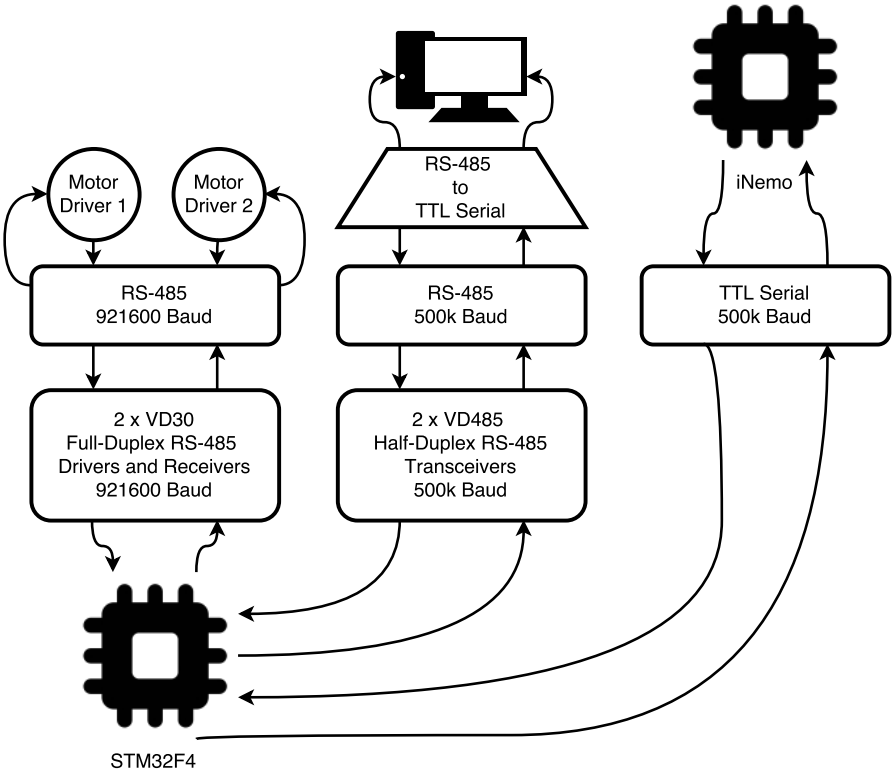


Figure 7.5: STM32F4 UART Serial Protocol.

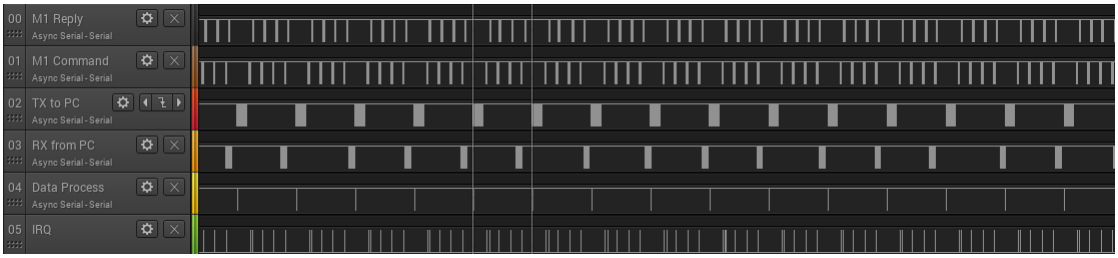


Figure 7.6: Communication protocol packet timing with 5 ms sampling rate.

to achieve upwards of $500\ Hz$ control loop frequencies resulting in much more accurate virtual model control.

The PC and iNemo data rate was limited to 500k baud for stability and noise rejection. The Qt application began to miss packets when the data rate was increased above 500k baud - this is likely due to the lack of DMA on personal computers and the overhead of the application. The iNemo was to be used for transmitting relatively small packets containing accelerometer and gyroscope data, so the data rate of 500k baud was adequate for this purpose.

If the need arises to append additional data to the communication packets the baud rate could be increased further to accommodate this within the $5\ ms$ maximum transmission period.

7.3.3 Direct Memory Access

DMA was used for all UART communication on-board the STM32F4. DMA enables control of the data transmission and reception process to be handed over to the DMA controller, of which there are two on board the STM with a number of streams. This means that rather than having an interrupt to process each received byte (10 bits in practise with the parity bit etc.), the memory is directly accessed by the DMA controller when transferring data from the receive buffer to a memory address which avoids using CPU time.

7.4 Graphic User Interface

To enable rapid prototyping, a basic pre-existing serial logging platform was used and plug-ins were developed to adapt the software for controlling, configuring, logging and live plotting of the Baleka robotic leg platform.

qSerialTerm, described as "A Qt based Serial Port terminal emulator", was built upon. It is distributed under the GNU General Public License (GPL) v3, which allows distribution, customisation and even sale of software published under the license, so long as basic conditions are met. It is copyright 2012 by Jorge Aparicio who's software repository can be found on GitHub: `https://github.com/JorgeAparicio/qSerialTerm`[20].

Qt Creator CPP was used for the software development and can be compiled to run on both Linux and Windows platforms - for reference all development was completed on a Linux platform.

Three .cpp files were used for the majority of the added functionality, namely: CRC.c, framewidget.cpp and serialportwidget.cpp along with their respective header files and Qt forms.

7.4.1 Serial Communication

The serial interface is the first step in configuring UART communications. After successfully opening a port it allows you to select basic configuration options. The GUI extract can be seen in fig. 7.7[20].

A button was added, *Default*, to allow quick set up of the software for experimentation with the Baleka leg.

The maximum stable baud rate with the Qt software was 500k, any higher and the accuracy of the packet timing decreased due to application overhead.

The refresh rate, when set to any value lower than $10\ ms$ tended to vary by as much as $5\ ms$ during operation. On Unix based operating systems the quoted accuracy of software timing is $\pm 1\ ms$ and for Windows machines it is $\pm 5\ ms$, making the decision to use a Linux machine natural. The Qt software timer resolution was increased to allow the possibility of a $200\ Hz$ refresh rate.

The refresh rate was used to set the packet reception and transmission frequency. Due to the inconsistent timing real-time control of the leg from the Qt application was not possible, this was because of the large ± 60 byte packets being transferred and the application overhead.

A compromise was found by using the Qt application for configuration of the virtual model and performing non-time critical tasks, as well as real-time logging, while the real-time control was implemented on the STM32F4 embedded system.

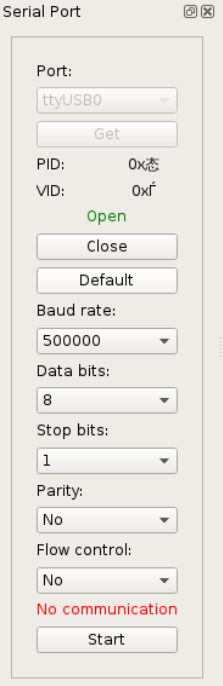


Figure 7.7: Serial port configuration.

7.4.2 Logging and Live Plotting

The logging software was adapted to create CSV files of all the data received from the microcontroller embedded system after it has been appropriately converted to a human readable format.

The live plotting function of the software takes an array of data of a known data size and endianness and plots the change in these values over time. There are a number of plotting options such as samples shown and format options. The live plotting proved useful for debugging of data processing issues and during experimentation to see spring-damper responses as virtual model configuration options are changed.

A number of data type conversions took place from serial reception to data plotting and logging, as follows. Qt has a number of application specific data types, usually prepended by a 'Q':

1. Serial data is read into the QByteArray RXBuffer.
2. RXBuffer is then copied to a char array to enable the packet to be decoded using the packed struct method of subsection 7.2.1.
3. Struct members are accessed and a union is used to convert between data types

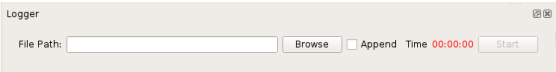


Figure 7.8: Packet data CSV logging.

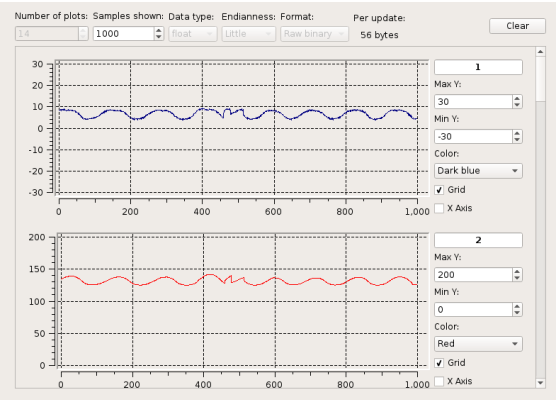


Figure 7.9: Live plotting of motor feedback and controller data.

and perform calculation operations.

4. Processed data is converted to a QString using the number operator and appended to the QByteArray PlotBuffer as well as the QStringList CSVList which is converted to CSV format.

5. Finally both CSVLog and PlotBuffer are emitted as Qt signals for file logging and live plotting respectively.

A CRC check was implemented to check the packets being received are valid. Occasionally data points are not logged or plotted - this is occasionally due to CRC failures, but usually due to the variation in processing frequency of packets due to inaccurate timing discussed in subsection 7.4.1.

All logging and plotting development took place in the *serialportwidget.cpp* file and corresponding header files and Qt forms.

The GUI for logging can be seen in fig. 7.8 and for plotting in fig. 7.9.[20]

7.4.3 Control Plug-in

The control plug-in was custom designed and developed for the Baleka robotic leg application. The following primary functions were integrated:

1. Safety.
2. Configuration.
3. Initialisation.
4. Virtual model control.
5. Sequence triggering.
6. Current and position control.
7. Control loop gain adjustment.

The safety function of the plug-in covers not only controls to disable the motors in a case of malfunction, but also ensures that users can not enter invalid values or values that result in kinematic collisions.

The configuration and initialisation section seen in fig. 7.10 allows the motor drivers to be enabled, on-board configuration set up to be selected and the motor driver encoder position to be calibrated.

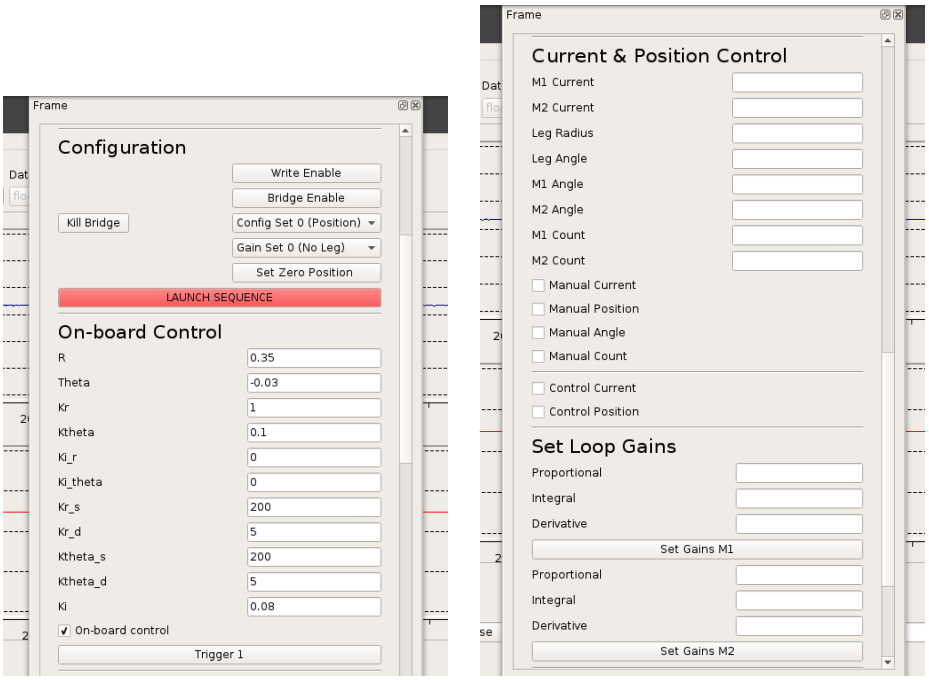
The encoder calibration takes the current leg position and sets both ϕ_1 and ϕ_2 to 180^o, measured from vertical as seen in section 4.1.

The *launch sequence* button combines all configuration options for a quick set up of the leg for virtual compliance or jump sequence control.

For the *on-board control* and *current & position control* sections various selection boxes and text boxes exist. The underlying software ensures that, for example, current and position can not be controlled at the same time. Whenever a value changes, an update event is triggered and the data is compiled, along with an op-code, before being sent to the embedded system. The op-code then indicates to the embedded system how the data should be processed and what configuration settings should be changed.

The data processing of the control plug-in works with op-codes being added to a processing queue. When a button is pressed or a value changed, the queue is processed until empty while a switch statement based on the op-code determines where the data is placed in the packet and how it is processed. The event handlers complete the in application configuration changes.

All control plug-in functionality development took place in the *framewidget.cpp* file and corresponding header files and Qt forms.



(a) Configuration & On-board Control

(b) Current/Position Control & Control Loop Gains

Figure 7.10: Control interface plug-in.

8 Controller Development

"Jump!"

— Van Halen, *1984*

8.1 Active Compliance

Passive compliance consists of using mechanical spring-damper components to adjust the dynamics of a robotics end effector. These come in the form of torsional springs, linear springs, hydraulic and pneumatic dampers to name a few.

In the study by G.A. Pratt and M.M. Williamson passive compliance in the form of a Series Elastic Actuator (SEA) was said to provide shock tolerance, lower reflected inertia, more accurate and stable force control, less damage to the environment, and the capacity for energy storage.[21]

Active compliance (AC) in robotic platforms is a relatively new area of research. Active compliance was chosen instead of passive compliance (PC) for the following reasons:

1. AC is mechanically simpler.
2. AC is mechanically lighter.
3. Good PC is expensive to implement due to lack of application specific off-the-shelf parts.

Most importantly, AC can be configured on the fly to adapt to the environment of the robot. This allows a robot to, much like their biological equivalents, comply to the environment based on various sensory inputs, or to perform various acrobatic tasks making use of spring-damper compression and decompression.

Various combinations of AC spring-damper configurations were designed, implemented and tested in order to determine the best topology.

A high level controller was used to switch between the active compliance models during phase transitions as developed in section 8.6 and seen in fig. 8.3.

8.2 Dynamic Stability

Stand still, look around, turn around - these apparently simple tasks are composed of a network of biologically advanced sensors and muscular motor movements that seem intuitive. To do the same relatively static and stable movement with a physically free robot is complex.

The alternative to static stability is dynamic stability. Much like a spinning top remains stable due to gyroscopic effects when a stationary top topples over; a dynamically hopping leg remains upright with a relatively simple control system compared to a leg trying to remain upright and walk forward.

In the book by M.H. Raibert et al., *Legged Robots That Balance*, an easily adaptable simple control algorithm for dynamic legged hopping was published. Raibert showed that complex robots can be dynamically stable, but when at rest this is more difficult. For this reason the control algorithm used for Baleka relies on the fact that Baleka is constrained in a single axis. Control of Baleka in the vertical axis will be implemented during jumping.

8.3 Mechanical Impedance

The leg design has inherent mechanical impedance in the form of friction, slack and inertial mass. Without mechanical impedance the leg would continue acting like an ideal system when actuated. This can be seen in section 9.3 where the leg is modelled as a spring-damper - the amplitude of the oscillation of the virtual spring model with spring constant $K_s = 100\ N/m$ and no damping decays with time, confirming mechanical impedance.

This mechanical impedance is not easily accounted for in the dynamic model of the leg as it is highly non-linear in the case of a complex linkage system as used in the Baleka leg - this makes it difficult to control. The mechanical impedance was treated as a disturbance and in the case of dynamic movement of the leg it was ignored. Energy will be lost in the hopping motion and during leg movement, but this is insignificant and only noticeable when doing spring-damper tests as seen in fig. 9.4.

8.4 Control Loop Sampling Frequency

For high bandwidth proprioceptive force control S. Kalouche showed that the control loop sampling frequency should ideally be in the kHz scale.[9] This allows the foot force to ideally match the expected foot force when performing high frequency dynamic movements.

The control loop sampling frequency was practically limited by the motor driver response time as seen in fig. 7.6. For every packet sent to the motor driver a response is required before the next packet can be sent otherwise the driver system becomes unstable. A maximum stable sampling frequency of $200Hz$ was achieved with room for packet decoding, processing and controller action.

During the jump tests in section 9.5 the limitations of a low sampling frequency were seen. Highly dynamic manoeuvres had a duration of approximately $20\ ms$ and this left 4 sample points for the controller to act.

8.5 Force Control

Using the virtual model developed in chapter 5 we need to take the polar spring damper force and map it to the Baleka two degree of freedom system where the motors provide the needed torque.

The forward kinematic Jacobian developed in chapter 4 is used to determine what the motor torques are for specific r and s force components, where F_r and F_s are the components of the foot force vector. In order to calculate the virtual model foot force components, F_r and F_s, we use the method developed in section 5.1.

The general Euler-Lagrange dynamic equation used in [22] can be further simplified for the virtual model force control problem:

$$M\ddot{q} + C\dot{q} + G = B\tau_m + Q + A^T F \tag{8.1}$$

Assuming we can ignore centrifugal and Coriolis forces, C, as well as gravitational forces, G. The generalized force vector, Q, will be ignored for simplicity - comprising all disturbances and friction. Disturbances and friction will be accounted for during foot force calibration, under the assumption of a constant frictional force and miscellaneous disturbances during motion.

Assuming both motors contribute to the end effector force equally, we can make the input mapping matrix B the identity matrix.

Taking the Jacobian of the kinematic mapping $f(\phi_1, \phi_2)$ the foot force vector, F, can be transformed to the motor torque commands, τ using eq. (8.1) as follows:

$$\tau_m = J^T F \tag{8.2}$$

Substituting for the Jacobian and force vectors:

$$\begin{aligned} \tau_{m_1} &= J_{11}^T \times F_r + J_{12}^T \times F_s \\ \tau_{m_2} &= J_{21}^T \times F_r + J_{22}^T \times F_s \end{aligned} \tag{8.3}$$

The motor current needed to produce the resulting motor torques in eq. (8.3) is then calculated as further discussed in subsection 8.5.2.

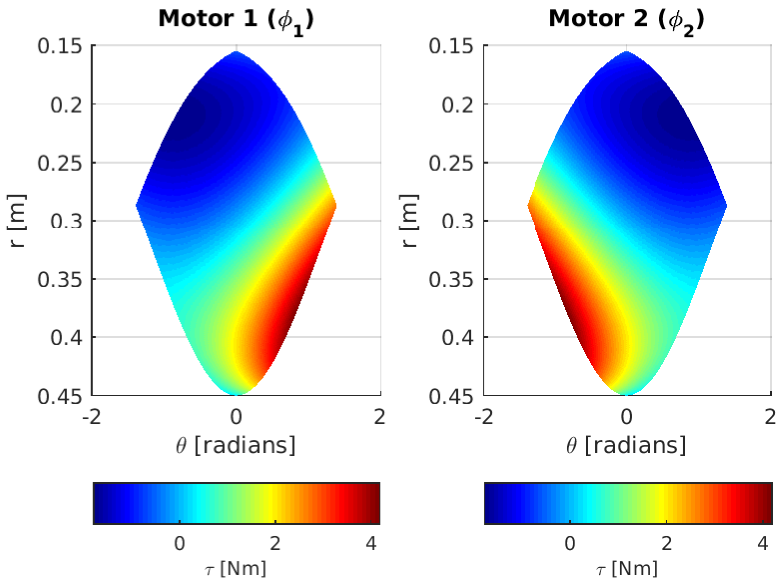
Motor Torque Predictions

Using a virtual model compliance model with a spring force of $K_s = 250\ N/m$ on both the radial and rotational leg polar coordinates, the motor torque predications were simulated over the kinematic workspace for each motor. Figure 8.1 shows the motor torque plots for full leg compliance control and fig. 8.1 uses purely a torsional spring.

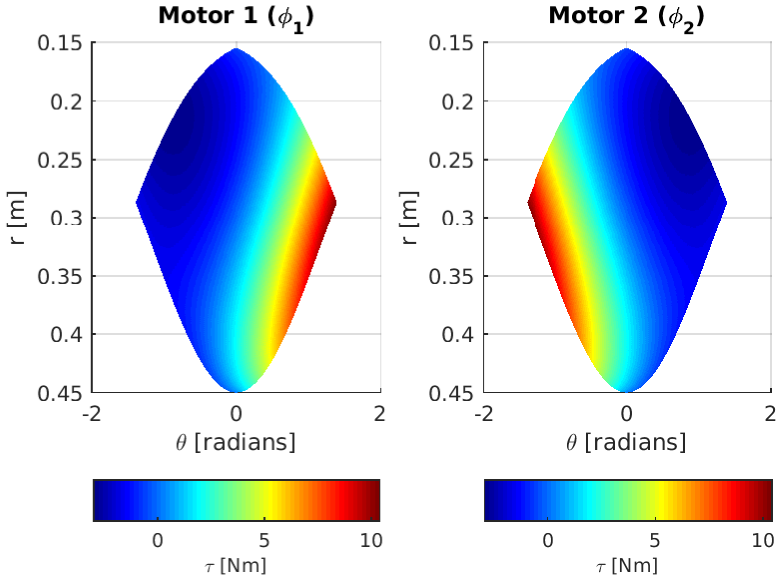
The data analysis shows that a higher motor torque is needed to implement a torsional spring based on the arc-length measure than the radian measure, as developed in section 8.7. When the leg is displaced rotationally, the motor on the opposite side to the rotation develops the most torque to try and correct the error. The motors clearly develop more torque for a rotational offset than for a radial offset, as shown by the heat map hot spots on the edges of the plot.

8.5.1 Non-conservative Forces

In further study the non conservative forces could be determined and accounted for, but a key to virtual model control is to allow the system dynamics to naturally occur, as discussed in [15], and the virtual components are effectively transposed on this response. If inverse dynamics are used to effectively cancel the non-conservative forces, then the system can become unstable.

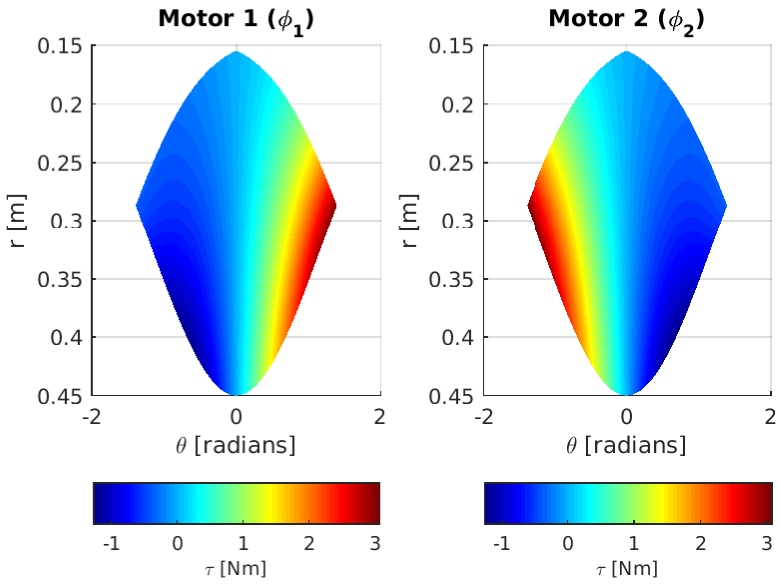


(a) $K_s = 250$, $K_d = 0$, using arc-length measure.

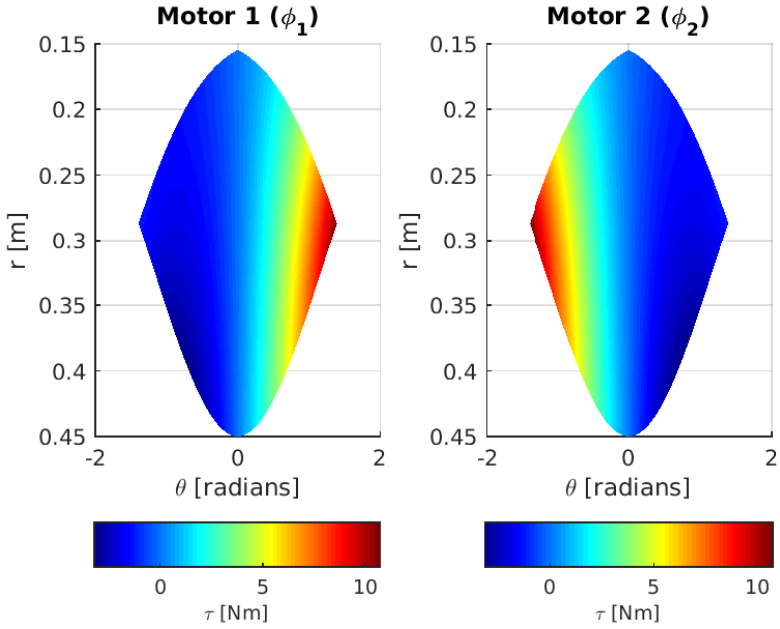


(b) $K_s = 250$, $K_d = 0$, using radian measure.

Figure 8.1: Radial and rotational spring foot force mapping to motor torque.



(a) $K_s = 250$, $K_d = 0$, using arc-length measure.



(b) $K_s = 250$, $K_d = 0$, using radian measure.

Figure 8.2: Rotational spring foot force mapping to motor torque.

The beauty of virtual model control is in its simplicity. For the particular case of dynamic movement and jumping, the external forces can be allowed to take place and the virtual model components can be adjusted according to non-exact specifications. If on the other hand the system needs to perform precise manoeuvres, for example during robotic surgery, you'd want to be able to predict the non-conservative forces and account for them accurately.

8.5.2 Current Control

The specific motor current requirements for force control were determined in the simulation in fig. 6.11. A maximum current of $52.3\ A$ was simulated and $60\ A$ motor drivers were selected.

The torque constant K_t was experimentally calculated in subsection 6.6.3 and determined to be $0.08\ Nm/A$. Using eq. (8.3) we can calculate the motor torque required to produce the necessary virtual model foot force. This foot force can then be mapped to a current command using the ideal linear relationship between motor torque and current:

$$\tau_m = \frac{1}{K_t} J^T F \tag{8.4}$$

It must be noted that in the implementation on the embedded system it took some experimenting to determine the correct current polarity for each motor.

During testing the current saturated at $60\ A$ during jumping. This effectively means the necessary impulsive force needed to jump to a set height was not delivered. At lower height jumps, with less current saturation, it was possible to better control the height.

Current limits in the embedded system control loop were set to just under the peak motor driver current in order to avoid current cut-out. During further testing, current cut-outs were not avoidable due to the continuous current limit of $30\ A$. In appendix E an example of a current cut-out during a jump is shown.

8.5.3 Calibration

Using a load cell the value for K_t was calculated in subsection 6.6.3 and the experiment performed in section 9.2. By iteratively adjusting K_t until the force output matched the

commanded force, we can accurately control the motor torques with the appropriate current command.

8.6 Control Loop Design

Figure 8.4 is an adaptation of the control loop design found in [9]. In [9] continuous position loop gain scheduling is used to implement force control, whereas Baleka uses virtual spring-damper gain scheduling. This means that instead of using a position control loop and changing the gain to achieve the appropriate force, the force is changed by directly changing the virtual model spring constants.

Borrowing from Raibert type controllers the jumping action can be split into phases, each with their own dynamic properties. These phases, and the transition between them, can be seen in fig. 8.3 as developed for Baleka.

At each phase transition a high level finite state machine switches between appropriate virtual model configurations. In the control loop, fig. 8.4, these transitions are triggered by the radial position of the leg. The three specific control algorithms are for flight, launch and stance control.

The integral error action of the control loop ensures that there is zero steady state error. The rest of the control loop blocks implement the theory developed in the dynamic modelling section, chapter 5, and the force control section, section 8.5.

The only part of the control loop that is left to the motor controller is the current control loop, which implements a PI controller. The motor controller gains were configured in subsection 6.6.4.

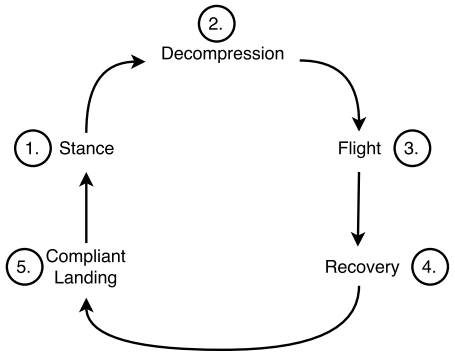


Figure 8.3: Jump phase transitions.

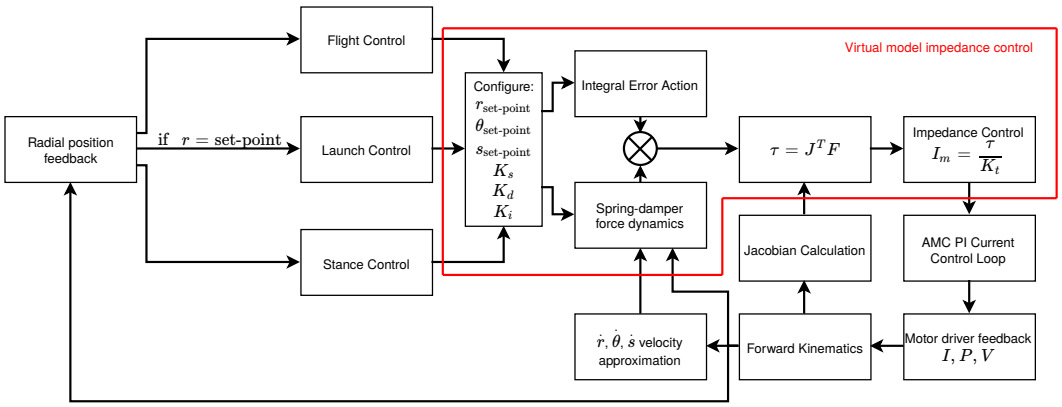


Figure 8.4: Virtual model impedance control loop.

8.7 Torsional Spring-damper

8.7.1 Force Normalisation

In previous studies using controllers for legs of similar topology, such as those by Duperret et al [3] and Kalouche [9], Cartesian coordinate systems have been used for virtual model compliance control in the case of [9] and in the case of [3] a Raibert type controller was used with a polar coordinate system.

The study by Kalouche [9] used a convenient coordinate system that had all axis of foot force and placement control in the same unit of measurement, namely distance in meters.

By using a polar coordinate system, virtual model compliance control is made more intuitive as further developed in chapter 4 and chapter 5.

The drawback to the classic polar coordinate system is the different units of measurement for radial $[m]$ and angular $[radians]$ measure. When applying torsional spring-damper dynamics to the model via control this results in vastly disproportionate force vectors that when mapped to motor torques via the Jacobian, derived in chapter 4, result in an unstable control system unless the foot force components are properly scaled.

An elegant solution to this problem and a way to normalise the angular foot force component is to use the relation in eq. (8.5) and visualised in fig. 8.5.

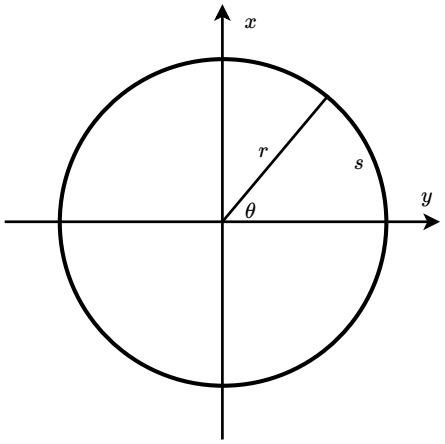


Figure 8.5: Arc-length vs. radian measure geometry.

$$s = r\theta \tag{8.5}$$

Using the arc-length of the leg model instead of the radian measure results in a unit of measurement in meters, a more easily interpreted force component, and a better behaviour under ground reaction forces as explained in subsection 8.7.2.

8.7.2 Ground Reaction Force

A ground reaction force (GRF) is a force exerted by the ground on a body that is in contact with it. In the case of the leg model it generates a torque around the center of mass of the body. The more distance there is between the center of mass and the ground, the more prominent the ground reaction forces will be.

There are two ways to deal with ground reaction forces - either by treating them as disturbances in the control model or by making the dynamic model more resilient to the torques generated by GRFs, or perhaps a combination of both.

The GRFs are most significant during the leg's impact with the ground after flight. During this landing stage all of the kinetic energy of flight is transferred to spring potential energy and damper kinetic energy. The leg radius also changes during this stage which has a direct relationship to torques generated by the GRF around the center of mass.

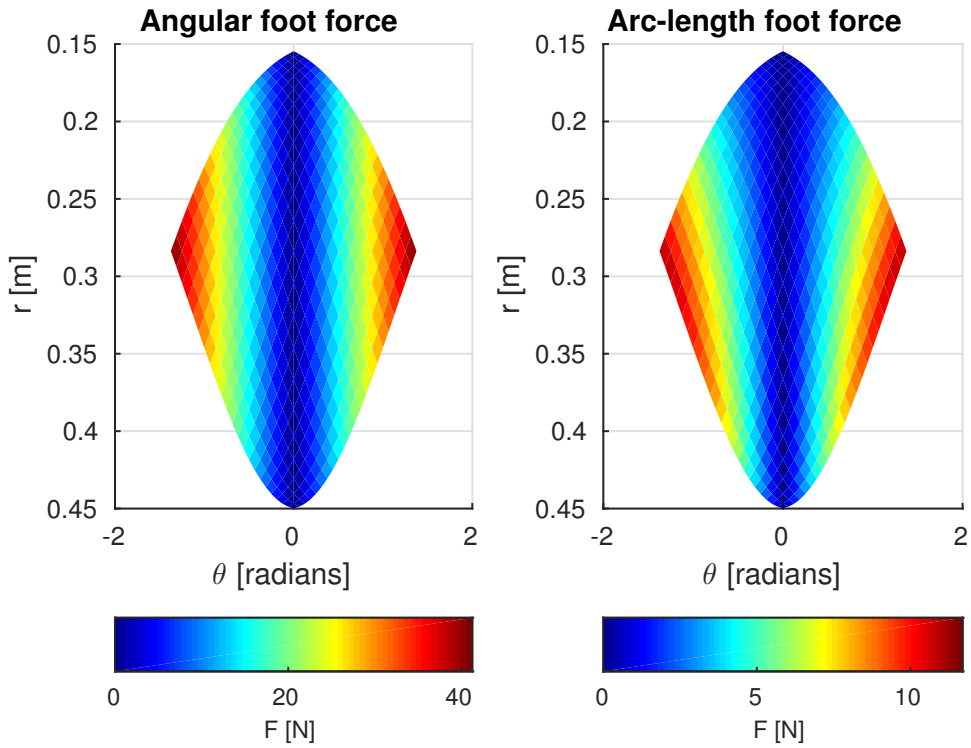


Figure 8.6: Rotational foot force comparison using angle and arc-length torsional spring virtual model.

By using arc-length for the virtual spring-damper control model it couples the resulting torsional foot force to the GRF torque. The result of this control method, as well as the effect of normalisation of torsional foot forces, is clearly seen in the foot force simulation of fig. 8.6. The simulation shows the relation between all possible foot positions in relation to the leg body with a colour map of the resulting torsional foot force at those positions overlayed. This is assuming a torsional spring constant of $300\ Nm$, zero damping, and a rotational set-point of $s = 0\ m$ - visualised in fig. 5.1.

Practically, using arc-length as a measure of rotational off-set for spring-damper virtual model implementation results in a coupling between radial and rotational spring forces. This has beneficial effects, as developed above, on the dynamics and control of the system.

8.8 Full-leg vs. Joint Active Compliance

The theory of torsional and linear spring-damper systems was developed in chapter 5. Theoretically, if the necessary kinematic and resulting Jacobian mappings can be derived, a torsional or linear spring-damper can be applied in the virtual compliance model around any joint or axis of movement. This leaves endless possibilities for leg virtual compliance topologies.

Two topologies were used in the study [9], where a slightly different multi DOF leg model was used. Namely full-leg and joint active compliance. For consistency and to compare results with the study, both of these topologies were tested in chapter 9.

These two topologies are not intuitively comparable, but by using simulation and performing experimental tests it is possible to compare the two. Realistically the full-leg compliance model is better suited to hopping control because of the polar vs. motor angle orientated general co-ordinates, where foot position can be directly controlled using the full-leg compliance model.

The full-leg spring-damper model can be visually seen in fig. 5.1 and the joint spring-damper model in fig. 5.2.

8.9 Simulation

The virtual model controller, kinematic mapping, and motor dynamics were all modelled in Simulink.

The high level dynamic and kinematic control simulation model can be seen in fig. 8.7. It was developed using a number of custom functional blocks to mathematically implement the inverse and forward kinematic mappings (in orange), the motor dynamics (in yellow), the velocity mappings (in green) and the force controller (in blue).

The kinematic mappings used the mathematical relations developed in section 4.2 and the motor model was based on the brushless DC motor model parameters developed in subsection 6.6.3. The motor simulation model can be seen in fig. 8.8.

The virtual model force control block used the virtual model force control calculations, including the kinematic Jacobian calculations, in order to control the current of the motors to command a torque. The torque was mapped assuming a spring-damper system with a

nominal spring constant of $K_s = 250\ N/m$ and damping constant of $K_d = 10\ N/(m/s)$ for both the radial and angular polar coordinates.

A radial set-point of $0.4\ m$ was commanded with an angle of 0^o. The inverse kinematic block was used purely to set up the correct initial conditions and no further position control was implemented. In the virtual model block a radial set-point of $0.3\ m$ was used. The initial radial offset resulted in the spring-damper response simulated in fig. 8.9.

The simulation was performed with the following aims:

- Ensure the controller signals are within an acceptable range for safe operation.
- Investigate the validity of the control loop structure.
- Confirm virtual model configuration is reasonable.
- Provide data to compare real-life tests to for validation.

The simulation model made the following assumptions:

- The motor drivers take an equivalent DC current command, and thus an equivalent DC motor model was used for the brushless DC motors.
- A load torque of $1\ Nm$ was applied at the motor output to simulate the approximate load of the leg. This load was assumed to be constant.

The simulation had the following limitations:

- Motor model did not accurately describe the T-Motor BLDC motor being used, despite the derivation of motor model parameters. This is due to the assumption of an equivalent DC model. This model was adequate to critically investigate the aims listed above.
- The simulation performed was for a single radial set-point and was not a comprehensive test of dynamic motion.

8.9.1 Data Analysis

At a glance all the plots, seen in fig. 8.9, follow the expected sinusoidal decay response of a spring-damper system with an initial offset and left to freely oscillate.

The radial position plot shows the leg was appropriately limited between the minimum and maximum radial set-points of $0.15\ m$ and $0.45\ m$ respectively. The radial position then settled at $0.3\ m$ as expected, equal to the virtual model radial set-point.

The radial velocity plot had a peak of $3\ m/s$. In reality during jump testing the leg reached a maximum radial velocity of exactly $-3\ m/s$, which confirms the validity of this result.

The radial force reached a peak of approximately $45\ N$. In previous kinematic workspace simulations the radial force reached approximately $40\ N$ and inn testing the radial force reached a value of just under $40\ Nm$ for a spring-constant of $300\ N/m$. All of these results are reasonable and compare well.

The motor current reached a peak of $30\ A$ and a low of $-10\ A$. In the active compliance motor current requirement simulation seen in fig. 6.11 a peak of $52.3\ A$ is estimated with a low of $-22.46\ A$. Both of these simulations compare well and also match the results found during testing.

8.9.2 Summary

The kinematic response (consisting of radial position and velocity), and the controller command and motor response (consisting of motor current and radial force respectively), all met valid limits and settled at the appropriate values expected from the theoretical response of a spring-damper system with initial offset.

The kinematic workspace simulations performed to determine current and force requirements of the motor and leg validated the controller simulation plots.

The plots were distorted and thus can not be directly compared to results from the experimental tests performed on the leg. Despite this the simulations behaved as experienced during testing of the leg.

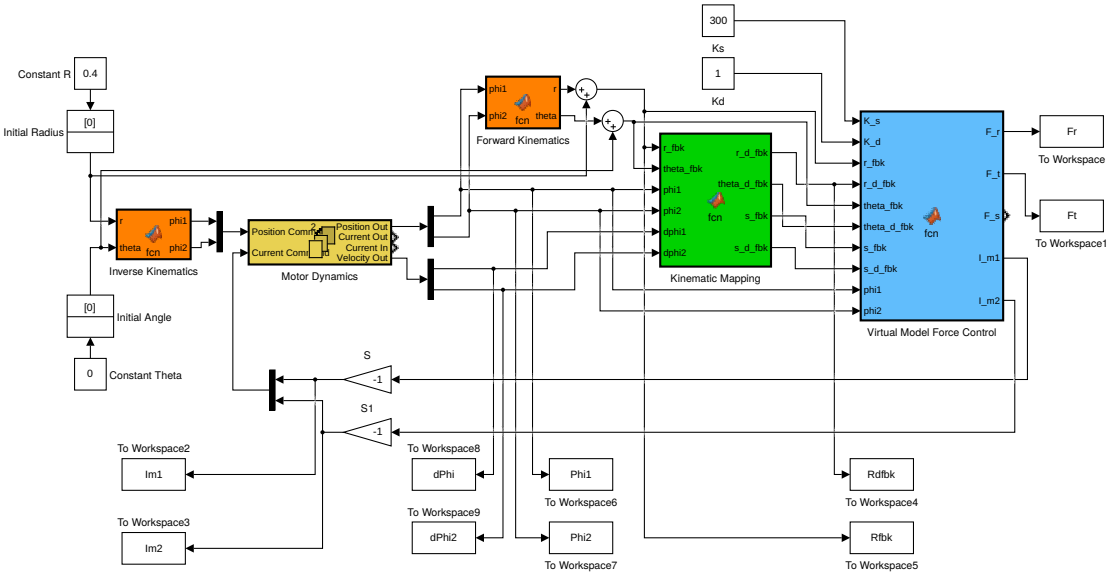


Figure 8.7: Dynamic and kinematic control simulation model.

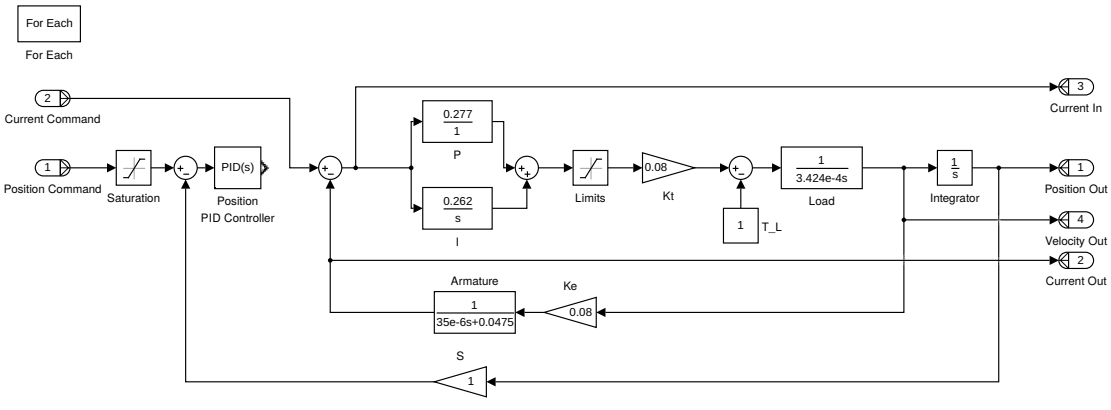
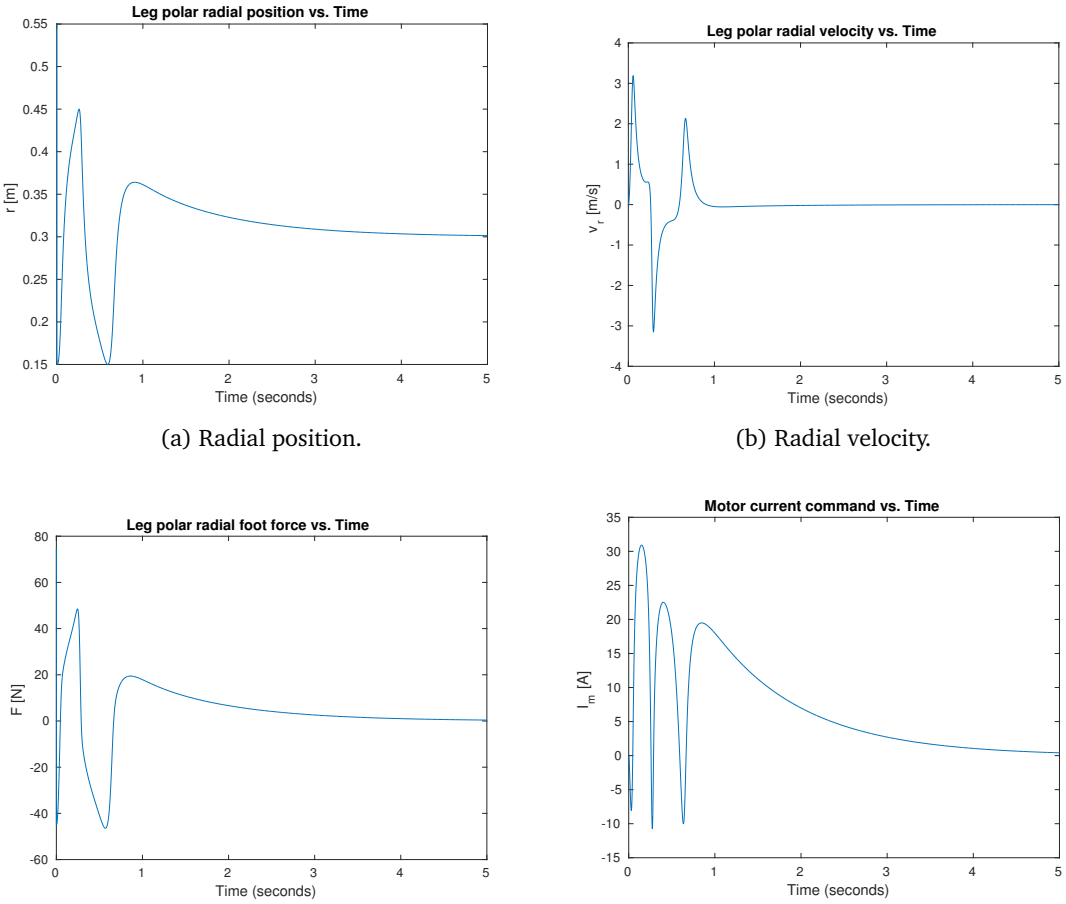


Figure 8.8: Motor simulation model.



(a) Radial position.

(b) Radial velocity.

(c) Radial force.

(d) Motor current.

Figure 8.9: Control system simulation plots.

9 Experimental Testing

"Data! Data! Data!' he cried impatiently. 'I can't make bricks without clay."'
— Sherlock Holmes in Arthur Conan Doyle, *The Adventure of the Copper Beeches*

9.1 Velocity Mapping vs. Backwards Difference

A number of numeric approximations exist for calculating the velocity mapping from one co-ordinate system to another.

In chapter 4 the kinematic equations for the leg system between $[r, \theta]$ and $[\phi_1, \phi_2]$ were shown. The Jacobian was also calculated using these kinematic equations.

Two methods for calculating the radial and rotational velocity of the leg foot were used and experimentally tested.

The first method, using the backwards difference approximation, can be seen in eq. (9.1), where i is the sample point index and t_s is the sample time of $5\ ms$. This equation relies on the fact that we know the radial and rotational distance of the foot using the kinematic mapping from Cartesian motor position to polar foot position.

$$\dot{r}(i) = \frac{r(i) - r(i-1)}{t_s} \tag{9.1}$$

The second method, using the Jacobian to map motor velocities to foot velocities, uses the equation derived in section 4.3.

9.1.1 Data Analysis

Initially the backwards difference was used as a quick and simple to implement approximation for velocity. Based on the experimental comparison, in fig. 9.1, it was clear to see the Jacobian mapping results in a much more accurate representation of the radial velocity.

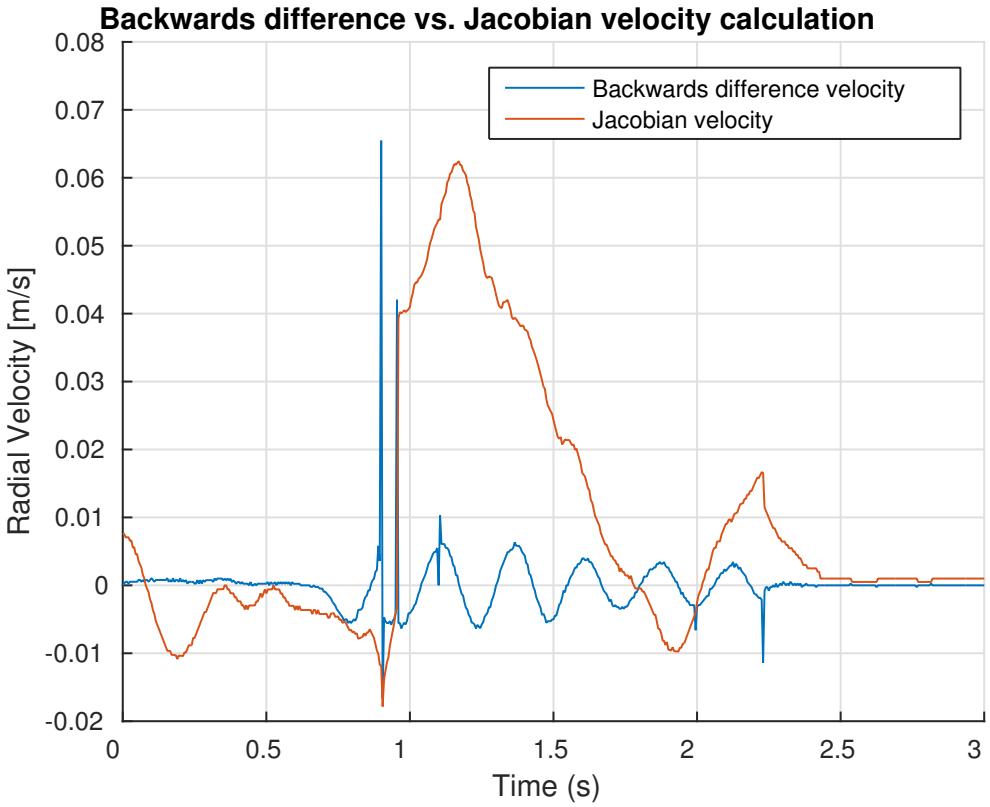


Figure 9.1: Comparison of backwards difference and Jacobian velocity calculation methods.

In further experimentation it was found that using the Jacobian derived velocities resulted in a much more smooth spring-damper motion because of the damper force calculation using both radial and rotational velocity calculations. The backwards difference is a relatively inaccurate way of calculating a derivative and was replaced with the Jacobian method for all calculations.

9.2 Force Control Calibration and Fidelity

Force control was implemented by using the relationship between motor torque and the torque constant K_t, as developed in section 8.5. The calibration method was further discussed in subsection 6.6.3.

The following experiment was developed to:

- Calibrate the torque constant K_t to achieve proprioceptive force control.

- Validate the transparency of the coupling between the motor torque and the end effector force.
- Confirm the linear relationship, with gradient K_t, between motor torque and current control input.

A load cell was used with a force display. Force values were manually logged while the radial set-point offset Δr was changed with the rubber foot placed against the load cell. By varying the radial set-point with the leg firmly attached to the linear guide, the radial set-point offset changes appropriately and the virtual model makes the system behave like a compressed spring with the corresponding force output.

A virtual compliance model spring constant of $K_s = 200\ N/m$ with zero damping was used which results in the theoretical radial force output derived in eq. (9.2).

$$F_r = K_s(r_{feedback} - r_{command}) \tag{9.2}$$

The theoretical radial force output using eq. (9.2) was plotted against the force values logged from the load cell.

9.2.1 Experimental Limitations

In future experimentation a digital load cell capable of continuous logging should be acquired. This was unavailable at the time of testing. The use of a load cell with manual logging meant that only static tests could be performed. In reality we are interested in the dynamic force estimation of the system during jumping.

The experiment used a limited number of set-point offsets due to logging constraints.

9.2.2 Data Analysis

Figure 9.2 shows various radial set-point offsets measured during the experiment. Three set points were used, and marked on the plot.

As can be seen in fig. 9.3, both the theoretical and measured forces follow a gradient closely matching each other. The theoretical force line crosses the origin, as expected, but the measured force line has an offset of $-5.0156\ N$. By calibrating the torque constant

until this offset is zero, we can match the theoretical and measured force measurements for true proprioceptive force control.

9.2.3 Summary

Based on the initial investigative questions, the following results were obtained:

- **Calibrate the torque constant K_t to achieve proprioceptive force control:** A value of $K_t = 0.08Nm/A$ was derived by iterative calibration of the force output. This was confirmed by [9] that determined a value of $K_t = 0.071Nm/A$.
- **Validate the transparency of the coupling between the motor torque and the end effector force:** Both the virtual model estimated force and load cell measured force followed the same gradient, within a margin of experimental error. As expected with a virtual spring constant of $K_s = 200N/m$ the theoretical force followed a gradient of 200, while the load cell force followed a gradient of 185.54 - this indicates limited mechanical impedance and dynamic interference between the motor and end effector coupling.
- **Confirm the linear relationship, with gradient K_t, between motor torque and current control input:** As covered in the previous point, the force gradients were near identical. Both the theoretical and measured force followed a linear pattern which confirms the linear relationship between motor torque and control current! This is sometimes not the case when non-linear dynamics are involved, which once again confirms transparency.

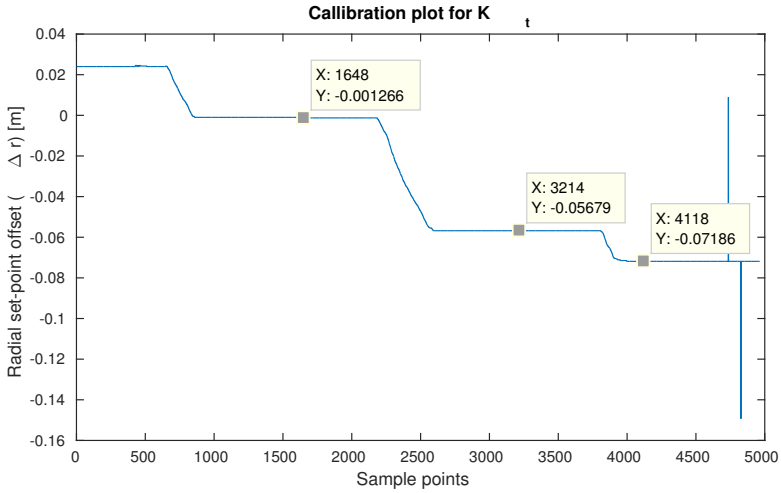


Figure 9.2: Calibration of K_t: plot of radial set-point offset over time.

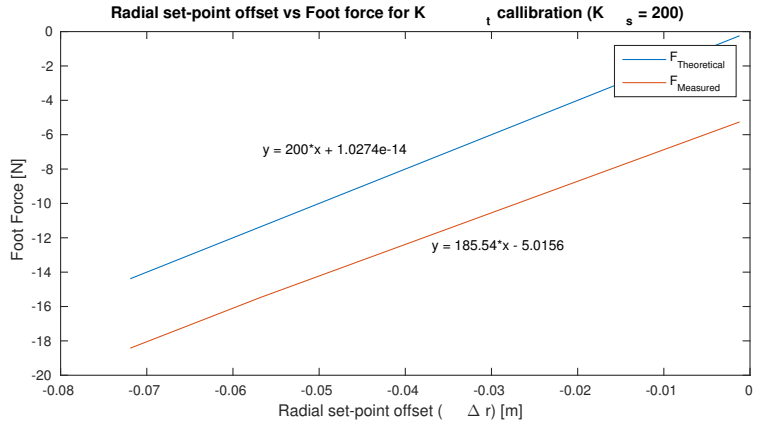


Figure 9.3: Calibration of K_t: plot of radial force (theoretical and measured) vs radial set-point offset.

9.3 Virtual Spring-damper Tests

The virtual spring-damper model, developed in chapter 5 and comprised of a linear and torsional spring-damper, was tested with various constants and in full-leg and joint spring-damper topology.

The experiment aimed to investigate:

1. Accuracy of theoretical virtual model design versus practical implementation.
2. Ideal spring-damper configuration for leg jump control.
3. Effect of virtual model spring-damper topology on leg dynamics.

The leg was fixed at a height of $0.4\ m$ and allowed to freely move in mid-air. The torsional spring-damper was set to a value of $K_s = 300\ N/m$ and $K_d = 10\ N/(m/s)$ in order to keep the leg from rotating. The purpose of the experiment was to investigate a single spring-damper pair, namely the linear spring damper.

For all tests the radial set-point, r_0, was set to $0.3\ m$ and offset by $r_{offset} = r - r_0 = 0.13\ m$ before being released to oscillate. The leg weight was measured as $0.5\ kg$ which served as an approximation for the virtual model mass in mid-air.

9.3.1 Full-leg Spring Damper Topology

The full-leg spring-damper model in fig. 5.1 serves as a reference for the experimentation performed.

Using eq. (5.12) to calculate the theoretical natural frequency of oscillation, ω_0, we get:

$$\begin{aligned}
&\text{Spring constant of 100 N/m:}\\
&\omega_0 = \sqrt{\frac{100}{0.5}} = 14.14\ rad/s\\
&\text{Spring constant of 200 N/m:}\\
&\omega_0 = \sqrt{\frac{200}{0.5}} = 20\ rad/s\\
&\text{Spring constant of 300 N/m:}\\
&\omega_0 = \sqrt{\frac{300}{0.5}} = 24.49\ rad/s
\end{aligned} \tag{9.3}$$

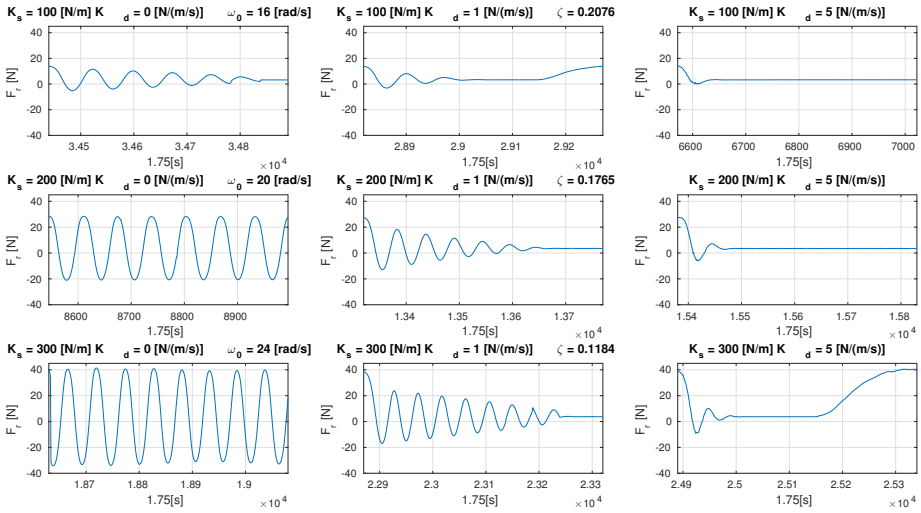


Figure 9.4: Full-leg spring damper testing for radial offset.

The experimental natural frequencies were all within a reasonable experimental error of the theoretical values. The natural frequency for a spring constant of $200\ N/m$ and higher seemed to be a higher fidelity representation of the virtual model. This is likely due to the mechanical impedance that exists in the leg design, as discussed in section 6.4, causing the relatively low force experiment for $K_s = 100\ N/m$ to not overcome the deadband that exists.

The mechanical impedance in question caused the $K_s = 100\ N/m$ foot force to decay rapidly whereas the higher spring constant tests continued to oscillate, thus better representing the virtual model and resulting in a natural frequency much closer to theory.

As the damping increased, the decay rate increased dramatically. For a damping of $K_d = 1\ N/(m/s)$ the damping ratio, ζ, was calculated. This was only calculated for a subset of the plots because the calculations were not theoretically accurate, as discussed in subsection 9.3.3. These damping ratios show a clear relationship between spring constant and damping constant, where eq. (5.9) defines the theoretical relationship. The damping ratio decreases with increasing spring constant, as expected.

The motor control plots in fig. 9.5 show accurate current command set-point tracking, symmetric position and velocity plots and a controlled decay in radial foot force. A maximum motor velocity of $100\ rad/s$ was experienced with a corresponding current command of $20\ A$ a radial foot force of $40\ N$.

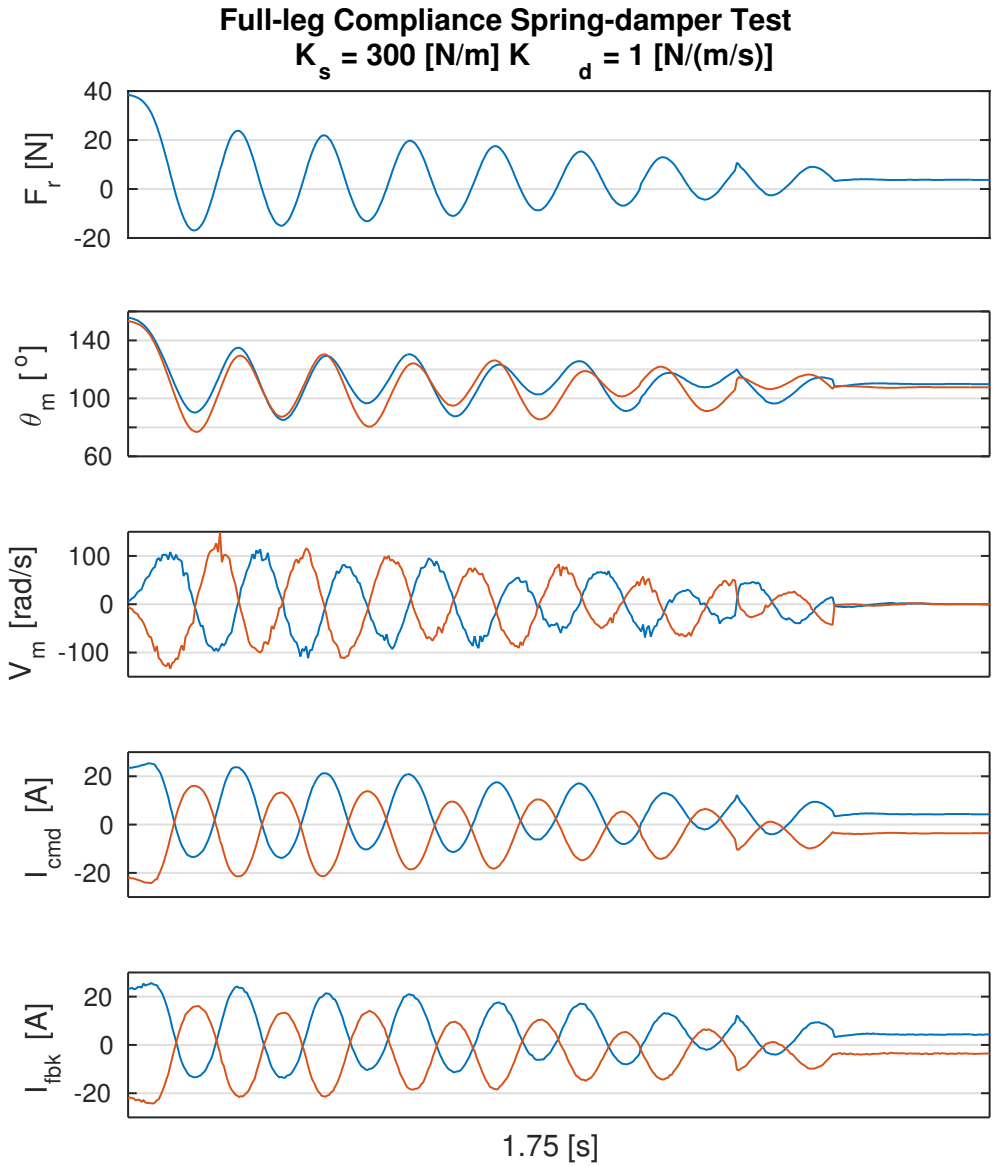


Figure 9.5: Full-leg spring damper motor control.

9.3.2 Joint Spring Damper Topology

The joint spring-damper model in fig. 5.2 serves as a reference for the experimentation performed.

A scaling factor of 0.01 was used for the joint compliance model spring and damping constants. This was done so that the constants were effectively normalised, rather than dealing with the small spring and damping constants needed to effect proper control.

Using eq. (5.12) to calculate the theoretical natural frequency of oscillation, ω_0, we get:

$$\begin{aligned} &\text{Spring constant of 300 N/m:} \\ &\omega_0 = \sqrt{\frac{300}{0.5}} = 24.49\ rad/s \\ &\text{Spring constant of 400 N/m:} \\ &\omega_0 = \sqrt{\frac{400}{0.5}} = 28.28\ rad/s \end{aligned} \tag{9.4}$$

The comparison of theoretical and experimental natural frequency is not as clearly defined when dealing with a joint spring damper model, although the experimental natural frequencies were within a reasonable margin of error of the theoretical values.

The motor control plots in fig. 9.7 show a relatively fast foot force decay with the current command for each motor offset significantly from zero. A maximum motor velocity of $60\ rad/s$ was experienced with a corresponding current command of $20\ A$ a radial foot force of $35\ N$.

9.3.3 Experimental Limitations

The undamped spring tests used to determine the natural frequency were difficult to perform as the leg oscillated unstably. In cases were it looked like the oscillation stopped, this was a case of having to forcibly stop the oscillation in order to prevent any damage to the platform.

The tests were performed before the velocity mapping was improved by using the Jacobian method instead of the backwards difference. This means the damping effect, although visible and functional, was not scientifically accurate when compared with the theoretical calculations. For this reason, the damping ratios were not included for most of the plots. The experiments rather served to theoretically compare natural frequencies,

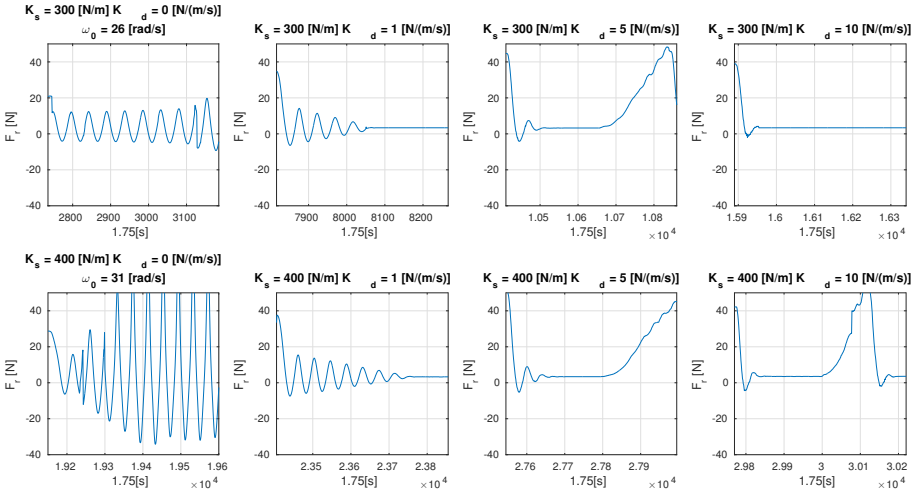


Figure 9.6: Joint spring damper testing for radial offset.

and to practically determine the best configuration for spring-damping during jump control.

9.3.4 Summary

The following conclusions were made based on the original investigative aims:

1. **Accuracy of theoretical virtual model design versus practical implementation:** The natural frequencies measured from the undamped systems closely matched the theoretical values. Damping had the expected effect on oscillations, but was realistically not scientifically sound because of mechanical impedance that exists in the leg design and the use of the backwards difference for velocity mapping.

2. **Ideal spring-damper configuration for leg jump control:** The ideal spring-damper configuration for reduction of oscillations during flight while maintaining rapid leg position recovery was determined to be approximately $K_s = 300\ N/(m/s)$ and $K_d = 5\ N/(m/s)$.

3. **Effect of virtual model spring-damper topology on leg dynamics:** Using full-leg spring damper control resulted in a more desirable mid-air motion and a faster recovery.

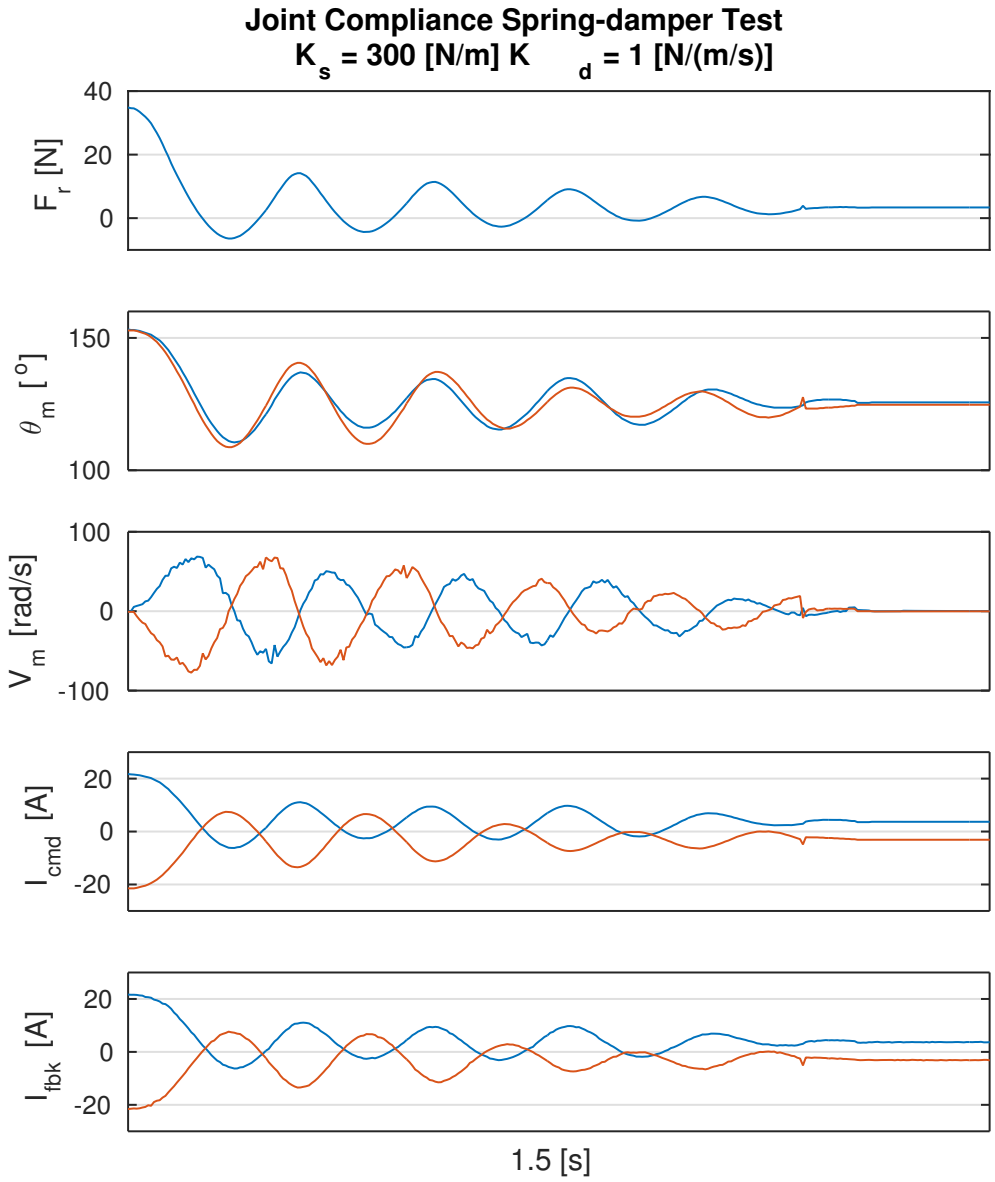


Figure 9.7: Joint spring damper motor control.

9.4 Drop Tests

The drop test experiments aimed to investigate the leg response to impact and the effect of spring constants and damping on the spring damper mass motion derived in section 5.2. The results of these tests lead to the final choice of spring-damper constants implemented in the virtual model and tested in the jump tests during the landing phase seen in fig. 9.9.

A theoretical value for the spring constant K_s, with $K_d = 5\ N/(m/s)$, for radial spring-damping upon impact was determined in section 5.3 using conservation of energy and an ideal dynamic response. The calculated spring constant of $632.8\ N/m$ was used as a starting point for testing and the damping was varied from $0\ N/(m/s)$ to $5\ N/(m/s)$ to confirm the values derived.

The drop tests plots of fig. 9.8 show a plot of radial foot force versus sample points at $5\ ms$ per sample, with the total plot duration shown for reference.

9.4.1 Experimental Limitations

The drop tests were performed from a maximum height of $0.5\ m$ and as such were limited to a subset of possible drop heights. The linear guide provided significant damping in the form of friction - this must be taken into account when analysing the data. In an ideal experiment the leg would oscillate continuously with zero damping, but in reality the oscillation were damped very quickly.

9.4.2 Data Analysis

With a spring constant of $K_s = 500\ N/m$ a damping constant of $K_d = 10\ N/(m/s)$ resulted in no overshoot, but significant undershoot.

The spring constant of $K_s = 800\ N/m$ served as a practical maximum due to the severe rebound experienced with zero damping - as the leg impacted the ground it oscillated twice mid-air before landing again. A damping value of $K_d = 20\ N/(m/s)$ provided a relatively smooth response with diminishing returns for higher damping constants.

Theoretically with a spring constant of $K_s = 500\ N/m$, and using the critical damping equation derived in section 5.2, a critical damping constant of $66\ N/(m/s)$ should result in no overshoot. Practically it was not possible to implement a damping constant higher

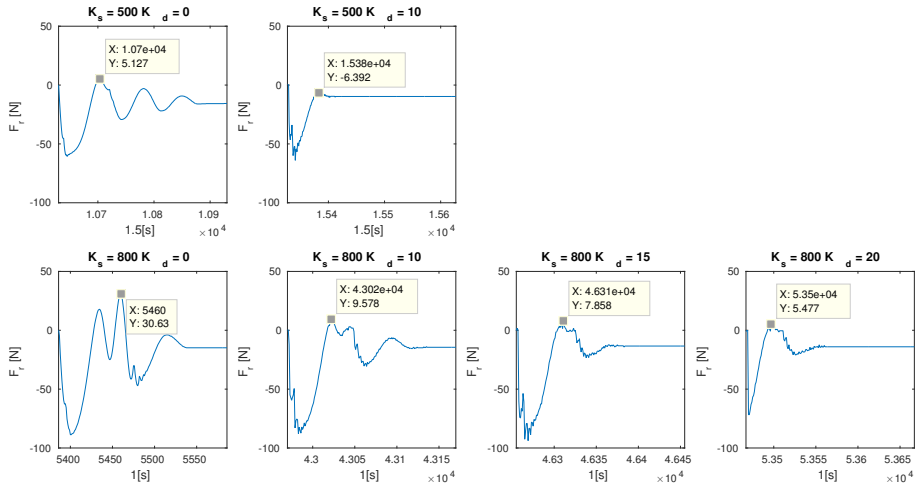


Figure 9.8: Leg spring damper drop testing.

than $30\ N/(m/s)$ as the high frequency noise was amplified, as determined in eq. (9.6), resulting in an unstable system.

9.4.3 Summary

Based on the experimental data analysis, the following conclusions can be made:

1. The system can adequately dissipate the energy from high speed impacts.
2. The effect of damping can not be fully tested due to its high frequency noise amplification characteristic which causes an unstable system.
3. The spring-damper virtual model behaves as expected on impact.
4. The spring constant values between $K_s = 500\ N/m$ and an upper limit of $800\ N/m$ can be used for impact absorption during jump experimentation.

9.5 Jump Test

A summary of the jump phases and experimental setup can be seen in fig. 9.9. The radial foot forces experienced at each jump phase are included as well as the radial and rotational spring-damper constants. The development of this experiment will be expanded upon in more detail.

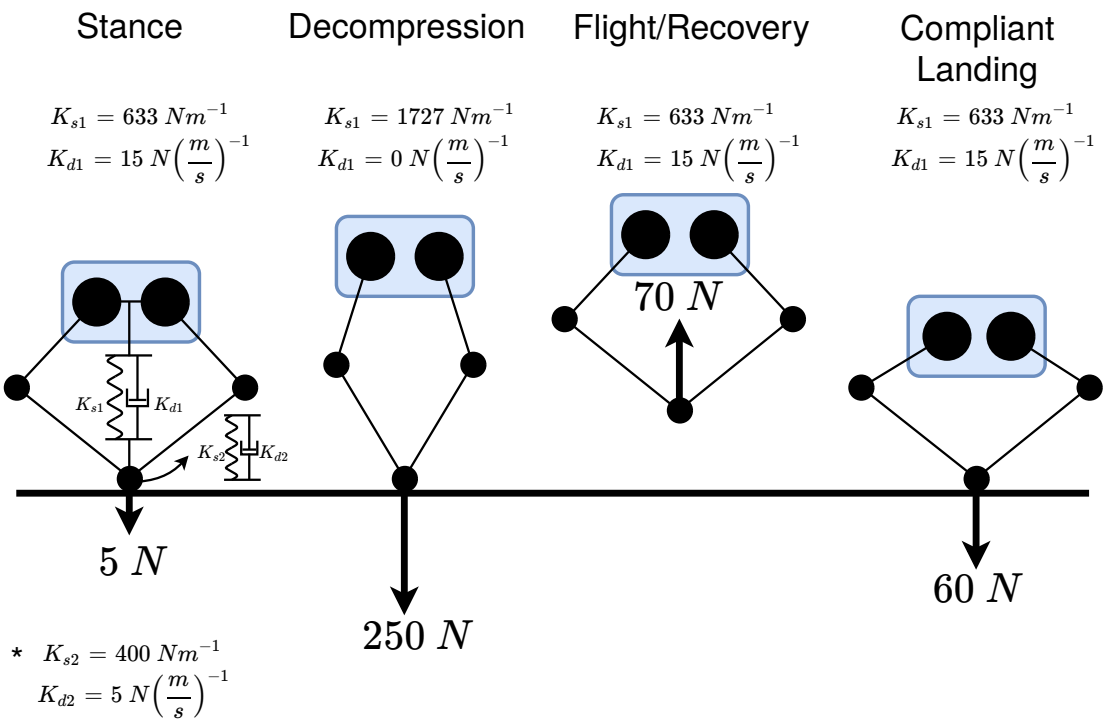


Figure 9.9: Jump phases and experimental setup.

Phase	K_{s1} $[Nm]$	K_{d1} $[N/(m/s)]$	K_{s2} $[N/m]$	K_{d2} $[N/(m/s)]$	r $[m]$
Stance	632.8	15	400	5	0.25
Decompression launch	**1726.6**	**0**	400	5	**0.4**
Recovery	632.8	15	400	5	0.3
Compliant landing	**632.8**	15	400	5	**0.3**

Table 9.1: Leg launch virtual model configuration.

9.5.1 Configuration of Jump Phase Parameters

Spring-damper mass motion of the leg model was theoretically developed in section 5.2. Ideal spring-damper constants for flight, impact and launch were developed using conservation of energy relationships and spring-damper dynamic equations.

The table of spring-damper constants and other configuration parameters, table 9.1, was found for ideal jump phase dynamics relating to the spring constants in fig. 5.1.

Parameters were derived both experimentally and theoretically. The theoretical (orange), drop test experimental (blue), and spring-damper test experimental (red) configuration parameters were highlighted appropriately.

9.5.2 Finite State Machine and Triggering

Listing 9 in appendix E shows the code implementation of a finite state machine that determines which spring-damper parameters are used in which phase of jumping. A similar finite state machine virtual model control method was considered in [15] for phase changes.

Using *if statements* various conditions were tested to determine the phase of jumping. Leg radial distance, foot force and time were used as conditions for phase change points.

Theoretically a more reliable method for determining ground contact would have been to use a foot trigger or a distance sensor. Both of these methods were attempted, but when it came to testing neither were reliable. The foot trigger was subjected to multiple impacts and was not mechanically robust. Due to the long cable from the foot trigger to the microcontroller, and possibly the noise from motor EMI, caused the foot trigger to be triggered prematurely causing damaging mid-air high gain force commands. The distance sensor beam width was too wide and due to the construction of the platform did not reliably detect the distance of the body from the platform.

The most reliable method proved to be using the radial set-point to determine when the leg was compressed after impact.

9.5.3 Testing Platform Setup

The testing platform designed in subsection 6.2.3 was used during the jump experiments. It was securely mounted against a vertical backdrop to ensure it was steady during jumps. During tuning of the leg prior to jumps it was secured on the linear guide before being allowed to freely rest on the base. The sandpaper and rubber foot ensured the foot did not slip before launching and during impact.

9.5.4 Video Data Extraction

In appendix E the data extracted from the video relating to body height, velocity, and phase versus time is included to show the data used to plot figs. 9.13 to 9.15.

The video was filmed at $24\ fps$. The testing platform linear guide has markers on it indicating height. For each frame the relevant height data, measured from the original resting height of the leg, was recorded along with the time. Using the backwards difference the velocity and acceleration for the interval was calculated - as such the values must be critically analysed.

9.5.5 Experimental Limitations

The jump experiment was limited by the motor driver limit of $60\ A$. Foot force is related to motor current - to achieved height control or accurate foot force impulses a larger current limit is needed. During experimentation the current saturated immediately during jumping, showing that a larger current range was needed.

When two consecutive high energy jumps were attempted a motor driver current cut-out was triggered. A video extract of this action can be seen in appendix E fig. E.1.

To perform multiple consecutive jumps as seen in section 9.6, a lower energy jump was performed with a maximum jump height of $0.25\ m$.

9.5.6 Data Analysis

Launch Phase

In order to achieve as rapid a vertical acceleration as possible the motor current needs to be saturated. Figure 9.12 shows that as the launch phase starts the current command is saturated at just under the motor driver peak current of $60\ A$ to avoid current cut-out.

The radial foot force of fig. 9.11 is directly related to the difference between the radial set-point, summarised in table 9.1, and the actual leg radius feedback. The foot force instantaneously increased from a magnitude of near $0\ N$ to a maximum of $-270\ N$ - in reality the foot force during this transition may be more slow to react, as the foot force is theoretically determined using the virtual model.

The launch phase has a duration of $10\ ms$ before the next control phase takes over. As the leg radius extends beyond $0.38\ m$ the next phase, the flight phase, is triggered as expected.

Flight Phase

At this point the current command switches polarity quickly to restore the leg position for landing. The leg radius is restored within $15\ ms$ to the radial set-point of $0.3\ m$ with no overshoot or steady state error. This validates the control method and virtual model parameters for the flight phase. In three sample periods the controller managed to accurately command a kinematic set-point.

During the free-fall phase the downwards acceleration of the body had a mean value of $9.51\ m/s^2$ as seen in fig. 9.15. This is expected, with a gravitational constant of $9.81\ m/s$ and a coefficient of kinetic friction as calculated in section 6.4, the mean body acceleration should be less than that of gravity.

The impact phase occurs approximately $20\ ms$ later leaving lots of room for error in leg restoration.

Landing Phase

During the landing phase the radial length of the leg decreased to a minimum of approximately $0.24\ m$, as seen in fig. 9.10. This was a relatively severe undershoot, with a

set-point of $0.3\ m$. The radius recovered to $0.28\ m$ within $15\ ms$ or three sample points. This shows that during impact a lot of force is experienced. The spring-damper constants should be increased in future to account for this. The impact was absorbed without oscillation, and so was still stable as designed.

Figure 9.12 shows the current feedback versus time. The current was very oscillatory. Increasing the control sampling frequency would have dealt with the high speed impact with more accuracy and perhaps have prevented the radial undershoot.

Although the foot force during impact was relatively small compared to the launch phase, the current was at near it's maximum upon impact. This shows the effect of saturation. At launch, the leg wanted to output $-270\ N$ of force, and would have done so without the current saturation, but in practise it probably experienced a much lower foot force. If a foot force plate was available for measurement, the true foot force could have been determined.

Foot Placement

In the plot of radial and angular foot force versus time in fig. 9.11, it can be clearly seen that the angular force makes up very little of the total foot force. This shows that the control system is accurately keeping the foot at a zero angular set-point and that the launch and landing phases put very little stress in the angular direction.

Figure 9.17 confirms the accurate angular foot placement where extracted video frames show minimal angular movement, with the linear guide as a reference point. Even during transitions between phases, where the most angular movement is expected, very little was experienced.

Jump Energy

The theoretical jump energy can be calculated using eq. (9.5).

$$E_{jump} = mgh_{max} \times \frac{1}{Efficiency[\%]} \tag{9.5}$$

Assuming the jump is $100\ \%$ efficient and the maximum jump height logged of 0.4 is used, the jump energy is $8.63\ J$. This jump energy is compared to other similar legged robots in table 10.1.

Control Sampling Time

The controller performed well considering that many of the critical sections of the jump phases occurred in under $20\ ms$ or 4 sample periods. A higher sampling frequency, in the kHz time-scale would result in a much higher fidelity virtual model, but in practise the sampling frequency of $200\ Hz$ worked well both for jump experimentation and the spring-damper tests of section 9.3.

9.5.7 Summary

Based on the experimental data analysis, the following conclusions can be made:

1. The motor driver peak current limit of $60\ A$ severely limits the amount of force available during launching. Theoretically a $-270\ N$ force would have been experienced with a current saturation of around $150\ A$, but in practise it was much lower and limited how much control could be had over impulse forces and how high the leg could jump.
2. At a $200\ Hz$ sampling rate, once again limited by the motor driver response time, high fidelity virtual model control is difficult to achieve. In practise many of the critical phase sections were less than $20\ ms$, leaving 4 sample points for the controller to effect the necessary commands.
3. The foot placement during launch and recovery was accurate and well controlled. During impact a higher spring constant and damping were required to achieve less undershoot, but no oscillation was experienced.
4. Considering the current limit of $60\ A$, the energy delivered, $8.63\ J$, compared favourably to the GOAT leg, $20.11\ J$ with three motors, a higher current limit, and custom controllers and gearing.

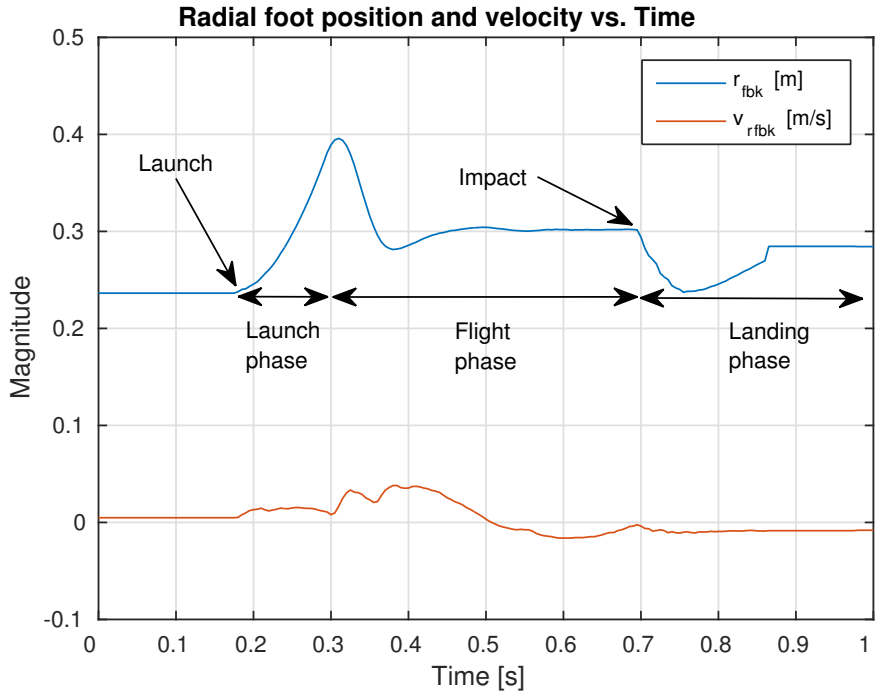


Figure 9.10: Jump foot radial position and velocity (launch and compliant landing).

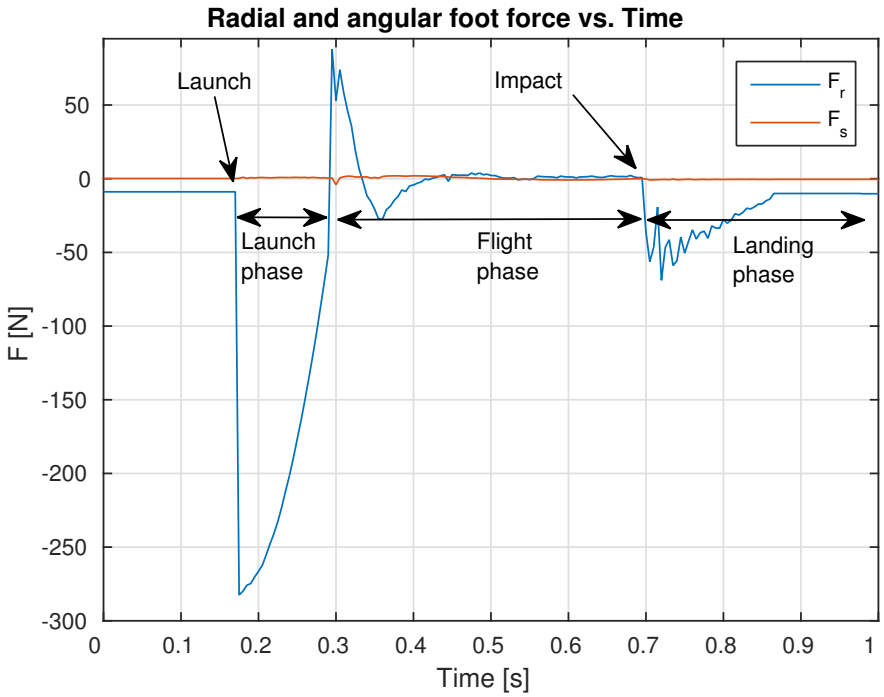


Figure 9.11: Jump foot force output (launch and compliant landing).

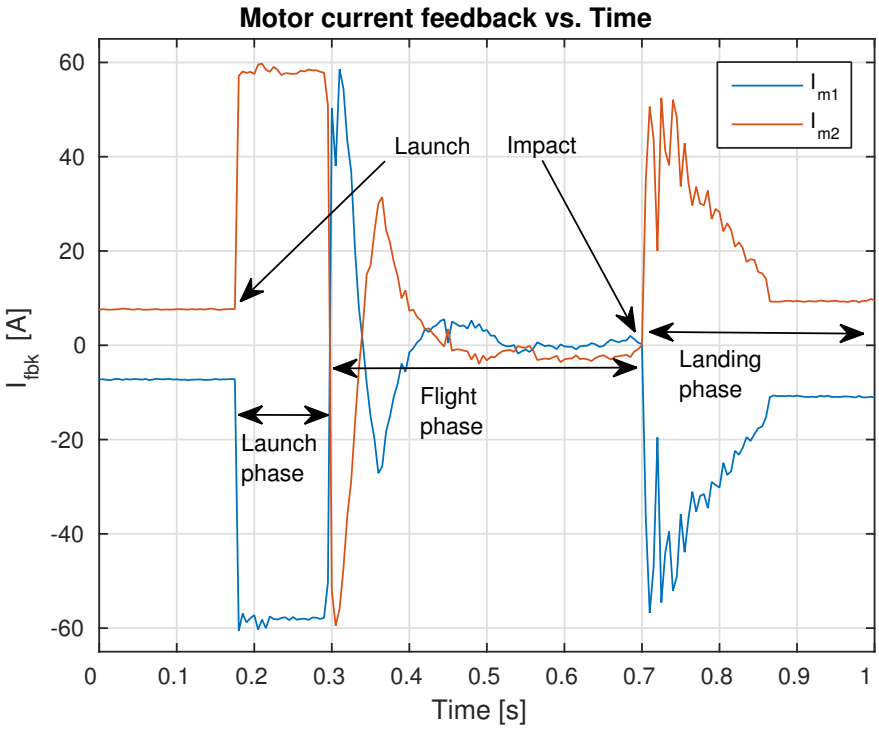


Figure 9.12: Jump motor current feedback (launch and compliant landing).

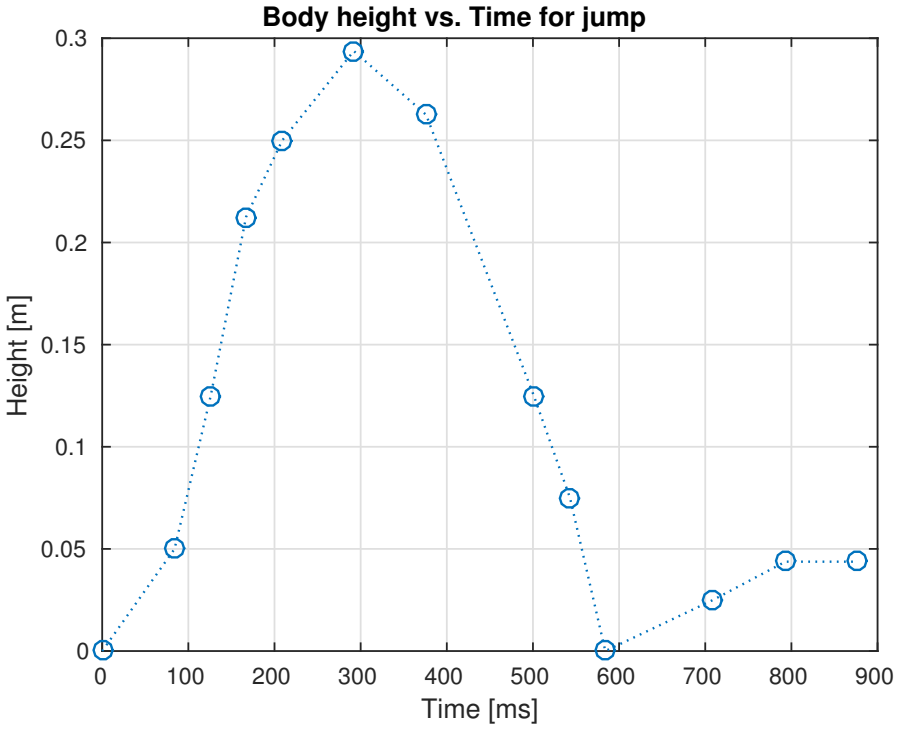


Figure 9.13: Height vs. time plot relative to body starting position.

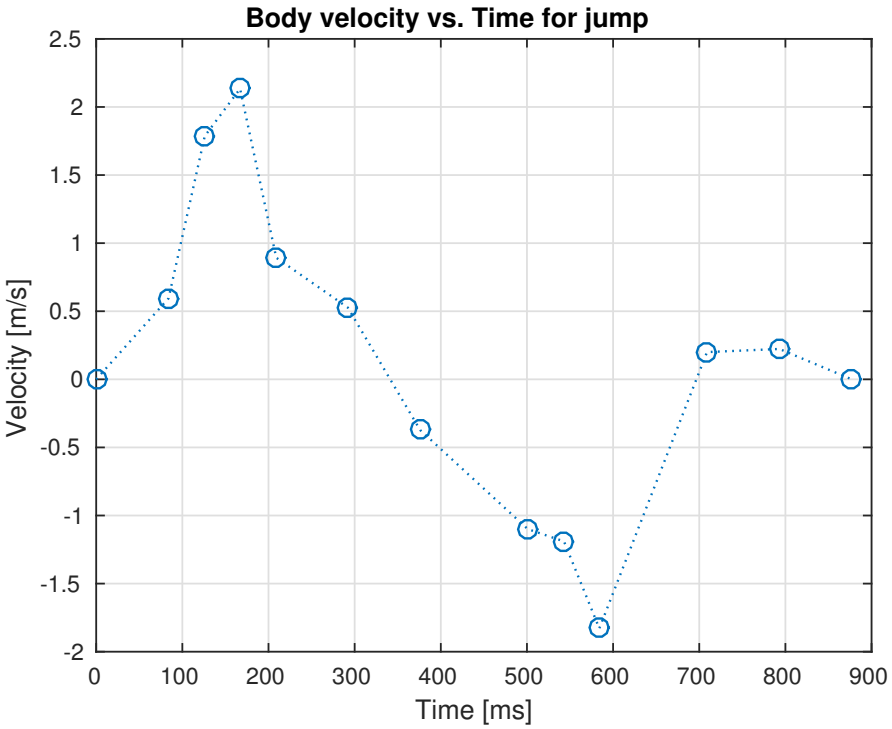


Figure 9.14: Velocity vs. time plot relative to body starting position.

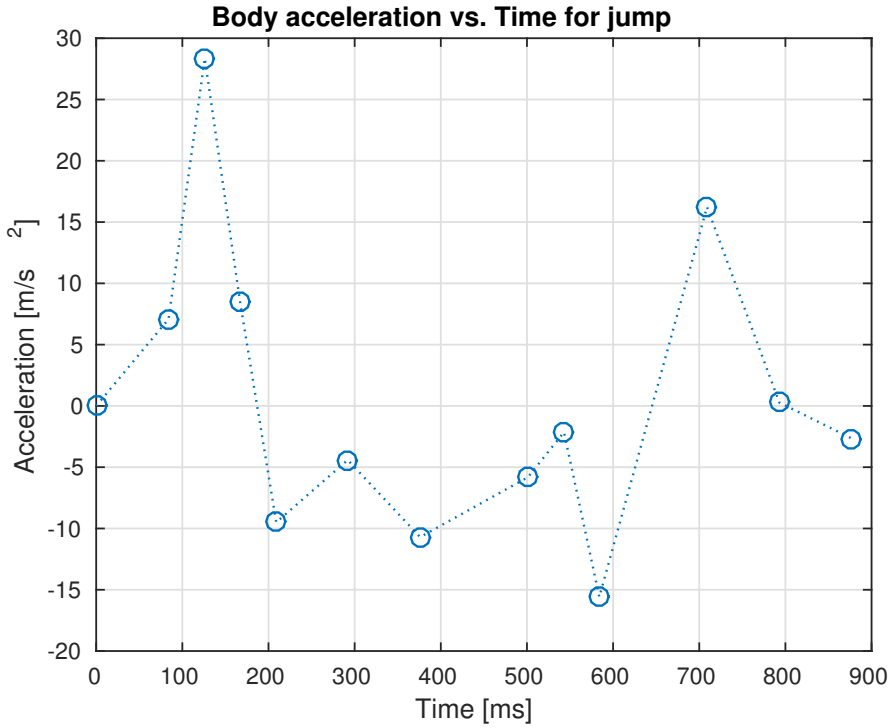
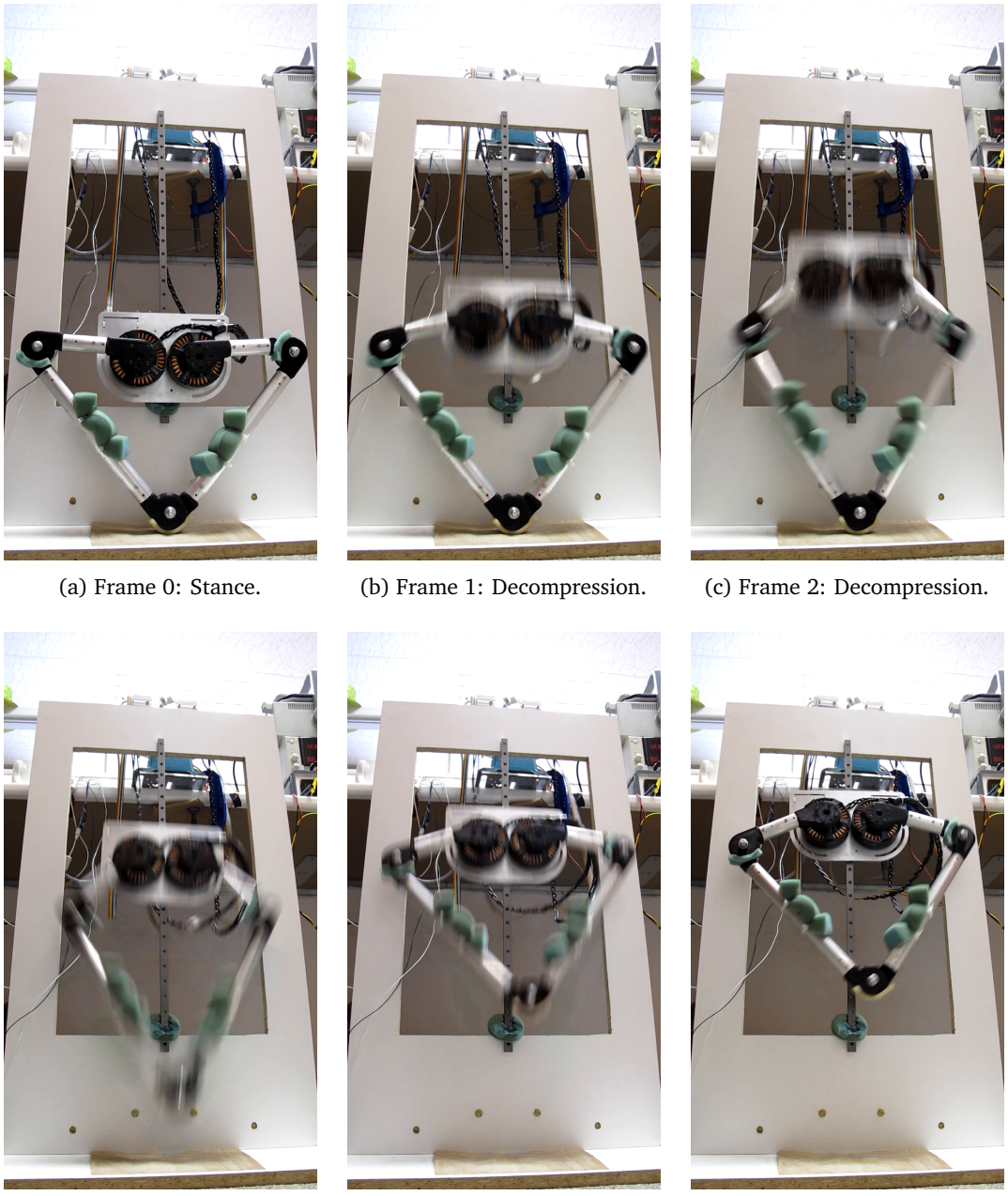


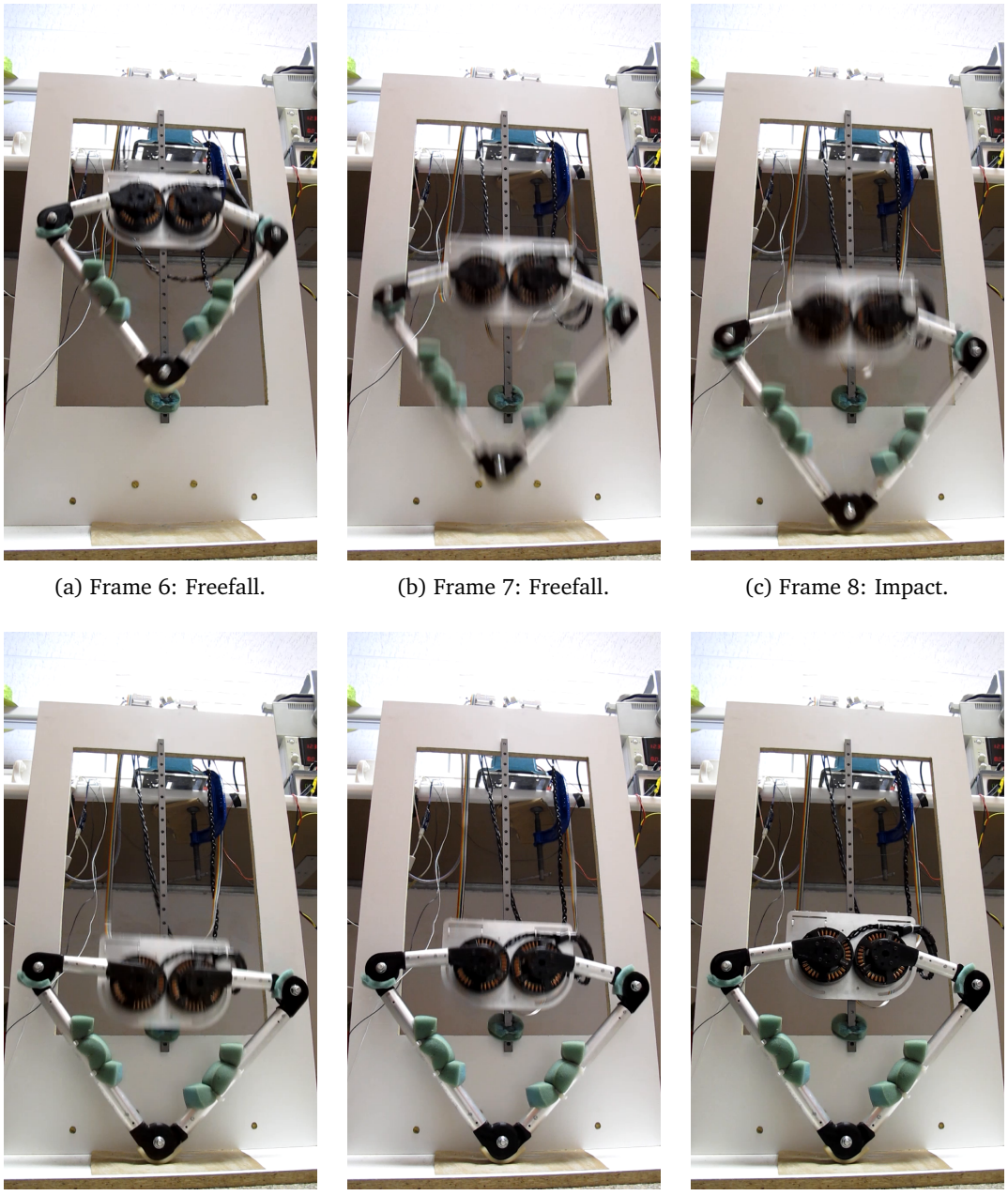
Figure 9.15: Acceleration vs. time plot relative to body starting position.



(a) Frame 0: Stance. (b) Frame 1: Decompression. (c) Frame 2: Decompression.

(d) Frame 3: Flight. (e) Frame 4: Flight/recovery. (f) Frame 5: Flight/recovery.

Figure 9.16: Jump: Stance and Flight Phase.



(a) Frame 6: Freefall. (b) Frame 7: Freefall. (c) Frame 8: Impact.

(d) Frame 9: Compliant landing. (e) Frame 10: Compliant landing. (f) Frame 11: Compliant landing.

Figure 9.17: Jump: Freefall, Impact and Compliant Landing Phase.

9.6 Consecutive Jump Repeatability

In order to test the platform's design with robustness and repeatability as performance measures, multiple consecutive jump were performed without interfering with the control system or platform in any way.

The leg was mounted on the platform in a central foot position before the launch sequence was initialised. Seven consecutive jumps were performed to a moderate height of $25\ cm$.

A plot of foot force versus time was created with all seven plots overlayed on the same set of axis, as seen in fig. 9.18. The starting points of the jumps were synchronised to show changes in jump patterns.

9.6.1 Data Analysis

A maximum foot force deviation of $32\ N$ was seen for jump 4 with the mean being a foot force of $130\ N$.

A maximum time phase shift of $0.15\ s$ was seen between jumps, over a total mean jump duration $0.7\ s$ - so the time deviation was a $21\ \%$ change.

The foot forces all settled at approximately the same value of $8\ N$ before dropping to $-10\ N$ in preparation for the next jump. This pattern was consistent.

Both of the above deviation measurements were thrown off by the outlier of jump 4. If this jump is ignored due to experimental error, then the maximum deviation in time is approximately $0.06\ s$, or 8.57%, and for force is negligible.

9.6.2 Summary

The experimental data analysis showed two results:

1. The leg control system is capable of **robust** force control with a **negligible deviation** for force and time.
2. The construction and design of the robotic leg platform enables **reliable and repeatable force control** to implement consecutive jumping sequences.

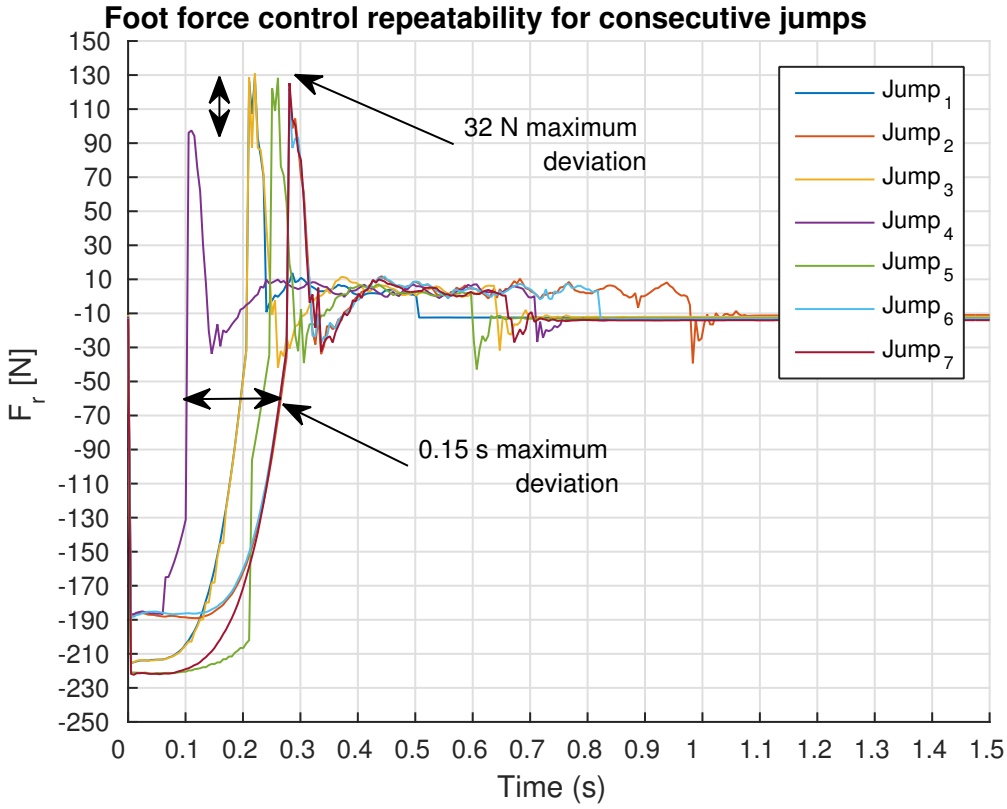


Figure 9.18: Consecutive jump foot force to investigate repeatability.

9.7 Current Tracking

For high fidelity force control using a virtual model to be achieved, two things need to be ensured:

1. Transparent and efficient torque coupling between motor and linkages.
2. Accurate high speed current tracking during dynamic movements.

The transparency and efficiency of the torque coupling is almost entirely a mechanical specification. By using a direct drive motor system this is mostly taken care of.

Accurate high speed current tracking ensures that when an impact occurs, the foot force will respond appropriately and absorb the impact energy (assuming a spring-damper virtual model).

9.7.1 Data Analysis

Figure 9.19 shows a plot of current feedback from motor driver 1 versus the current command from the force controller - during an impact after the jump test in section 9.5. The force command is ideally directly related to the torque constant, K_t, and Jacobian force mapping derived in section 8.5.

In fig. 9.19 at the point of impact, the current command leads the current feedback in a sudden negative drop. The current feedback tracks the command very well with the expected one sample delay. In reality, the current tracking may be better than we are able to measure - if the current could be measured directly after the command this would give a better idea of the true performance.

During the recovery period from $0.1\ s$ to $0.2\ s$ the set-point tracking is consistent and stable. After the current command settles we can see that there is zero steady state error.

9.7.2 Summary

The current **reacts quickly** to sudden high magnitude changes. Set-point tracking is at most **delayed by one-sample**, $5\ ms$, and **tracks consistently**. **No error** exists during steady state.

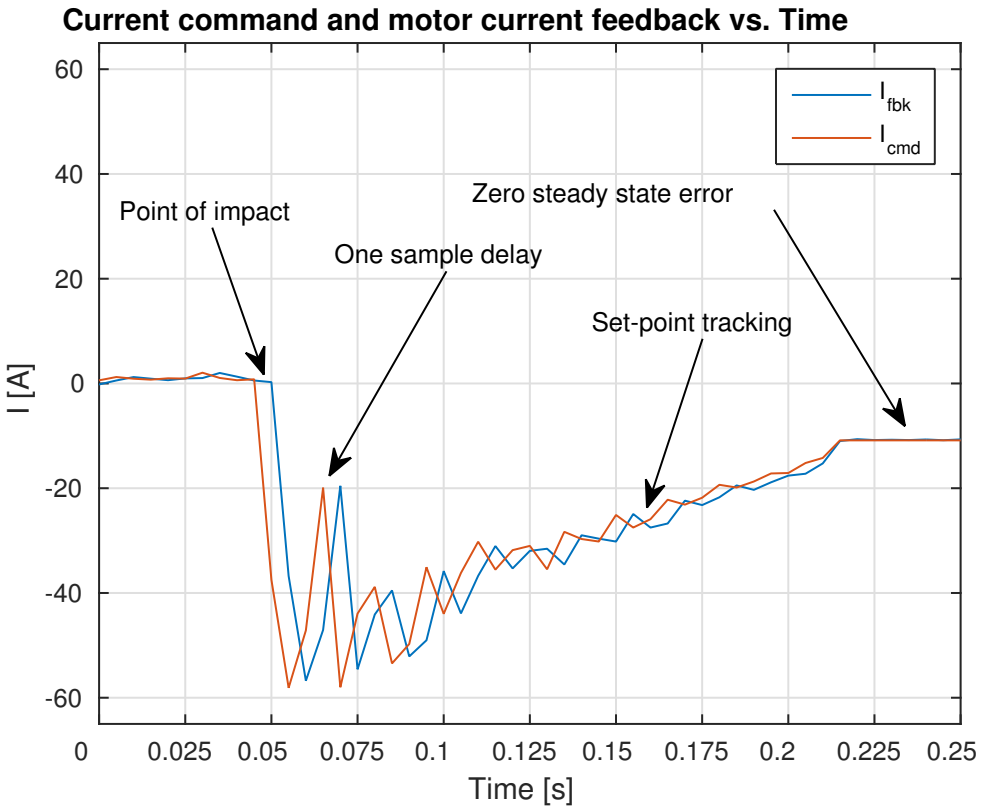


Figure 9.19: Current control tracking.

9.8 Trajectory Tracking

The trajectory tracking experiments were performed to investigate the following:

1. Effect of spring-damper constants on tracking performance, and to determine the best combination for trajectory tracking.
2. Linear distortion caused by simplified kinematic mapping model.
3. Effect of integral gain on tracking correlation, and to determine the best integral gain for accurate trajectory tracking while validating the integral control method.
4. Capability of leg to perform complex movements for three dimensional running trajectories.

To fully test the leg's movement capabilities, both the maximum and minimum radial lengths of $0.45\ m$ and $0.25\ m$ respectively were used for the y range. In order to generate a circular and square trajectory, the x range was set from $-0.1\ m$ to $0.1\ m$.

The Matlab programming environment was used to map the Cartesian trajectory coordinates to the leg's polar coordinates.

The experiments were performed over 10 second periods by using a lookup table comprised of 2000 sample points, with the control loop running at $200\ Hz$.

In all cases the same spring, K_s, damper, K_d, and integral, K_i constants were used for both the torsional and linear virtual model spring-dampers - as was the case in previous experiments, a scaling factor of 0.1 was used for torsional spring-damper constants.

The limitations of this experiment are that only two trajectories were tested at a constant speed. For high speed trajectories further experimentation will be needed.

9.8.1 Circular Path

A circular trajectory was chosen in order to test bio-inspired leg movement which usually follows a semi-circular trajectory.

Creffig:Trajectory tracking for circular path shows a clear relation between spring constant and accurate trajectory tracking, while the damper constant has a less noticeable effect.

The damper constants were varied between 1 and $20\ N/m$. The reason for the upper limit is that any value above $20\ N/m$ amplified the high frequency noise to an unstable

level. The cause of this effect can be seen by looking at eq. (9.6):

$$X(j\omega) = j\omega \tag{9.6}$$

In eq. (9.6) at high gains the high frequency noise is amplified and can be physically seen by the increased foot position vibrations beyond a value of $K_d = 20\ N/m$.

During slow speed movements the leg experiences significant joint friction, gravitational forces and leg inertia - these effects cause a deadband that the force controller has to overcome. By increasing the spring constant there is more force being supplied during the movement and the trajectory is more accurately tracked, so long as the damping constant is increased proportionally.

The effect of gravitational forces can be seen by the offset that exists between the feedback and the trajectory near the minimum radius of $0.25\ m$. At this point the leg radius is greater than the set-point due to the forces that need to be overcome to raise the leg and reduce the error.

The ideal spring constant for trajectory tracking was found to be $K_s = 800\ N/m$. The trajectory tracking above this spring constant value proved to be unstable and prone to motor driver current cut-out.

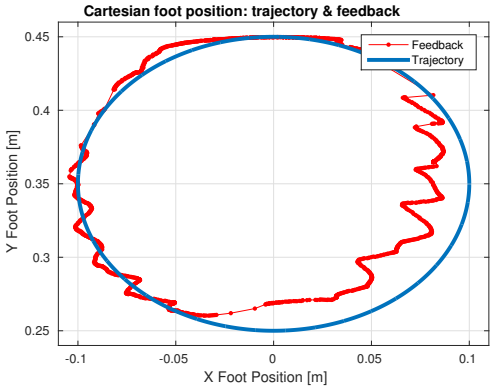
9.8.2 Angular Path

An angular path was primarily chosen to investigate the linear distortion caused by a simplified kinematic mapping model. The leg model was simplified in chapter 4 by assuming the motors were coaxial. This makes computation more manageable on an embedded system, with the possible disadvantage of distorting the leg trajectory - this experiment aimed to test that.

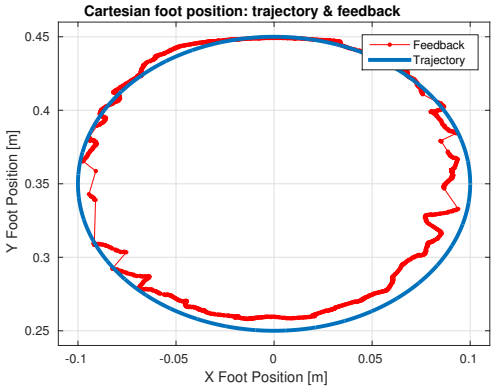
The sharp angles of a square also test the leg's rapid direction changing capabilities. In the case of three dimensional movement this experiment would be critical to understanding the leg's motion during sharp turns, like in the case of a Cheetah.

The ideal spring constant for trajectory tracking, as determined in subsection 9.8.1 to be $K_s = 800\ N/m$, was used, and damping constants of $K_d = 10$ and $20\ N/m$ were used along with an integral gain of $K_i = 500$.

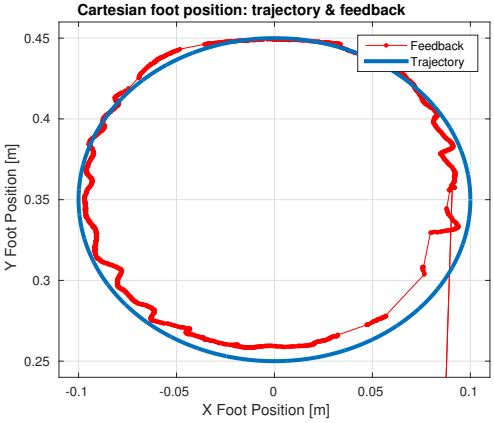
The linear distortion can be most clearly seen along the top edge of the square, at a maximum radius of $0.45\ m$. This distortion proved to be insignificant at an error of



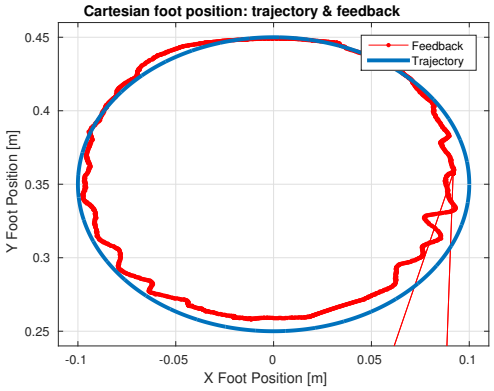
(a) $K_s = 300\ N/m$; $K_d = 5\ N/(m/s)$



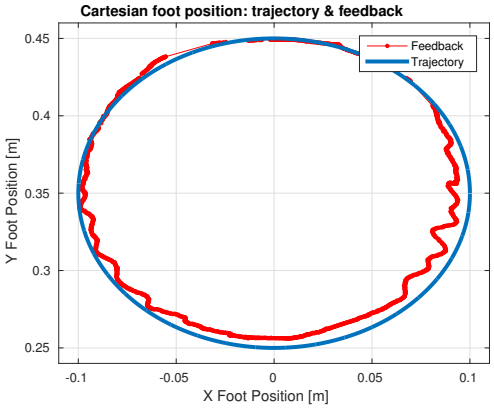
(b) $K_s = 600\ N/m$; $K_d = 1\ N/(m/s)$



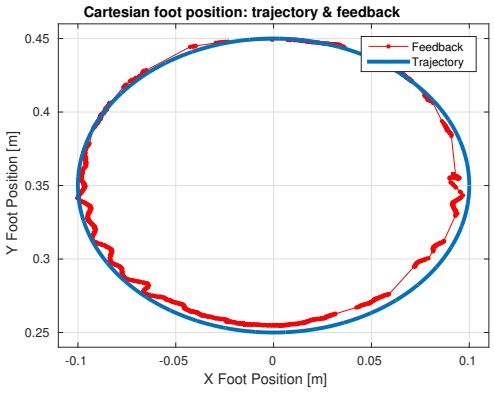
(c) $K_s = 600\ N/m$; $K_d = 10\ N/(m/s)$



(d) $K_s = 600\ N/m$; $K_d = 20\ N/(m/s)$

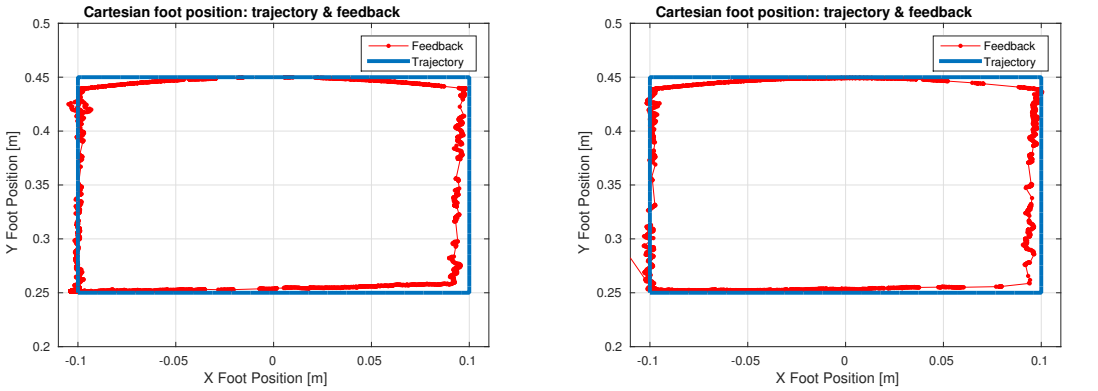


(e) $K_s = 800\ N/m$; $K_d = 20\ N/(m/s)$



(f) $K_s = 1000\ N/m$; $K_d = 20\ N/(m/s)$

Figure 9.20: Trajectory tracking for circular path to investigate effects of spring-damper constants on tracking.



(a) $K_s = 800\ N/m$; $K_d = 10\ N/(m/s)$; $K_i = 500$ (b) $K_s = 800\ N/m$; $K_d = 20\ N/(m/s)$; $K_i = 500$

Figure 9.21: Trajectory tracking for angular path to investigate linear deformation.

$0.01\ m$ at its maximum at the corners.

Asymmetry in motor movement was also discovered in the experiment. Along the right edge a relatively consistent offset of $0.01\ m$ exists. This could be caused by any number of reasons, including motor driving current loop tuning errors, or mechanical differences between leg linkages. This asymmetry can be easily corrected in the control loop by scaling one of the motor current commands until near zero error is achieved, or until both trajectory sides match. In practise this asymmetry has little effect on leg jumping performance, but for more complex dynamic movement it should be investigated.

9.8.3 Integral Gain Trajectory Correlation

Using the ideal spring constant for trajectory tracking, $K_s = 800\ N/m$ as determined in subsection 9.8.1, and with a damping constant of $K_d = 20\ N/m$, the experiment was performed using a circular trajectory.

Using Matlab the Pearson correlation values for the x and y Cartesian axis, ρ_x and ρ_y, were calculated. The circular trajectory was first linearly interpolated so that the trajectory and feedback vectors for the x and y axis were of the same length. A correlation value of $\rho = 1$ indicates perfect trajectory tracking with zero error, whereas a correlation of $\rho = 0$ indicates no trajectory tracking.

In table 9.2 a general upward trend in correlation can be seen for increasing integral gain. Any divergence from this trend is due to experimental error.

Integral gain clearly has a positive effect on tracking correlation and reducing steady

K_i	ρ_x	ρ_y
5	0.6716	0.4463
10	0.63109	0.48778
200	0.84726	0.632237
500	0.86448	0.72518
800	0.82182	0.72223

Table 9.2: Cartesian Pearson correlation values for a circular trajectory with varying integral gain.

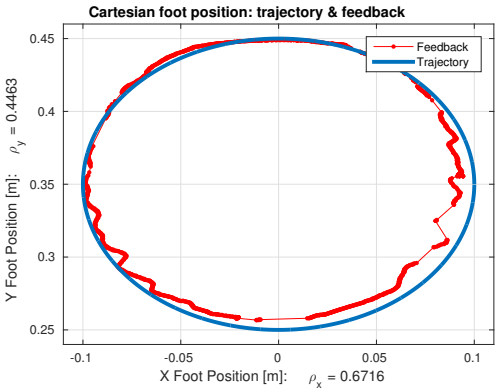
state error. The integral error control method was only successfully implemented in this experiment after noticing the error in tracking performance. This experiment proved to validate this control method. In previous experiments for jumping and spring-damper tests the steady state error was not necessarily critical to the experiment, and even in the case of trajectory planning the error was never greater than $0.01\ m$.

The point of diminishing performance returns for integral gain was $K_i = 500$. The maximum integral gain value of $K_i = 800$ was determined based on instability in the control system for greater integral gains. Based on this we can say the ideal integral gain for stable trajectory tracking is $K_i = 500$.

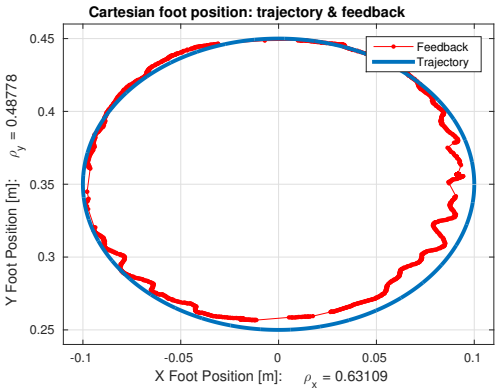
9.8.4 Summary

The following conclusions were made based on the original investigative questions:

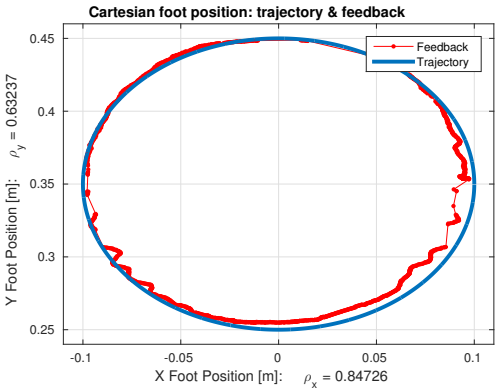
1. **Effect of spring-damper constants on tracking performance, and to determine the best combination for trajectory tracking:** The spring-damper constants had minimal effect on tracking performance once the initial force deadband was overcome - this resulted in an ideal spring-damper combination of $K_s = 800\ N/m$ and $K_d = 20\ N/m$.
2. **Linear distortion caused by simplified kinematic mapping model:** The kinematic mapping model created a slight warping of the square trajectory at an extended radial set-point. This warping was insignificant at $0.01\ m$ at maximum.
3. **Effect of integral gain on tracking correlation, and to determine the best integral gain for accurate trajectory tracking while validating the integral control method:** The integral error control method was successful in reducing error in



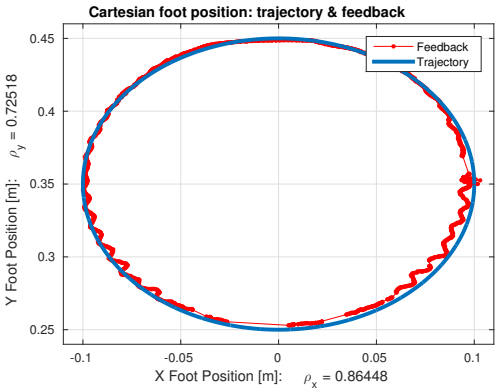
(a) $K_s = 800\ N/m$; $K_d = 20\ N/(m/s)$; $K_i = 5$



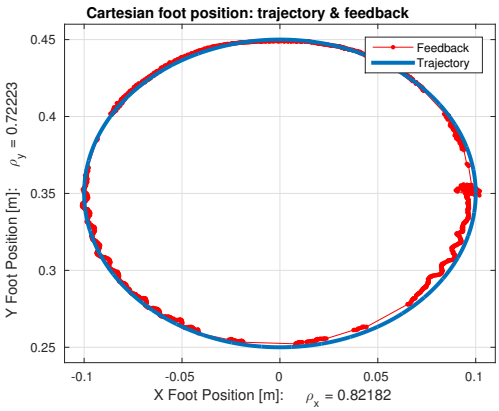
(b) $K_s = 800\ N/m$; $K_d = 20\ N/(m/s)$; $K_i = 10$



(c) $K_s = 800\ N/m$; $K_d = 20\ N/(m/s)$; $K_i = 200$



(d) $K_s = 800\ N/m$; $K_d = 20\ N/(m/s)$; $K_i = 500$



(e) $K_s = 800\ N/m$; $K_d = 20\ N/(m/s)$; $K_i = 800$

Figure 9.22: Trajectory tracking for increasing integral gain with x and y correlation.

trajectory tracking. The ideal integral gain was $K_i = 800$ for stability and performance.

4. **Capability of leg to perform complex movements for three dimensional running trajectories:** The leg was capable of performing both angular and circular trajectories with considerable speed. The trajectory tracking capabilities of the leg make more complex running trajectories possible, with insignificant distortion.

10 Discussion

10.1 Design Validation

10.1.1 Robotic Testing Rig

The testing rig, when properly lubricated to reduce mechanical impedance, performed well. The design choice to use a linear guide, instead of steel rods and ball bearings, paid off in the end - both a reduction in weight and complexity were achieved. The rig had a limited height for jump experimentation - due to the limitation in peak motor current which effectively limited the jump height to just under $0.4\ m$ the rig height was adequate.

The testing rig enabled a number of experiments to be performed successfully including both static and dynamic tests.

10.1.2 Embedded System & GUI Design

The embedded system design proved to be the majority of the work involved in the project. The choice of using a real time operating system to implement the communication and control enabled a complex, yes structured and easily adaptable system. The framework designed can definitely serve as a reference for future robotic projects where similar tasks are involved.

By choosing a functional GUI, qSerialTerm, as a starting point, extensive GUI development was made possible in the short duration available. The control and command plug-ins developed, based on the same packet packing method developed during the embedded system design, are intuitive and adaptable. As further experiments were performed, Qt GUI components could be appended and integrated quickly. The cross compilation of the GUI possible for both Windows and Linux means the platform can be used for future research without concern for compatibility.

The embedded system was theoretically capable of sample rates up to approximately $800\ Hz$ with the packet sizes involved, due to limitation of the motor drivers this was not possible. The sampling rate is especially critical for virtual model applications where

the response of the system needs to be consistently controlled to achieve a high fidelity response.

10.2 Results Validation

The process of modelling, simulation, implementation, and testing produced theoretical, simulated, and practical results. Using critical analysis and comparison we can validate these results.

10.2.1 Motor Current Predications

The active compliance motor current simulation in fig. 6.11 was used to validate the motor driver peak current specification.

The control simulation plots in fig. 8.9 showed slightly lower peak current predictions - this was expected as the control simulation was less comprehensive, whereas the active compliance motor current simulation covered all theoretical kinematic set-points.

In reality during experimentation, as seen in fig. 9.12, the current saturated during launch, but during the other phases of jumping the current requirements predicated were correct. This was expected as the current predictions did not take into account the impulsive force needed for launching.

Figure 9.19 shows the motor driver and current controller, both on-board the motor drivers and with the communication to the embedded system performed well. The minimum possible latency between current feedback and current command was achieved given the limitation of the motor driver communication system.

10.2.2 Motor Torque Predications

Motor torque predictions were performed in section 8.5. They showed a maximum torque of approximately $4 - 10\ Nm$ per motor depending on the virtual model topology used. When the arc-length rotational measure was used a maximum torque of $4\ Nm$ was predicted. In practise, during the jump tests, a maximum motor torque of $4.4\ Nm$ was seen. This is not considering the motor current and torque saturation needed for launching. This validates the torque predictions simulated before testing was performed.

10.2.3 Virtual Model Fidelity

The leg was modelled as a spring-damper system. In section 9.3 this virtual model was tested to determine, amongst other aims, the fidelity of the virtual model control system.

Various topologies were tested, and the results summarized in the analysis that followed. Although the tests showed that the natural frequency of the various responses were accurate when compared to theory, the damping factor was not easily measured for comparison - this was partially due to the use of a backwards difference velocity estimation method prior to changing to a Jacobian mapping for all further tests.

10.2.4 Force Control Fidelity

As performed in section 9.6, the experiment found that during repeated jumping, the force control system was both robust to disturbances and implemented reliable and repeatable force control.

The experiment performed in section 9.2 added to the validation of force control fidelity by showing the mechanical transparency between the motor torque and foot force coupling. In theory a linear force relationship was expected using a spring-damper model and the linear motor torque current relationship, with a varying radial offset - this was confirmed during experimentation.

Both of the above experiments show the fidelity of the force control. Although a Simulink force control simulation was performed, it was not directly comparable to the experiments, despite proving useful in determining the reliability of operation of the control system.

10.3 Performance Validation

Performance validation aims to both validate the performance of the system as originally expected during the research investigation stage, and as compared to other similar legged robots.

By considering the original problems to be investigated the following discussion can be had about the overall performance of the system:

10.3.1 Problems to be Investigated

- **High speed embedded system communication and packet processing:** Using a RTOS and DMA framework, high speed communication up to $800\ Hz$ was reliably achieved on the embedded system. Due to motor driver constraints this was limited to $200\ Hz$.
- **Virtual model control for accurate end effector force output:** Virtual model control force fidelity was shown by comparing simulation and testing to theory. It was found to be accurate for the majority of the testing, but without a proper load cell system it is difficult to properly measure performance.
- **Effective high speed kinematic control:** The kinematic control was most effectively tested during the trajectory tracking tests in section 9.8. Although accurate trajectory tracking was achieved, the high speed capabilities of the system were not extensively tested. During jump tests the maximum speed of $3\ m/s$ was tested and kinematic performance was adequate for that case.
- **Effective use of motor drivers to achieve rapid accelerations with a direct drive BLDC motor:** The motor drivers were limited in both current peak and communication speed specifications. They performed adequately for the jump tests, but during more rapid three dimensional tests with more robotic mass or jump height needed, better motor drivers would be required.
- **Development of a platform suitable for further use in dynamic legged motion research:** The software design is highly adaptable and capable of performing further research. The hardware design is adequate for the single vertical axis tests performed in this study, but would need to be more robust for further extensive testing.

10.3.2 State of the Art Comparison

Table 10.1 provides a performance comparison of various legged robots, as compiled by [9].

A few of the robots that specifically use the same linkage leg design as Baleka will be compared in more depth. Although the kinematic equations derived in [3] were adapted for Baleka, the study did not include the necessary detail needed to compare it's performance. The GOAT leg [9], the ATRIAS robot [23], and the Minitaur robot [2] will

be critically compared.

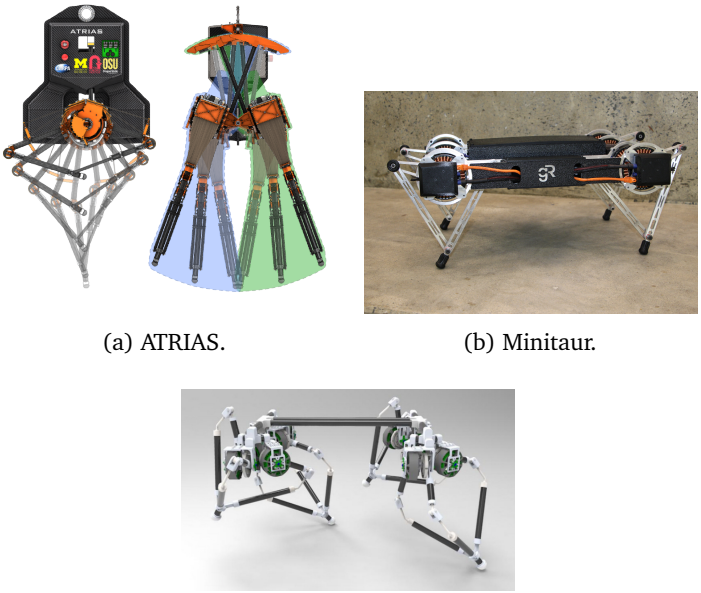
The robots mentioned are all of different size and configurations. The GOAT leg was tested in the single leg case and the ATRIAS robot was a fully sized $60\ kg$ beast. A fair comparison between these robots would be to use the energy to mass ratio in $[J/kg]$. This ensures the energy delivered is normalized, and that the large robots don't have an unfair advantage with larger actuators.

Baleka weighed in at an energy to mass ratio of $3.9\ J/kg$, ATRIAS at $1.1\ J/kg$, GOAT at $8.0\ J/kg$, and Minitaur at $4.7\ J/kg$. The places GOAT ahead of the pack with Minitaur and Baleka following just behind.

If on the other hand you consider the actuator mass ratio, it tells a different story. Baleka weighs in at $8.7\ J/kg$, ATRIAS at $10\ J/kg$, Minitaur at $11.75\ J/kg$, and GOAT at $16.7\ J/kg$. This means that the GOAT robot managed to get the most efficient use out of its motors, or the most energy delivered per kg of actuator. This performance measure also shows how Baleka failed to deliver as much energy due to the limitation of the motor drivers, as both the GOAT leg and Baleka leg used the same motors and virtual model control.

Considering the constraints on Baleka, it performed well for its class, delivering a middle of the range $3.9\ J/kg$. A number of the other robotic platforms did not meet these specifications, but it should also be considered that Baleka was specifically built for jumping whereas some of the robots could perform complex manoeuvres.

Considering that Baleka was built for jumping, we should compare jump heights. Baleka achieved a maximum height of $0.4\ m$, ATRIAS $0.11\ m$, Minitaur $0.48\ m$ and GOAT $0.82\ m$. These tests are a bit more difficult to compare given the different leg topologies used. Baleka was certainly capable of performing a much higher jump given more current, but considering that the GOAT leg jumped $0.82\ m$ with approximately the same current shows that the leg design and efficiency were superior to that of Baleka, and given more time and resources should be strived for.



(a) ATRIAS. (b) Minitaur.

(c) GOAT.

Figure 10.1: Robots for performance comparison.

Robot	Legs [no.]	DoF	Leg Length [m]	Mass [kg]	Motor Mass [%]	Gear Ratio	Jump Height [m]	Energy Delivered [J]
Baleka	1	2	0.3	2.2	45	n/a	0.4	8.63
Goat	1	3	0.26	2.5	48	n/a	0.82	20.11
MIT Cheetah	4	3	0.275	33	24	5.8	0.5	161.9
Minitaur	4	2	0.2	2	40	n/a	0.48	9.41
XRL	6	1	0.2	8	11	23	0.425	33.3
Delta Hopper	1	3	0.2	2	38	n/a	0.35	6.9
StarlETH	4	3	0.2	23	16	100	0.32	72.2
HRP3La-JSK	2	6	0.6	54	9.2	??	0.27	143
ATRIAS	2	3	0.42	60	11	50	0.11	64.7

Table 10.1: Legged robot performance comparison.[9]

11 Conclusions

"One always feels better when one has made up one's mind."
— C.S. Lewis, *The Last Battle*

As developed in chapter 1, the purpose of the study is stated below:

The purpose of this study is to develop a robust robotic leg platform and testing rig capable of rapid acceleration and high fidelity force control experimentation.

By critically considering the original research questions the purpose of the study can been validated:

Is a virtual model a suitable replacement for accurate dynamic modelling in complex robotic topologies?

In chapter 5 the virtual model was developed as a spring-damper compliance model coupled to the kinematic design developed in chapter 4.

During the virtual model dynamic spring-damper and drop tests, model parameters for the most part were accurate to the theoretical model development - the maximum deviation of the natural oscillation frequency, ω_0, was 14%.

During the trajectory tracking tests the maximum deviation from the trajectory set-point was $0.01\ m$ with mean Cartesian correlation values of $\rho_x = 0.77$ and $\rho_y = 0.60$.

These results conclude that a virtual model is a suitable replacement for an accurate dynamic model in controlling complex robotic topologies such as Baleka.

Can high fidelity force control be effectively implemented without using force feedback?

Force control was developed in chapter 8 using the forward kinematic Jacobian to map the end effector virtual model force to the necessary motor torques. A transparent coupling

between the direct drive motor and end effector was required in order to achieve high fidelity proprioceptive force control.

Although an accurate digital logging load cell was not available to test dynamic proprioceptive force control, it was tested in the static case. From the force control calibration and fidelity tests of chapter 9 the following conclusions were drawn:

1. A torque constant of $K_t = 0.08\ Nm/A$ was derived to properly calibrate the radial foot force output - previous research by [9] confirmed a torque constant of $0.072\ Nm/A$. This confirms the calibration methods for force control.
2. Using a spring constant of $K_s = 200\ N/m$ the estimated radial foot force followed a gradient of 200 in relation to radial offset. The force measured using a load cell followed a gradient of 185.54 - this shows the limited mechanical impedance between the motor and end effector coupling and validates the achievement of high fidelity proprioceptive force control.
3. Both the estimated and measured force profiles followed a linear model - this shows the limited effect of Jacobian force mapping distortion.

Is a virtual compliance control system effective in handling high speed impacts and executing rapid acceleration manoeuvres?

During the jump tests in chapter 9 the following dynamic performance was achieved:

1. During impact the robot end effector experienced a force of magnitude $55\ N$ - this force was dissipated within $15\ ms$ with accurate current command tracking. The virtual compliance model successfully handled high speed impacts.
2. The controller, using the virtual spring model decompression action, effected a highly impulsive force command delivering $8.63\ J$ of energy. The leg extended at a rate of $1.15\ m/s$ for the body to reach a maximum height of $0.4\ m$.
3. During the dynamic flight phase, the leg settled within $20\ ms$ after the recovery control action was triggered.

These performance measures validate the design, modelling, and control system's ability to handle dynamic manoeuvres such as impact absorption and launch control. The consecutive jump tests that were performed show that robust hopping control was

achieved with an 8.57% mean time shift and a negligible mean peak force deviation over 7 consecutive jumps.

In conclusion, a robust robotic platform was successfully developed that enabled high fidelity force control using a virtual compliance model. The research contributed a platform and control framework that can be effectively used in future rapid acceleration research in the UCT Mechatronics Lab.

12 Recommendations

Certain areas of research were out of the scope of this study, but would be interesting to investigate in future work. Various short-comings and limitations were also outlined in the report, these can be addressed in further study.

Virtual model control and its applications in robotics are endless and relatively new in the field of control. With the UCT Cheetah project under way and the Baleka platform successfully created, research can now be performed beyond what has been achieved in this study. The primary purpose of this study was the design and development of the platform, with less emphasis on the modelling and control further than virtual model control.

12.1 Modelling

The design of the leg was performed in Solidworks which enables extensive stress testing and finite element analysis of the leg under load. In the case of a similar leg being implemented on a complete robotic platform such as the UCT Cheetah, the leg will need to be properly modelled to ensure it can withstand these stresses and perform with increased joint torque.

The Lagrangian dynamics of the robot proved to be complex and worthy of a study in itself. [16] developed this dynamic model for a leg of similar topology, which could be studied and adapted for use with the Baleka platform.

12.2 Design

Mechanically the leg design needs a lot of work. The linkage systems, although performing adequately under the scope of the project, would need to be machined and redesigned for use in the 3 dimensional case. During dynamic jumping the leg experienced significant torque around the joints which resulted in mechanical impedance - the use of washers, nuts and bolts was not suited to this and under further extensive testing may prove problematic.

Due to time constraints the smaller testing rig linear guide rail and carriage were chosen due to availability. The linear guide experienced significant forward rotational torque - either the guide should be increased in size, or the robot should be better balanced in this axis by using co-linear motor mounts.

The motor drivers reached a current saturation point due to the $60\ A$ peak current limit. For added mass or more dynamic movement, this will need to be significantly increased to avoid current cut-out.

The virtual model would benefit from a control sampling frequency in the kilohertz time scale. Due to motor driver command response time limits and baud rate limits this was not possible. The RTOS framework developed for Baleka was capable of kilohertz time scale sampling frequencies and could be adapted for future robotic research.

12.3 Control

Although the virtual model performed well for energy control with simple jumping, MPC control could be investigated as an alternative.

Trajectory planning and optimization could be investigated and potentially make up a research project in itself, in the three dimensional case.

Further development of a Raibert type controller, using the kinematics and virtual model control already implemented, should be investigated for forward movement as seen in [13].

12.4 Experiments

Further experimentation should be performed by designing a rotational test rig. This would add another dimension to the control problem and allow further research in Raibert type hopping control.

A digital logging load cell or on-board force sensor should be used to better validate the virtual model force control implemented in this study. Manual logging, although adequate for calibration of static force output, does not guarantee dynamic force control fidelity.

References

[1] C. T. Farley and O. González, "Leg stiffness and stride frequency in human running," *Journal of Biomechanics*, vol. 29, no. 2, pp. 181–186, 1996.

[2] G. Kenneally, A. De, and D. E. Koditschek, "Design Principles for a Family of Direct-Drive Legged Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, jul 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7403902/

[3] J. M. Duperret and D. E. Koditschek, "An Empirical Investigation of Legged Transitional Maneuvers Leveraging Raibert's Scissor Algorithm." [Online]. Available: http://kodlab.seas.upenn.edu/Jeff/Robio15

[4] M. H. Raibert, J. Brown, H. Benjamin, M. Chepponis, J. Koechling, J. K. Hodgins, D. Dustman, W. K. Brennan, D. S. Barrett, C. M. Thompson, J. D. Hebert, W. Lee, and L. Borvansky, "Dynamically Stable Legged Locomotion," *MIT Technical Report*, no. 4148, p. 134, 1989. [Online]. Available: http://dspace.mit.edu/handle/1721.1/6820

[5] D. E. Galloway, Kevin C and Clark, Jonathan E and Koditschek, "Design of a tunable stiffness composite leg for dynamic locomotion," *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, no. August, pp. 215–222, 2009. [Online]. Available: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1650302

[6] A. Wang, S. Seok, A. Wang, D. Otten, and S. Kim, "Actuator Design for High Force Proprioceptive Control in Fast Legged Locomotion Actuator Design for High Force Proprioceptive Control in Fast Legged Locomotion," no. February 2016, pp. 1970–1975, 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6386252/

[7] "Bose Suspension System - Bose Automotive Systems Division." [Online]. Available: http://www.bose.com/prc.jsp?url=/automotive/bose{_}suspension/index.jsp

[8] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, dec 2008. [Online]. Available: http://content.iospress.com/doi/10.1080/11762320802557865

[9] S. Kalouche, "Design for 3D Agility and Virtual Compliance Using Proprioceptive Force Control in Dynamic Legged Robots," 2016.

[10] R. M. Alexander and A. Vernon, "The mechanics of hopping by kangaroos (Macropodidae)," *Journal of Zoology*, vol. 177, no. 2, pp. 265–303, aug 2009. [Online]. Available: http://doi.wiley.com/10.1111/j.1469-7998.1975.tb05983.x

[11] M. H. Raibert and E. R. Tello, "Legged Robots That Balance," *IEEE Expert*, vol. 1, no. 4, 1986.

[12] M. L. Laboratory. (1999, jan) Mit leg laboratory. [Online]. Available: http://www.ai.mit.edu/projects/leglab/robots/robots.html

[13] M. H. Raibert, "Hopping in Legged Systems: Modeling and Simulation for the Two-Dimensional One-Legged Case," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-14, no. 3, pp. 451–463, 1984. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6313238

[14] J. E. Pratt and G. A. G. Pratt, "Exploiting Natural Dynamics in the Control of a Planar Bipedal Walking Robot," *Technology*, vol. 223, no. September, pp. 253–262, 1998. [Online]. Available: ftp://theory.csail.mit.edu/pub/users/jpratt/natural{_}dynamics.pdf$\delimiter"026E30F$nhttp://dx.doi.org/10.1007/BFb0035216

[15] J. Pratt, "Virtual Model Control: An Intuitive Approach for Bipedal Locomotion," *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, feb 2001. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/02783640122067309

[16] H. Yu, "Modeling and control of hybrid machine systems – a five-bar mechanism case," *International Journal of Automation and Computing*, vol. 3, no. 3, pp. 235–243, 2006. [Online]. Available: http://www.springerlink.com/index/10.1007/s11633-006-0235-1

[17] G. Kenneally and D. Koditschek, "Kinematic Leg Design in an Electromechanical Robot," *Optimal Design Workshop, ICRA*, no. 1, pp. 1–4, 2014.

[18] H. E. Tseng and D. Hrovat, "State of the art survey: active and semi-active suspension control," *Vehicle System Dynamics*, vol. 53, no. 7, pp. 1034–1062, jul 2015. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/00423114.2015.1037313http://dx.doi.org/10.1080/00423114.2015.1037313

[19] G. Cook. (2016, Oct) Catalogue of parametrised CRC algorithms with 16 bits. [Online]. Available: http://reveng.sourceforge.net/crc-catalogue/16.htm

[20] J. Aparicio. (2013, Jan) qSerialTerm - Qt based serial port terminal emulator, data logger/plotter. [Online]. Available: https://github.com/JorgeAparicio/qSerialTerm

[21] G. A. Pratt and M. M. Williamson, "Series elastic actuators," *IEEE/RSJ International Conference on Intelligent Robots and Systems. 'Human Robot Interaction and Cooperative Robots'*, vol. 1, no. 1524, pp. 399–406, 1995. [Online]. Available: http://ieeexplore.ieee.org/document/525827/http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=525827

[22] A. Patel, "Understanding the motions of the cheetah tail using robotics," 2015.

[23] J. Grimes, "Atrias 1.0 & 2.0 - enabling agile biped locomotion with a template-driven approach to robot design." Masters Thesis, Oregon State University, 2013.

A Workspace Data

Research data is separated into the folders as listed below, available on CD or by request:

- background: various literature review documents.
- baleka-stm32f4-code: embedded system code.
- BalekaQtApp: Qt GUI code.
- baleka-system-kicad: start of electronics CAD representation of embedded system.
- communication: embedded system and motor driver communication protocol work.
- datasheets: various datasheets of components used in leg design.
- documentation: start of documentation write-up for Driveware use and configuration.
- drawio: draw.io graphic design application files used for report.
- experiments: experimental test data and video data.
- kinematics: various files with kinematic constraints and motor driver position limits.
- LATEX: report write-up in LATEX.
- miscellaneous: mass logs, motor model tuning, and performance tables.
- solidworks-original: compressed folder of all Ben Bingham's original CAD work.
- VacWorkLeg-original: compressed folder of all Luke Bell's original work.
- workspace-atollic: initial development of embedded system code.
- workspace-driveware: Driveware motor driver configuration files.
- workspace-inemo: work completed on iNemo code developed by Callen Fisher.
- workspace-matlab: Matlab scripts and saved workspaces.
- workspace-openscad: OpenSCAD programmatic CAD models.
- workspace-solidworks: CAD assemblies, designs and virtual models developed in Solidworks.

The final report write-up PDF can be found in the root directory, named report.pdf.

B Communication Protocol Code

Included pieces of code that may not be obvious to users or code that was particularly important to the operation of the protocol.

```
//Communication Timing
uint8_t Ts = 25; //Sampling time in 1/_X_ ms
uint8_t Td = 3;
//NB: #define configTICK_RATE_HZ ((TickType_t)_X_000) in FreeRTOSConfig.h
```

Listing 2: FreeRTOS timing configuration.

```
struct __attribute__((__packed__)) RXPacketStruct {
        uint8_t START[2];
        ...
        uint8_t StatBIT_1 : 1; //Bit field
        uint8_t StatBIT_2 : 1;
        uint8_t StatBIT_3 : 1;
        ...
        uint8_t CRCCheck[2]; //CRC-CCITT
        uint8_t STOP[2];
};
```

Listing 3: PC RX "packed" packet structure.

```
union {
        uint32_t WORD;
        uint16_t HALFWORD;
        uint8_t BYTE[4];
} WORDtoBYTE;
```

Listing 4: Byte conversion union.

```
HAL_UART_Receive_DMA(&PC_UART, RXBufPC, sizeof(RXPacket));
if(xSemaphoreTake( PCRXHandle, portMAX_DELAY ) == pdTRUE) {
        rcvdCount = sizeof(RXPacket);
        START_INDEX = findBytes(RXBufPC, rcvdCount,
        RXPacket.START, 2, 1);
        if(START_INDEX>=0) {
                memcpy(RXPacketPTR, &RXBufPC[START_INDEX],
                sizeof(RXPacket));
                RX_DATA_VALID = 0;

                WORDtoBYTE.BYTE[1] = RXPacket.CRCCheck[0];
                WORDtoBYTE.BYTE[0] = RXPacket.CRCCheck[1];
                CALC_CRC = crcCalc(&RXPacket.OPCODE, 0, PAYLOAD_RX, 0);

                //A useful tool when calculating and
                //confirming CRC values of various types:
                //https://www.lammertbies.nl/comm/info/crc-
                //calculation.html

                if(WORDtoBYTE.HALFWORD==CALC_CRC) {
                        RX_DATA_VALID = 1;
                        ... //Packet processing
```

Listing 5: PC RX packet processing.

```
struct __attribute__((__packed__)) BaseCommandStruct {
        uint8_t START[2];
        uint8_t CB;
        uint8_t INDOFF[2];
        uint8_t LEN;
        uint8_t CRC1[2];
        uint8_t DATA[4];
        uint8_t CRC2[2];
};

uint8_t SNIP;

struct BaseCommandStruct BaseCommand[50];
struct BaseCommandStruct* BaseCommandPTR;
```

Listing 6: Motor packet array of structs.

```
BaseCommandPTR = &BaseCommand[RXPacket.OPCODE];
BaseCommandCompile(RXPacket.OPCODE, 0b0011, 0x02, 0x45, 0x02,
RXPacket.M1C, 2, 0);
xQueueOverwrite(ICommandM1QHandle, &BaseCommandPTR);
```

Listing 7: Motor packet compilation current command example.

C Motor Driver Command Protocol

Command	Packet Index	Command Bits	Sequence Bits (Op-code)	Address Index	Address Offset	Function	Notes
KILL_BRIDGE	0	0x02	0001	0x01	0x00	Kill motor driver bridge.	
WRITE_ENABLE	1	0x02	0010	0x07	0x00	Enable write access.	
BRIDGE_ENABLE	2	0x02	0100	0x01	0x00	Enable motor driver bridge.	
CURRENT_COMMAND	20/21	0x02	0011	0x45	0x02	Command a current.	Conversion needed.
POSITION_COMMAND	22/23	0x02	1010	0x45	0x00	Command a position	Conversion needed.
READ_CURRENT	5	0x01	1100	0x10	0x03	Read current motor.	Conversion needed.
READ_POSITION	6	0x01	1111	0x12	0x00	Read motor position in counts.	Conversion needed.
READ_VELOCITY	7	0x01	0101	0x11	0x02	Read motor velocity in cts/s.	Conversion needed.
ZERO_POSITION	8	0x02		0x01	0x00	Re-callibrate encoder zero position.	
GAIN_SET	9	0x02		0x01	0x01	Select gain set 1 or 2.	Requires bit toggling.
GAIN_CHANGE_M1: P	10	0x02		0x38	0x00	Set driver position loop gain.	
GAIN_CHANGE_M1: I	11	0x02		0x38	0x02	Set driver position loop gain.	
GAIN_CHANGE_M1: D	12	0x02		0x38	0x04	Set driver position loop gain.	
GAIN_CHANGE_M2: P	13	0x02		0x38	0x00	Set driver position loop gain.	
GAIN_CHANGE_M2: I	14	0x02		0x38	0x02	Set driver position loop gain.	
GAIN_CHANGE_M2: D	15	0x02		0x38	0x04	Set driver position loop gain.	
CONFIG_SET	16	0x02	1001	0xD1	0x00	Select configuration 1 or 2.	Requires bit toggling.

Table C.1: Motor driver command protocol.

D Kinematic Simulation Code

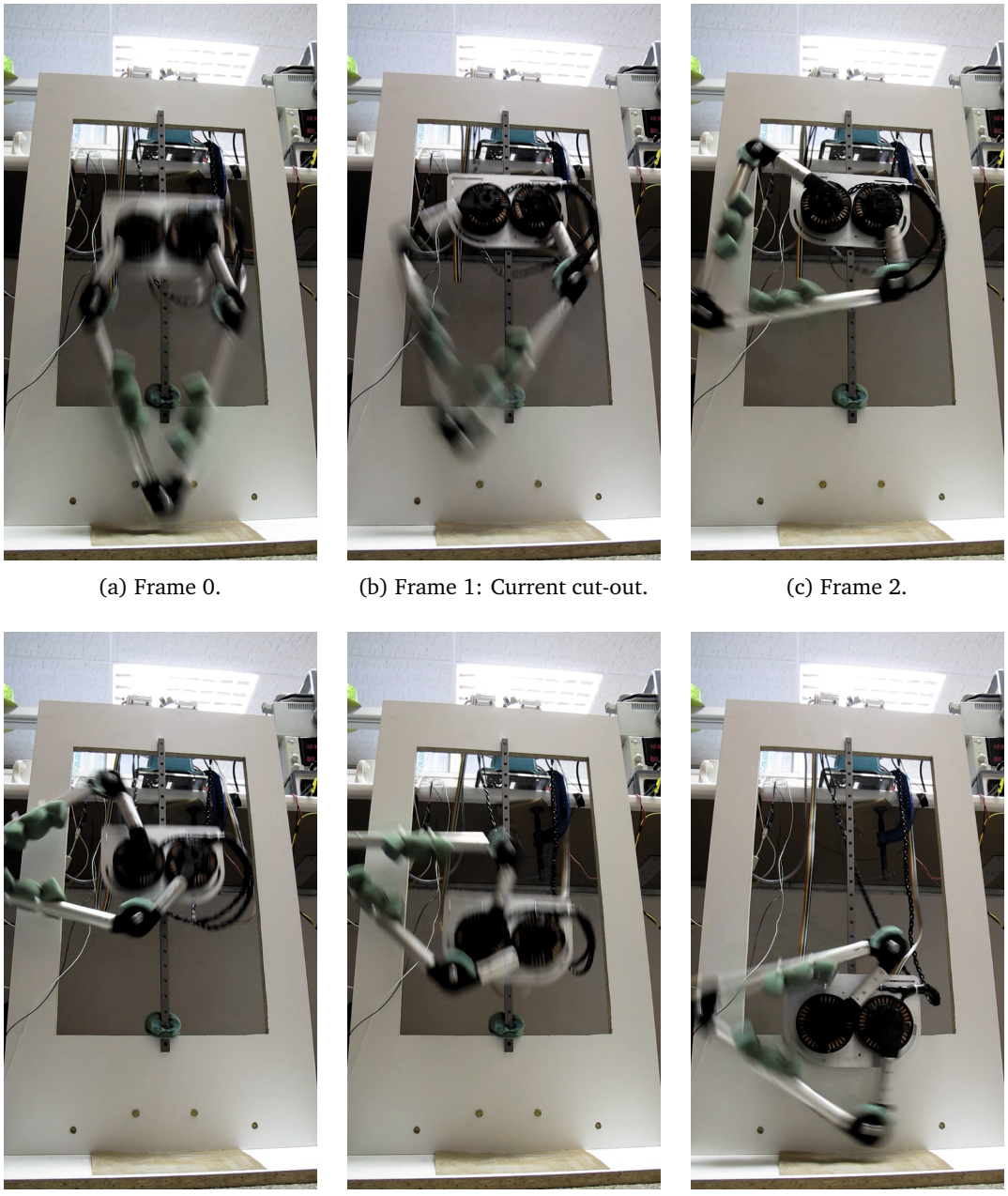
```
l1 = 0.15; %length of upper linkage in m (measured from center of joint of 5 cm diameter)
l2 = 0.3; %length of lower linkage in m (measured from center of joint of 5 cm diameter)

phi1 = (1/9*pi):0.125:pi; % all possible phi1 values
phi2 = (1/9*pi):0.125:pi; % all possible phi2 values

[PHI1, PHI2] = meshgrid(phi1, phi2); % generate a grid of phi1 and phi2 values

R = -l1*cos((PHI1 + PHI2)./2) + sqrt(l2 2 - l1^2*sin((PHI1 + PHI2)./2).^2)
THETA = (PHI1 - PHI2)./2;

plot(THETA(:), R(:), 'r.', 'MarkerSize', 20);
```

Listing 8: Kinematic simulation code to generate kinematic workspace.

E Jump Experiment

Frame (no.)	Time (ms)	Height (m)	Interval Velocity (m/s)	Phase
0	0	0.00000	0.000	Stance
1	84	0.05000	0.595	Decompression launch
2	126	0.12500	1.786	Decompression launch
3	167	0.21250	2.134	Flight
4	209	0.25000	0.893	Flight + Recovery
5	292	0.29375	0.527	Flight + Recovery
6	376	0.26250	-0.372	Freefall
7	501	0.12500	-1.100	Freefall
8	543	0.07500	-1.190	Impact
9	584	0.00000	-1.829	Compliant landing
10	709	0.02500	0.200	Compliant landing
11	793	0.04375	0.223	Compliant landing

Table E.1: Launch and compliant landing video frame data.



(a) Frame 0. (b) Frame 1: Current cut-out. (c) Frame 2.

(d) Frame 3. (e) Frame 4. (f) Frame 5.

Figure E.1: Motor driver over-current cut-out.

```
if(SHOT && r_fbk >= 0.38) {
        r_cmd = 0.3;
        k_r = 1;
        k_s = 0.1;
        ks_r = 633;
        kd_r = 15;
        ks_s = 400;
        kd_s = 5;
        SHOT = 0;
        TRIGGER = 1; //DANGEROUS!
}
if(PULLED && (ELAPSED > 1000)) {
        r_cmd = 0.4;
        k_r = 1;
        k_s = 0.1;
        ks_r = 1726;
        kd_r = 0;
        ks_s = 400;
        kd_s = 5;
        SHOT = 1;
        PULLED = 0;
        ELAPSED = 0;
}
if(TRIGGER) {
        r_cmd = 0.25;
        k_r = 1;
        k_s = 0.1;
        ks_r = 633;
        kd_r = 15;
        ks_s = 400;
        kd_s = 5;
        ELAPSED = 0;
        TRIGGER = 0;
        PULLED = 1;
}
else{
        ELAPSED++;
}
```

Listing 9: Jump control condition loop.