THE UNIVERSITY OF SYDNEY

HONOURS THESIS

Balancing Control Mechanisms for the Cassie Robot on One Foot

Author:

William LAVENDER

Advisor:

Prof. Ian MANCHESTER

A thesis submitted in fulfilment of the requirements

for the degree of Bachelor of Engineering (Mechanical)

in the

SCHOOL OF AEROSPACE, MECHANICAL AND MECHATRONIC ENGINEERING

October 31st, 2024

Abstract

This thesis investigates advanced control strategies to achieve single-leg balance on the Cassie biped robot, addressing a critical challenge unexplored in prior research. As robots increasingly perform complex tasks such as walking, navigating obstacles, and maintaining stability on uneven terrain, mastering single-leg balance is essential for autonomous deployment in dynamic environments. This study proposes and evaluates a Linear Quadratic Regulator (LQR) controller using the Multi Joint Dynamics with Contact (MuJoCo) physics simulator, highlighting its effectiveness relative to more sophisticated control techniques. The controller is fine-tuned to optimise performance within a specific operating region, and its capabilities are rigorously tested through simulation. These findings offer valuable insights into the feasibility of real-world implementation, contributing to the broader field of robotic balance and locomotion control.

Statement of Contribution

I hereby declare that the content of this thesis is the product of my own work and that all assistance received in the preparation of this thesis and relevant sources have been acknowledged.

The work I contributed to includes:

- I designed the problem statement investigated in this thesis with the guidance of my supervisor.
- I carried out the literature review, examining existing control methods and robots related to my work.
- I developed the methodology and implementation outline.
- I developed my own Python code unique to the Cassie robot and generated all results myself.
- I developed each script in Matlab to import results from Python and visualise the results of each simulation.

The work not performed by me but utilised in this thesis includes:

- My Python LQR code script was initially structured similarly to an LQR control implementation for a 27 Degree of Freedom (DoF) humanoid robot to gain foundational understanding, and I have referenced this appropriately.

Students Full Name & Signature:

Acknowledgements

I would like to begin by expressing my deepest gratitude to my supervisor, Prof. Ian Manchester, for his exceptional guidance, expertise, and encouragement throughout the development of this thesis. Despite his busy schedule, he has always made time to provide invaluable feedback and unwavering support, which have been instrumental in helping me navigate and overcome the challenges of this project. I am equally grateful to PhD candidate Nicholas Barbara, whose mentorship and technical expertise have significantly shaped the direction and execution of my research. His patience, generosity with his time, and willingness to share his knowledge have greatly enriched my learning experience and contributed to the success of this work.

FIGURE 1: On-Campus Experimentation with the The University of Sydney (USYD) Cassie Robot

I extend my heartfelt thanks to my friends and family for their unwavering support and encouragement, not only throughout this thesis but across my entire academic journey. Their belief in my abilities has been a constant source of motivation, and I am deeply grateful for their patience and understanding, particularly during

moments of stress and doubt. This milestone has been a long time coming, and I could not have reached it without such encouragement. Finally, I would like to acknowledge the broader team at USYD, whose support made it possible for me to work with the Cassie model which provided an incredible and enriching opportunity to explore.

This thesis has been both a challenging and rewarding journey, and I hope the insights presented here will serve as a valuable foundation for future research. I look forward to the day Cassie achieves single-leg balance in real-life applications, building upon the groundwork established in this study. It is my hope that this work inspires further advancements in bipedal robotics and contributes meaningfully to the field of autonomous systems.

Contents

List of Figures

List of Tables

List of Acronyms

3D 3 Dimensional

AI Artifical Intelligence

CFDM Centred Finite Difference Method

CoM Centre of Mass

CoP Centre of Pressure

DARPA Defense Advanced Research Projects Agency

DoF Degree of Freedom

EKF Extended Kalman Filter

IDP Inverted Double Pendulum

LTI Linear Time Invariant

LQR Linear Quadratic Regulator

MJM Momentum Jacobian Matrix

MPC Model Predictive Control

MIMO Multiple-Input Multiple-Output

MuJoCo Multi Joint Dynamics with Contact

RL Reinforcement Learning

SEA Series Elastic Actuator

USYD The University of Sydney

ZMP Zero Moment Point

Chapter 1

Introduction

1.1 Background and Motivation

Since the creation of the first bipedal robot in 1937, achieving stable balance has been a central challenge in advancing humanoid robotics [1]. Bipedal robots, which stand on two legs and emulate the mechanics of human or animal gait, require robust balance to navigate their environments effectively and perform tasks that demand stability, such as walking, running, or kicking a ball [2]. Without reliable balance, these robots would be restricted to controlled settings and unable to interact meaningfully with the world around them [1].

Over the past decade, the capabilities of bipedal robots have grown significantly, with robust locomotion now commonplace in both academic research and industrial applications. These robots are increasingly deployed across fields such as manufacturing, healthcare, agriculture, and space exploration [3]. This progress has been driven by advancements in hardware design and control methodologies, with optimisation techniques playing a pivotal role in improving performance [3, 4].

However, current research lacks depth in achieving effective one-legged balance for bipedal robots, as control methods primarily focus on multi-limb stability for tasks like walking and jumping [1]. This emphasis has limited advancements in single-leg

tasks, disregarding the potential to enhance agility and adaptability of gait locomotion.

Addressing this gap is critical, as robots still face significant challenges in tasks that humans accomplish effortlessly, such as grasping irregular objects, navigating dynamic environments, and adjusting to unpredictable changes in terrain or obstacles [5]. Reliable single-leg balance could enable autonomous navigation in unpredictable terrains, enhance agility for collision avoidance, or even support autonomous applications across diverse fields [1, 2]. A reliable controller would enable biped robots to perform with greater agility and responsiveness, enabling adaptive reactions to shifts in weight or terrain. This enhancement would not only increase their operational capability but also broaden their practical application, allowing for meaningful advancements in autonomy and adaptability that are crucial for real-world, unstructured environments.

Balancing on a single foot presents unique challenges compared to balancing on both feet. These include reduced stability due to a smaller support polygon and increased control complexities from the kinematic coupling between the CoM and the suspended leg [1]. The robot's physical interaction with its environment further complicates this inherent instability. Unlike humans, the Cassie robot's simpler design—with narrow feet and no upper body—limits its ability to shift its CoM for balance adjustments [6]. In the past, effective dynamic mobility has required both appendages, as relying on a single leg significantly restricts movement capabilities [7]. Managing balance in bipedal robots remains a complex issue, owing to intricate multi-body dynamics and various sources of uncertainty in computational and experimental settings [8].

As such, the following research aims to address these challenges by developing an effective control technique tailored to balance Cassie on one leg. Integrating this advanced control methodology can enhance the robot's stability and adaptability, leading to improved performance in unpredictable environments. This will facilitate more efficient, flexible deployment in autonomous applications, even on rough terrain.

1.2 Research Objectives

The primary focus of this thesis is to select and evaluate an effective control method for balancing the Cassie robot on one leg. Existing control methods will be assessed for their applicability and potential need for adaptation, and the most promising methodology will then be implemented and tested in a simulated 3 Dimensional (3D) model within the MuJoCo physics engine. Throughout this process, design decisions and optimisation approaches will be closely monitored to assess the effectiveness and robustness of the proposed controller. This comprehensive evaluation will ultimately determine the method's suitability for real-world implementation. The derivation, implementation details, and simulation results will be presented in this thesis to help evaluate the controller's feasibility. More formally, this report will address the following research objectives:

1. Compare existing control methodologies based on academic research and carefully choose a suitable method tailored to the Cassie biped robot.
2. Determine if this method can be successfully implemented using a 3D physics simulator to enable the Cassie robot to balance on one leg. If so, analyse the effectiveness of this method in simulation.
3. Finally, assess whether the proposed method can be effectively implemented on the physical model in the future by adjusting design parameters such as mass, friction, and disturbances in simulation.

1.3 Thesis Structure

This thesis is divided into six chapters, each addressing a distinct aspect of the research problem.

Chapter 2 provides a historical overview of bipedal robots, exploring their applications, advancements, and challenges inherent in their development. It also examines the control methods commonly used to achieve robotic balance.

Chapter 3 describes the methodology for implementing the chosen LQR controller on Cassie to achieve single-leg balance. The process for identifying the robot's linearisation position is explained, followed by a formulation of the $\mathbf{Q}$ and $\mathbf{R}$ matrices to optimise the LQR controller for balancing.

Chapter 4 presents the simulation results obtained using the methodology described in the previous chapter, following the implementation of the LQR controller in MuJoCo. The results demonstrate successful stabilisation of the model, with movements showcasing the controller's effectiveness. Additionally, graphs generated in MATLAB illustrate the performance of each state, providing a comprehensive evaluation of the controller's ability to achieve single-leg balance.

Chapter 5 discusses the simulation results, comparing them against existing literature. This Chapter evaluates the controller's effectiveness in maintaining Cassie's balance, analysing its stability, response time, and robustness under various simulated conditions. This chapter also addresses the limitations observed in the LQR-controlled balance simulations and explores possible improvements.

Finally, Chapter 6 concludes the thesis with a summary of the research findings, emphasising the effectiveness of the LQR controller in successfully balancing Cassie in simulation. This chapter also suggests directions for future work, including real-world testing, application of more sophisticated control techniques, and any limitations of the proposed control methodology.

Chapter 2

Literature Review

2.1 A Brief History of Biped Robots

Biped robots have undergone significant evolution throughout history, transforming from simple, limited-functionality machines into complex, highly capable humanoids. Early examples, such as Elektro (1937), were characterised by their restricted mobility and substantial weight, standing 1.5m tall and weighing 110kg [9]. These early designs could only produce jerky, uncoordinated movements that required careful synchronisation of hip and knee actions [9]. Over the following decades, breakthroughs in Artifical Intelligence (AI), materials science, and motor control enabled the development of more sophisticated robots. Wabot-1, created by Waseda University in 1973, marked a significant advancement, demonstrating the ability to walk, communicate, and use artificial receptors to interact with its environment by measuring distances and directions [7, 10]. These achievements laid the groundwork for modern humanoid robots, including Robonaut 2 (2010), designed for space missions, and ASIMO (2011), known for its exceptional bipedal walking abilities (Figure 2.1). Today's humanoid robots, such as Ameca (2021), showcase astonishing capabilities, including lifelike facial expressions that make them more realistic and interactive than ever before [11].

(A) Elektro (1937) [9]

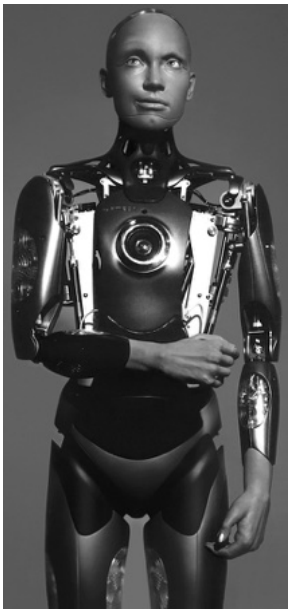
(B) Robonaut 2 (2010) [12]

(C) ASIMO (2011) [13]

(D) iCub (2013) [14]

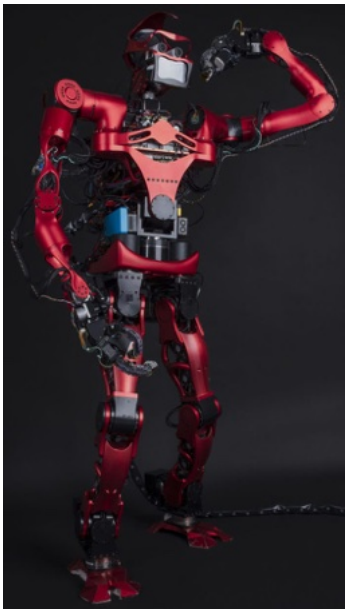
(E) DRC-Hubo+ (2015) [15]

(F) Atlas (2016) [16]



(G) Ameca (2021) [11]

(H) CyberOne (2022) [17]



(I) Tocabi (2023) [18]

FIGURE 2.1: Historical Humanoid Robots

TABLE 2.1: Recent Humanoid Robot Evolution [7, 10]

Year	**Robot Name & Company**	**Humanoid Robot Development**
2013	iCub 2.5 (Italian Institute of Technology)	Enhanced vision, touch, and motor control; open-source platform.
2013	Valkyrie (NASA)	Space exploration robot with advanced mobility and sensor systems.
2014	Manav (A-SET Training and Research Institutes)	India's first 3D-printed humanoid, used for education and research.
2015	HRP-2Kai (AIST)	Features improved flexibility and capabilities for disaster response.
2015	DRC-Hubo+ (KAIST)	Transforms from bipedal to wheeled for versatile mobility.
2016	Atlas (Boston Dynamics)	Agile robot performing backflips and complex parkour moves.
2017	Cassie (Agility Robotics)	Bipedal robot known for speed and balance; used in locomotion research.
2021	Tesla Bot (Optimus)	AI-equipped robot for repetitive tasks, designed with advanced mobility.
2021	Ameca (Engineered Arts)	Humanoid with realistic facial expressions for human-robot interaction research.
2022	CyberOne (Xiaomi)	Emotion detection and communication features for interactive tasks.
2023	Tocabi (Dynamic Robotic System Lab, Seoul National University)	33 DoF humanoid designed for complex, teleoperated tasks.
2023	Guardian XT (Sarcos)	Teleoperated robot with dual arms for hazardous operations.

2.2 Cassie Robot: Design, Capabilities, and Research Applications

The Cassie robot is a bipedal robot developed at Oregon State University in the USA, funded by a US$1 million grant from the Defense Advanced Research Projects Agency (DARPA) [19]. As shown in Figure 2.2, Cassie stands 1.15 metres tall, with a width of 0.55 metres and a length of 0.40 metres. Its design draws inspiration from the biomechanics of an ostrich, featuring backward-bending knees and a streamlined lower body structure [20].

In contrast to the models shown in Figure 2.1, Cassie has demonstrated exceptional bipedal mobility, showcasing impressive capabilities in balancing, navigating uneven terrain, and completing a 5 kilometre run in just over 53 minutes on a single charge [21]. These advancements have paved the way for new possibilities in legged gait design, with Cassie earning the Guinness World Record for the fastest 100 metres by a bipedal robot [21]. Further highlighting its agility, Cassie has also achieved a 1.4 metre long jump, all without any additional training, exemplifying the robot's real-world applicability [6].

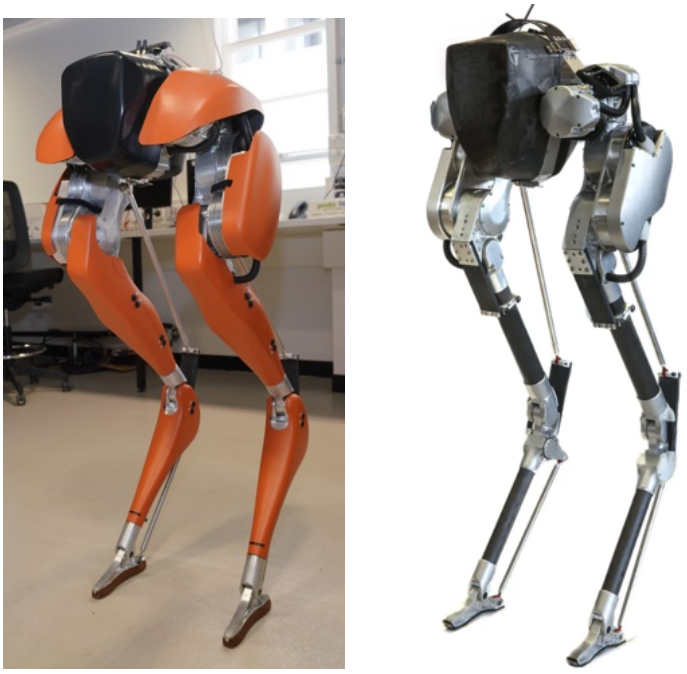
TABLE 2.2: Cassie Robot Dimensions [20]

Parameter	Unit	Value
R_w	m	0.55
R_l	m	0.40
R_h	m	1.15
Mass	kg	33.31
V_{max}	$\frac{km}{h}$	14.60

Many research laboratories across the world have worked and continue to work on the Cassie robot, including the Dynamic Robotics Laboratory at Oregon State University, the Dynamic Legged Locomotion Lab at the University of Michigan, the AMBER Lab at Caltech, and the Agile Robotics Laboratory at Harvard [22]. However, these labs have focussed primarily on dynamic applications such as running or walking upstairs and have neglected more simple applications such as balancing on one leg.

2.3 Balancing Methods

In order to achieve the research objectives stated above, the concept of 'balancing' must be precisely defined. Theoretically, balancing involves maintaining a stable upright posture, which occurs when an entity sustains its stance on a point, line, or very narrow area in at least one direction. In this framework, movements are continuously adjusted to keep the Centre of Pressure (CoP) – the point where all contact forces concentrate – within a stable region to prevent tipping over [7, 23].



(A) USYD Model With Cover

(B) Oregon State Model Without Cover [20]

FIGURE 2.2: Cassie Robot: Comparison of Models With and Without Cover

A robotic controller works by keeping the robot's state variables, such as its position, velocity, and orientation, within a defined stability zone; if these variables deviate beyond this zone, the robot becomes unbalanced and is likely to fall. For example, the robot's stability will be compromised if the pelvis becomes too close to the ground or if the knee angle overextends. This is particularly critical for humanoid robots, which have a high centre of mass and a small base of support (such as their feet), making them especially susceptible to imbalance from minor disturbances or uneven terrain [7].

There are two main types of balance - static and dynamic - both of which are important to understand when designing effective balance control systems for robots. Static equilibrium refers to perfect balance achieved without any movement or external forces, whereas dynamic balance involves continuous adjustments to maintain stability in the presence of disturbances [24]. In this report, Cassie will undergo dynamic balancing due to the need to maintain stability during motion and external disturbances, reflecting real-world conditions [24].

2.3.1 Centre of Mass and Support Polygon

In a Multiple-Input Multiple-Output (MIMO) system, maintaining control of the CoM is crucial, as excessive deviation can result in system failure. The CoM represents the average location of all mass within the system, proportional to each individual component, simplifying the analysis of complex motions by behaving as the single point where the system's entire mass distribution can be considered concentrated [25]. For simple, uniformly dense objects, the CoM aligns with the geometric centre (such as the centre of a solid disc). However, for non-uniform or hollow shapes like a ring, the CoM may lie outside the material, occupying an empty central point, as is the case for Cassie (see Figure 3.4) [25].

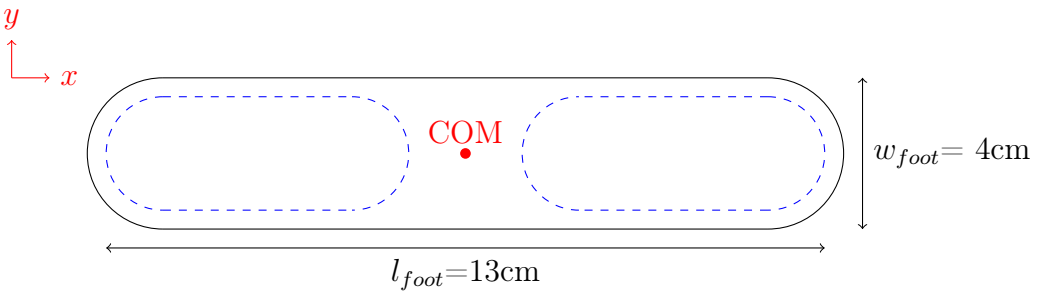
The CoM for a system of particles, where m_i represents the mass of the i-th particle and x_i, y_i, and z_i represent the coordinates, can be computed using the following equations, with M denoting the total mass of the system [25].

$$X_{\text{com}} = \frac{\sum_{i=0}^{n} m_i x_i}{M} \tag{2.1}$$

$$Y_{\text{com}} = \frac{\sum_{i=0}^{n} m_i y_i}{M} \tag{2.2}$$

$$Z_{\text{com}} = \frac{\sum_{i=0}^{n} m_i z_i}{M} \tag{2.3}$$

FIGURE 2.3: Single Leg Support Polygon



The Cassie robot's support polygon, shown in Figure 2.3, defines the area within which the robot's CoM must stay to maintain balance and prevent tipping [7]. A support polygon is formed by connecting the robot's contact points with the ground.

For stable balancing on one foot, the vertical projection of the CoM onto the ground must stay within this polygon, ensuring a weight distribution that keeps the robot upright. Maintaining the CoM within these bounds is critical for stability, as any deviation outside the support polygon can lead to loss of balance [26]. As portrayed in Figure 2.3, Cassie's support polygon is narrow, with a width of just 4cm and a length of 13cm, increasing the difficulty of the task at hand. In comparison, the Unitree G1 robot MuJoCo model has a length of 19cm and a width of 5cm [27].

2.3.2 Zero Moment Point

The Zero Moment Point (ZMP) is a concept in the dynamic stability of bipedal walking robots, serving as a stability criterion vital for ensuring balance during locomotion [7]. When the robot's foot is in contact with the ground, as shown in Figure 2.4, the combined reaction force and moment exerted on the sole can be represented by a single resultant force **R**. The point on the sole where the total reaction force intersects the ground surface is known as the ZMP, where the resultant moment is zero [7, 24]. For Cassie, maintaining the ZMP within the foot's contact area is crucial for dynamic stability during single-leg support.

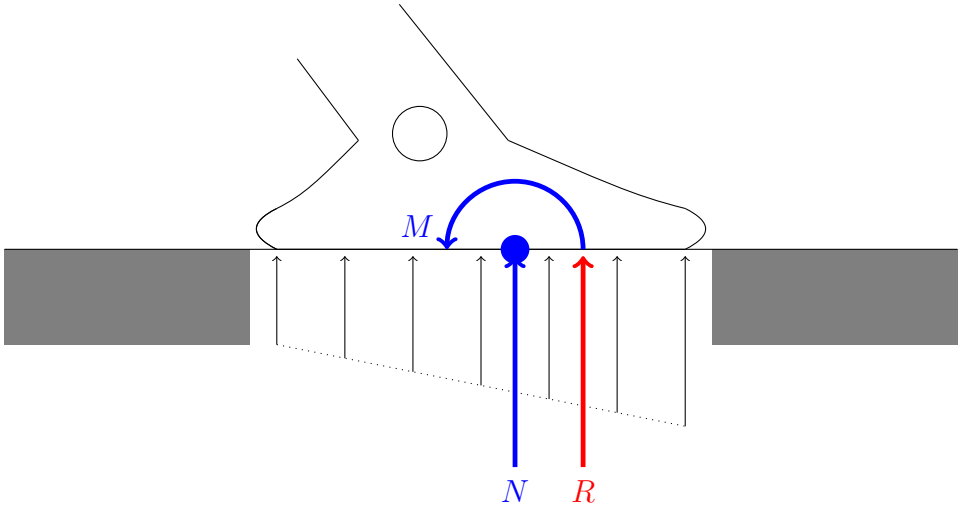


FIGURE 2.4: ZMP on Bipedal Foot

2.4 Control Methods

Balancing Cassie necessitates a sophisticated control strategy. This section explores several promising methods including LQR control, Model Predictive Control (MPC), H^∞ control, and Reinforcement Learning (RL). The following analysis will explore the strengths and weaknesses of each method, highlighting their unique perspectives and distinct advantages for balancing Cassie. This analysis aims to identify the optimal control strategy for Cassie by evaluating each method's primary point of difference.

2.4.1 LQR Control Theory

A LQR is a widely used approach in optimal control theory, designed to determine the best feedback strategy for a linear system by minimising a cost function that penalises both state deviations and control effort [28]. Extensively applied to stabilise non-linear systems locally, the LQR provides an exact solution, making it a powerful tool for control design.

LQR control theory applies to dynamic systems with continuous state and action spaces, where the control inputs $\mathbf{u}$ and states $\mathbf{x}$ are vectors of real numbers. Proposed in 1964 by Rudolph Kalman, the LQR is one of the most well-known and widely applied forms of optimal control, known for its relative simplicity and applicability to multivariable and time-varying linear systems [28]. The name derives from combining a linear system with a quadratic cost function, implying a Regulator objective: the cost function dictates that the system state should remain small (near zero) to minimise deviations [28].

Consider a system described by a Linear Time Invariant (LTI) model,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.4}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{2.5}$$

where,

- $\mathbf{x} \in \mathbb{R}^n$ is the state vector,
- $\mathbf{u} \in \mathbb{R}^m$ is the control input,
- $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the state matrix,
- $\mathbf{B} \in \mathbb{R}^{n \times m}$ is the input matrix.

The cost function to be minimised is,

$$J(\mathbf{x}(0), \mathbf{u}) = \int_0^T g(\mathbf{x}(t), \mathbf{u}(t))\, dt + g_T(\mathbf{x}(T)) + \mathbf{x}(T)^\top \mathbf{P}_T \mathbf{x}(T) \tag{2.6}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are the weighting matrices for the state and control input, respectively.

The optimal control law is given by Equations 2.7, 2.8 and 2.9 [28]:

$$-\dot{\mathbf{P}} = \mathbf{A}^\top \mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\top \mathbf{P} + \mathbf{Q}, \tag{2.7}$$

$$\mathbf{P}(T) = \mathbf{P}_T, \tag{2.8}$$

$$\mathbf{u}(t) = -\mathbf{R}^{-1}\mathbf{B}^\top \mathbf{P}\mathbf{x}(t) \tag{2.9}$$

In which the closed loop system is:

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} \tag{2.10}$$

2.4.1.1 LQR Evaluation

The most significant strength of the LQR is robustness: the property of a closed-loop system to remain stable despite variations in the open-loop system [29]. Under reasonable system and cost function specifications, LQR guarantees stability, ensuring the system can withstand disturbances and modelling inaccuracies [29, 30]. This is particularly beneficial for balancing a biped robot on one leg, as it helps the robot maintain balance despite unexpected shifts in weight or external perturbations.

LQR controllers are also computationally efficient and can be implemented in real-time applications. They provide optimal control laws that are linear functions of the state variables, simplifying the design and implementation process [29]. Moreover, LQR can handle multivariable systems, making it suitable for complex systems like the Cassie robot.

Despite its strengths, LQR control has some limitations, one of which is its reliance on an accurate system dynamics model. Discrepancies between the model and the actual system can lead to suboptimal performance or even instability, where slight differences between the theoretical model and the robot's actual behaviour could cause it to fall. Moreover, LQR control is inherently designed for linear systems. While it can stabilise non-linear systems locally, it may struggle with highly non-linear dynamics or significant deviations from the linearised operating point without modifications or additional control strategies [29, 30]. Additionally, LQR control does not account for physical system constraints and handles uncertainties and disturbances poorly [29]. For a biped robot, this could result in limbs rotating beyond safe angles or input torques exceeding motor capabilities.

2.4.2 Model Predictive Control Theory

MPC is an advanced control method that predicts and optimises a system's future behaviour by solving a constrained optimisation problem [31]. Using a system model, MPC predicts future states and determines optimal control inputs over a finite time horizon, all while respecting constraints on both [4]. This approach effectively handles multivariable systems with constraints, making it highly suitable for complex applications such as robotics.

For a basic overview, consider a system described by a LTI model as described by Equations 2.7, 2.8 and 2.9. Now the cost function to be minimised over a finite horizon remains the same as Equation 2.6 above. However, the optimisation problem

is solved at each time step to determine the optimal control inputs:

$$\min_{\mathbf{u}_0,\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}} J \quad \text{subject to} \quad \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \quad \mathbf{u}_k \in \mathcal{U}, \quad \mathbf{x}_k \in \mathcal{X} \tag{2.11}$$

where $\mathcal{U}$ and $\mathcal{X}$ represent the constraints on inputs and states, respectively [31].

2.4.2.1 MPC Evaluation

One significant advantage of MPC is its ability to handle constraints explicitly, which is particularly important in robotic applications where physical limits and safety requirements must be respected. By optimising control actions, MPC ensures that the robot will operate within an allowable range of motion, avoiding collisions and maintaining stability [30, 31].

Another strength of MPC is its predictive nature, which allows it to anticipate future events and adjust control actions accordingly. This predictive capability is beneficial for tasks such as balancing the Cassie robot on one leg, where the controller needs to account for the dynamics and possible disturbances that might affect the robot's stability. By continuously updating its predictions and optimising control inputs, MPC can maintain balance even in the presence of external perturbations [31].

Additionally, MPC's flexibility in handling multivariable control problems makes it suitable for complex systems. For instance, the MPC was employed on the MIT Cheetah 2 robot to clear a 40cm high obstacle by solving a constrained nonlinear optimisation, showcasing its power in dynamic and challenging scenarios [4].

Despite its strengths, MPC has some drawbacks. One of the main challenges is the computational complexity associated with solving the optimisation problem. This can be demanding, especially for systems with fast dynamics or when the control horizon is long [30]. For Cassie, the real-time requirements might necessitate significant computational resources or specialised hardware to achieve the desired performance.

Another issue with MPC is its reliance on an accurate system model, where inaccuracies can lead to suboptimal control actions or even instability. Humanoid robots often suffer from issues like backlashes and joint friction, walking instability, and limited payloads during motion execution [7]. These factors can introduce modelling errors, making it challenging for MPC to maintain accurate control [7].

2.4.3 H^∞ Control

H^∞ control is a robust control method designed to handle systems with uncertain parameters and disturbances. The goal of H^∞ control is to minimise the worst-case effect of disturbances on the system output, ensuring stability and performance even in the presence of model uncertainties and external perturbations [32].

Consider a system described by the following equations:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{H}\mathbf{w}(t) \tag{2.12}$$

$$\mathbf{z}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \tag{2.13}$$

where,

- $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector,
- $\mathbf{u}(t) \in \mathbb{R}^m$ is the control input,
- $\mathbf{w}(t) \in \mathbb{R}^p$ is the disturbance input,
- $\mathbf{z}(t) \in \mathbb{R}^q$ is the controlled output,
- $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the state matrix,
- $\mathbf{B} \in \mathbb{R}^{n \times m}$ is the input matrix,
- $\mathbf{H} \in \mathbb{R}^{n \times p}$ is the disturbance matrix,
- $\mathbf{C} \in \mathbb{R}^{q \times n}$ is the output matrix,
- $\mathbf{D} \in \mathbb{R}^{q \times m}$ is the direct transmission matrix.

The objective of H^∞ control is to design a controller that minimises the L^2 gain from the disturbance $\mathbf{w}(t)$ to the controlled output $\mathbf{z}(t)$. This can be formulated as an optimisation problem where the controller aims to achieve:

$$\|\mathbf{z}\|_2^2 \leq \gamma^2 \|\mathbf{w}\|_2^2 \tag{2.14}$$

for a given performance level γ.

The state feedback H^∞ control law can be derived by solving the following algebraic Riccati equation:

$$\mathbf{A}^\top \mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\top \mathbf{P} + \mathbf{Q} + \frac{1}{\gamma^2}\mathbf{P}\mathbf{H}\mathbf{H}^\top \mathbf{P} = 0 \tag{2.15}$$

where $\mathbf{P}$ is the positive definite solution, and the control input is given by the following [30, 33].

$$\mathbf{u}(t) = -\mathbf{R}^{-1}\mathbf{B}^\top \mathbf{P}\mathbf{x}(t) \tag{2.16}$$

2.4.3.1 H^∞ Evaluation

One of the key strengths of H^∞ control is its robustness to model uncertainties and external disturbances. By minimising the worst-case gain from disturbances to the controlled output, H^∞ control ensures the system can maintain stability and performance despite significant dynamic variations [33]. This is particularly beneficial for maintaining the stability of a biped robot on a single support leg, as it helps the robot remain steady despite unpredictable disturbances and inaccuracies in the model.

H^∞ control can also explicitly handle multivariable systems and constraints. This flexibility makes it suitable for complex robotic systems, where multiple control inputs and outputs must be managed simultaneously. Additionally, H^∞ control can be used to design controllers that achieve specific performance criteria, such as minimising the sensitivity of the system to certain types of disturbances (for

example, in the form of a wind gust or another autonomous vehicle crashing into Cassie) [33].

Regardless of its strengths, H^∞ control has limitations. One of the main challenges is the complexity of the controller design process, which often involves solving complex optimisation problems. This can be computationally demanding, especially for systems with fast dynamics or high-dimensional state spaces [30]. For scenarios where a biped robot relies on a single leg for support, the real-time requirements might necessitate significant computational resources or specialised hardware to achieve the desired performance.

Similar to LQR control, H^∞ control relies on accurate modelling of system uncertainties and disturbances. Although H^∞ control offers robustness to a certain extent of uncertainty, significant modelling errors or unmodelled dynamics can still compromise performance [33].

2.4.4 Reinforcement Learning (RL) Control Theory

RL is an area of machine learning where an agent learns to make decisions by performing certain actions and observing the outcomes or rewards of those actions. Unlike traditional control methods described above, RL does not require a detailed mathematical model of the system but instead optimises the control policy directly through environmental interaction, making it highly suitable for complex and dynamic systems such as robotics [4].

Consider a system described by the following equations in the context of RL:

$$\mathbf{x}_{t+1} \sim P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \tag{2.17}$$

$$r_t = r(\mathbf{x}_t, \mathbf{u}_t) \tag{2.18}$$

where,

- $\mathbf{x}_t$ is the state at time t,

- $\mathbf{u}_t$ is the action taken at time t,
- $P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ is the state transition probability,

The goal of RL is to find a policy $\pi(\mathbf{u}_t|\mathbf{x}t)$ that maximises the expected cumulative reward:

$$J(\pi) = \mathbb{E}\left[\sum t = 0^{\infty}\gamma^t r_t\right] \tag{2.19}$$

where $\gamma \in [0, 1)$ is the discount factor [34].

One popular RL algorithm is Q-learning, where the Q-function is defined as:

$$Q(\mathbf{x}, \mathbf{u}) = \mathbb{E}\left[r_t + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}_{t+1}, \mathbf{u}')\right] \tag{2.20}$$

The optimal policy is obtained by selecting the action that maximises the Q-value [34]:

$$\pi^*(\mathbf{x}) = \arg\max_{\mathbf{u}} Q(\mathbf{x}, \mathbf{u}) \tag{2.21}$$

2.4.4.1 RL Evaluation

One of RL's primary strengths is its ability to learn optimal control policies directly from data. This eliminates the need for a precise mathematical model of the system, which is often complicated and expensive to obtain (see Appendix for Cassie model overview). RL can continuously adapt and improve its control strategy through trial and error, leading to highly optimised and robust performance [34].

RL also excels in handling high-dimensional state and action spaces. Techniques like deep Q-learning and policy gradient methods can leverage powerful function approximators such as neural networks to learn complex policies. This capability is particularly beneficial for robotics applications, where the state space can be very large and the dynamics highly nonlinear [30, 34].

Furthermore, RL algorithms can improve performance over time by collecting data from interactions with their environment. This iterative learning process enables RL to handle uncertainties and adapt to changes in the environment, making it well-suited for tasks that involve continuous adaptation, such as maintaining balance [34].

Although RL also has significant weaknesses, such as the requirement for large amounts of training data. RL algorithms often need millions of interactions with the environment to converge to an optimal policy, which translates to extensive simulation time and high computational costs before deployment. Additionally, training in real-world settings is usually impractical due to the high risk of damage and the cost of failure [30, 34].

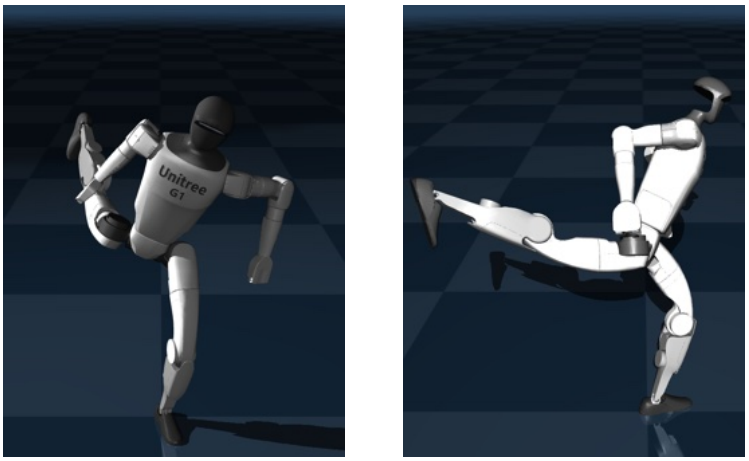
Another issue with RL is the lack of guarantees on stability and safety. Unlike traditional control methods such as a LQR, which offers theoretical assurances on performance and stability, RL methods often depend on empirical results. This absence of theoretical guarantees poses challenges for safety-critical applications [30]. Additionally, RL requires careful design of reward functions to ensure that the learned policy aligns with the desired behaviour. Poorly designed rewards can lead to suboptimal or unintended behaviours, particularly concerning in complex systems like biped robots [30].

2.5 Software

2.5.1 MuJoCo

MuJoCo, short for Multi-Joint Dynamics with Contact, is a powerful physics engine designed to simulate the movement of complex objects with joints and interacting environments [27]. It's particularly useful for fields like robotics, biomechanics, animation, and machine learning, where researchers need fast and accurate simulations.

Primarily aimed at programmers, MuJoCo is written in C and provides built-in visualisation tools with a complex interface to calculate physics-related properties [27].



(A) Front View (B) Side View

FIGURE 2.5: One-leg LQR Control of Unitree G1 MuJoCo Model [27]

Beyond its core simulation capabilities, MuJoCo facilitates advanced computations like control design, state estimation, and mechanism development. See Figure 2.5 above to view a MuJoCo G1 Unitree robot model balancing on one leg [27].

In this thesis, MuJoCo will be used to create a full-featured simulation pipeline of the Cassie robot [35]. The application will enable our model to replicate the full-body physical system, better understanding the robot's response to perturbations after being linearised at an initial starting position.

2.5.2 Python and MATLAB

For this experiment, Python was used to control and analyse the robot within MuJoCo. MuJoCo's Python bindings provided a robust simulation environment, enabling the implementation of control algorithms, data analysis, and interaction with the robot model [36, 37].

Data generated in Python was subsequently imported into MATLAB, which was employed to plot robot state trajectories and system responses, providing a clear and intuitive way to analyse the simulation results [38]. All code developed for this project has been uploaded to GitHub and is freely available via the following link.

2.6 Summary and Final Decision

The literature review provides an overview of key developments in biped robotics, explores control methodologies, and highlights fundamental concepts like CoM and ZMP, which are essential for achieving balance. It traces historical advancements from Elektro (1937) to Tocabi (2023), with notable milestones such as Wabot-1's walking and communication capabilities (1973) and ASIMO's advanced bipedal walking (2011) [7, 10]. Cassie (2017) represents a significant step forward in biped robotics, combining an avian-inspired leg design with impressive speed and agility in locomotion.

Building on this foundation, an LQR will be selected as the primary control method for balancing the Cassie robot due to its simplicity, efficiency, and proven performance in linearised systems. It is computationally lightweight, requiring only the solution of a single Riccati equation, and provides a robust baseline for evaluating more advanced control strategies while offering robustness guarantees for the nominal system.

By starting with LQR, valuable insights into the system's dynamics and stability properties can be gained, laying the groundwork for future research to transition to more sophisticated methods if necessary. For instance, if state or input constraints prove critical, MPC can be employed to address these limitations explicitly. Similarly, poor responses to disturbances or unmodelled dynamics can be mitigated by integrating H^∞ control to further enhance robustness.

In essence, as laid out in Section 1.2, Research Objective (1) has been satisfied.

Chapter 3

Methodology

This chapter describes the methodology for balancing the Cassie biped robot on one leg using MuJoCo, outlining several key steps that will be broken down using subsections. The approach involves an experimental evaluation of an LQR, achieved by linearising the full robot dynamics around an equilibrium point. As previously alluded, the infinite-horizon LQR controller offers an optimal solution for tracking a steady-state pose, minimising a quadratic cost on both state error and control effort [28].

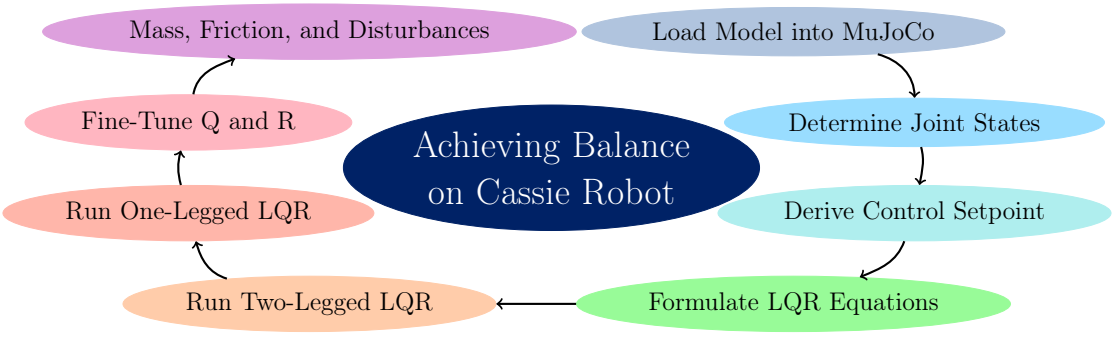


FIGURE 3.1: Methodology Overview for Achieving Single-Leg Balance

As portrayed by Figure 3.1 above, the methodology begins by loading the Cassie model into MuJoCo, and deriving a control setpoint for both one and two-legged stances. Next, LQR equations are formulated and implemented, with initial trials focused on optimising the controller in a two-legged stance through adjustments to

initial conditions. This optimised controller is then adapted for one-legged balance by refining the **Q** and **R** matrices to maximise the operating region with keyframe transitions. Finally, weight distribution adjustments and stability tests against disturbances, and frictional adjustments are applied to ensure robust performance.

3.1 MuJoCo Implementation

To analyse Cassie's balancing capabilities, a detailed simulation model was initialised using Agility Robotics' original Cassie design, released in 2017. The open-source model can be accessed via a repository currently monitored by Google Deepmind under the MuJoCo Menagerie initiative, which includes detailed simulations for various robotics applications such as the franka fr3 and xarm7 robotic arms, the a1 quad walking robot or the Unitree robot (see Figure 2.5) [39].

The MuJoCo physics engine, as detailed in Chapter 2.5.1, was installed using the package provided by Google DeepMind. Installation of MuJoCo was facilitated using Python bindings, enabling seamless integration with the simulation environment. Comprehensive instructions for installation and setup can be found in the 'README' file, accessible via this GitHub repository [40].

3.2 Cassie Robot Components and Properties

Figure 6.1 shows the default simulated state of Cassie, a highly coupled, non-linear, MIMO system with 35 DoF and 10 controlled actuators. Due to its complexity, Cassie presents significant control challenges, including underactuation, dynamic instability, and sensor noise on the physical model [28].

TABLE 3.1: Agility Based Cassie Robot Parts and Corresponding Weights [22,41]

No.	Part	Weight (kg)	No. Parts	Actuated
1	Pelvis	10.33	1	No
2	Hip Roll	1.82	2	Yes
3	Hip Yaw	1.17	2	Yes
4	Hip Pitch	5.52	2	Yes
5	Achilles Rod	0.16	2	No
6	Knee	0.76	2	Yes
7	Knee Spring	0.19	2	No
8	Shin	0.58	2	No
9	Tarsus	0.78	2	No
10	Heel Spring	0.13	2	No
11	Foot Crank	0.13	2	No
12	Plantar Rod	0.12	2	No
13	Foot	0.15	2	Yes
Weight		**33.31**	**25**	-

Table 3.1 lists the key components of Cassie distributed across 25 individual parts, along with their corresponding weights, totalling 33.31 kg. As shown in Figure 3.2, the robot has five motors on each leg for hip yaw, roll, and pitch, knee pitch, and toe pitch [22,41]. Beyond the actuated components, Cassie's design includes several passive elements like springs and connecting rods that play an important role in stability and energy efficiency. For example, the knee spring and heel spring absorb impact forces and store elastic potential energy during locomotion, helping to smooth movement transitions during high-impact actions like walking or running [41].

3.3 State Space Model

State-space models are mathematical frameworks that use state variables to represent a system through a set of first-order differential equations [42]. Instead of describing the system with a single high-order equation, this approach decomposes it into multiple interrelated equations that capture the dynamics of each state variable [42]. Consider a state-space model defined by:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3.1}$$

where:

- $\mathbf{x} \in \mathbb{R}^{64\times1} = \left[x, \dot{x}\right]^T$ represents the state vector
- $\mathbf{u} \in \mathbb{R}^{10\times1}$ represents the control input
- $\mathbf{A} \in \mathbb{R}^{64\times64}$ is the system matrix, capturing the linear dynamics of the state variables
- $\mathbf{B} \in \mathbb{R}^{64\times10}$ is the input matrix, mapping control inputs to the state dynamics

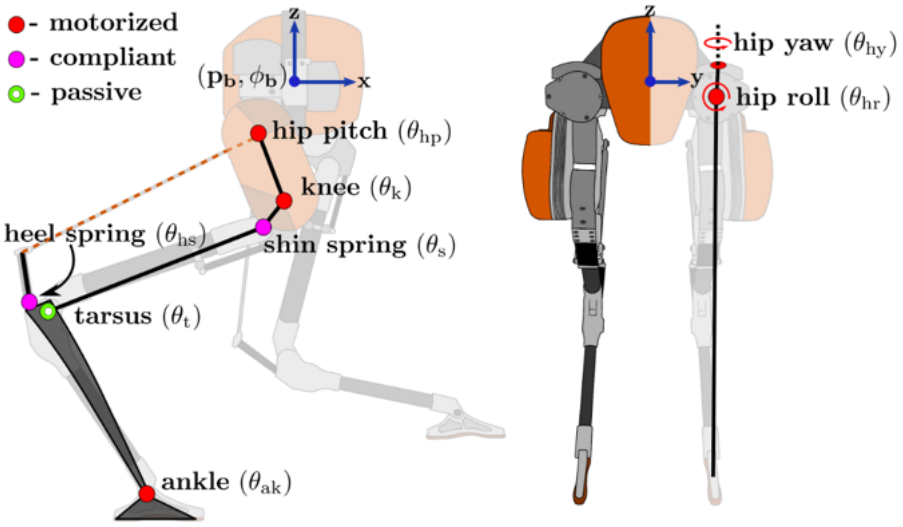


FIGURE 3.2: Cassie Robot Mechanics [8]

The state vector of the robot includes both the positions and velocities of each component listed in Table 3.1. The velocities consist of the linear and angular velocities of the pelvis (movements in x, y, and z directions, and orientations in wx, wy, and wz directions), along with the angular velocities of each joint. The full state vector and the corresponding control input are shown below in Equation 3.2. For clarity, the state vector $\mathbf{x}$ is divided into three components: $\mathbf{x}_p$ representing the pelvis, $\mathbf{x}_l$ representing the left leg, and $\mathbf{x}_r$ representing the right leg.

$$\mathbf{x}_p = \begin{bmatrix} x_{\text{pelvis}} \\ y_{\text{pelvis}} \\ z_{\text{pelvis}} \\ \omega_{\text{pelvis wx}} \\ \omega_{\text{pelvis wy}} \\ \omega_{\text{pelvis wz}} \end{bmatrix}, \ \mathbf{x}_l = \begin{bmatrix} \theta_{\text{hip roll}} \\ \theta_{\text{hip yaw}} \\ \theta_{\text{hip pitch}} \\ \omega_{\text{achilles rod wx}} \\ \omega_{\text{achilles rod wy}} \\ \omega_{\text{achilles rod wz}} \\ \theta_{\text{knee}} \\ \theta_{\text{shin}} \\ \theta_{\text{tarsus}} \\ \theta_{\text{heel spring}} \\ \theta_{\text{foot crank}} \\ \theta_{\text{plantar rod}} \\ \theta_{\text{foot}} \end{bmatrix}, \ \mathbf{x}_r = \begin{bmatrix} \theta_{\text{hip roll}} \\ \theta_{\text{hip yaw}} \\ \theta_{\text{hip pitch}} \\ \omega_{\text{achilles rod wx}} \\ \omega_{\text{achilles rod wy}} \\ \omega_{\text{achilles rod wz}} \\ \theta_{\text{knee}} \\ \theta_{\text{shin}} \\ \theta_{\text{tarsus}} \\ \theta_{\text{heel spring}} \\ \theta_{\text{foot crank}} \\ \theta_{\text{plantar rod}} \\ \theta_{\text{foot}} \end{bmatrix} \tag{3.2}$$

Now, the full state vector $\mathbf{x}$ and control input $\mathbf{u}$ can be defined as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_p \\ \mathbf{x}_l \\ \mathbf{x}_r \end{bmatrix}, \ \mathbf{u} = \begin{bmatrix} u_{\text{left hip roll}} \\ u_{\text{left hip yaw}} \\ u_{\text{left hip pitch}} \\ u_{\text{left knee}} \\ u_{\text{left foot}} \\ u_{\text{right hip roll}} \\ u_{\text{right hip yaw}} \\ u_{\text{right hip pitch}} \\ u_{\text{right knee}} \\ u_{\text{right foot}} \end{bmatrix} \tag{3.3}$$

The most critical states for control are those associated with motorised joints, as outlined above. These motorised joints are directly actuated, allowing explicit control of their velocities and positions. On the other hand, the non-motorised states, such as the achilles rods, shins, tarsus, and plantar rods, are passive and do not

require direct control. Both shin springs and heel springs are still part of the system dynamics but can be given less emphasis in the control design, as they are primarily influenced by the robot's natural motion and internal dynamics. [41].

3.4 Finding Optimal Initial Configuration

3.4.1 Precision Tuning of Initial Robot Position

To achieve balance, the Cassie robot must begin in an optimal position where the CoM of the entire connected body is aligned directly above the stance foot's CoM. This can be achieved by fine-tuning the model's initial states along each axis to establish a precise setpoint. The process begins with adapting the stance leg configuration from the two-leg case as a baseline for initial adjustment. The right hip pitch is then tuned to align the CoM with the foot in the x-direction, while the right hip roll is modified to position the CoM over the foot in the y-direction. Further refinements can be made by adjusting the grounded leg's hip roll to achieve optimal alignment.

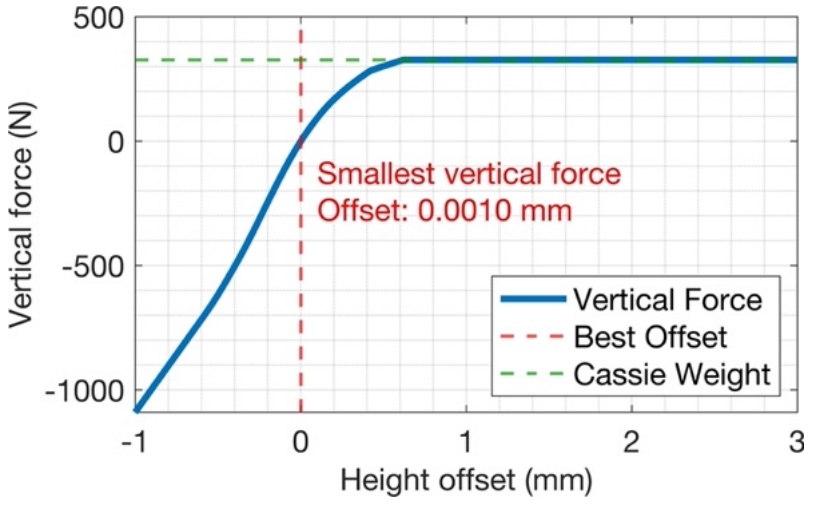
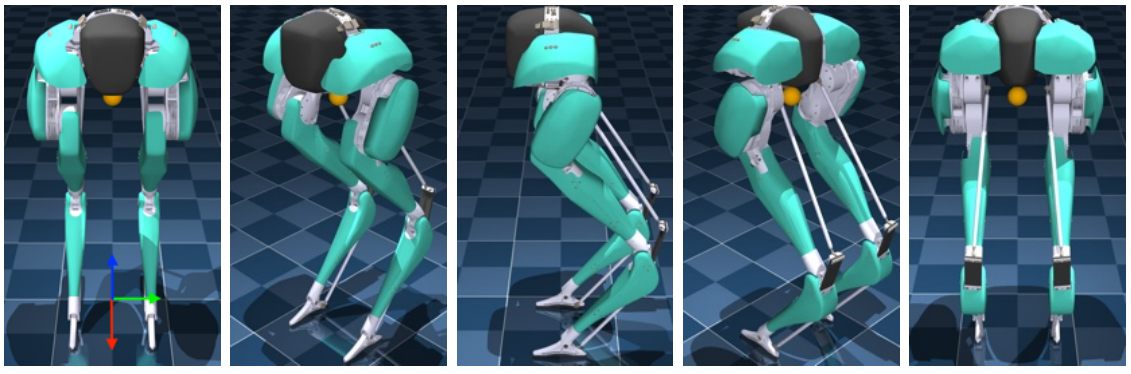


FIGURE 3.3: Height Offset Calculation for Initial Configuration

The pelvis is then shifted vertically to minimise reaction forces between the foot and the ground, ensuring proper balance without external forces. By carefully aligning the reaction forces with the robot's weight, stable contact is achieved without the

need for external corrections. This is illustrated in Figure 3.3, where a minor height offset of 0.001mm is required to minimise the vertical force introduced between the grounded leg and the floor. In Figure 3.3, notice that a large height offset results in the robot being suspended in mid-air, corresponding to a vertical force equivalent to the robot's weight. Once the height is optimally adjusted, it is stored as the initial pose, ensuring the system begins in an optimal position requiring minimal torque.

Upon running the simulation, the robot will stabilise at a steady-state position, which may slightly differ from the previously analysed reference position. This steady-state configuration can then be chosen as the keyframe reference, in which subsequent simulations under these conditions reveal minimal movement, demonstrating that the setpoint is highly accurate.



$\boldsymbol{\theta = 0°}$ $\boldsymbol{\theta = 45°}$ $\boldsymbol{\theta = 90°}$ $\boldsymbol{\theta = 135°}$ $\boldsymbol{\theta = 180°}$

FIGURE 3.4: Robot Visualisation with CoM and X (Red), Y (Green), Z (Blue) Axes

3.5 Finding Initial Control Setpoint

Finding an initial control setpoint for Cassie presented unique challenges due to its underactuated design and the inherent difficulty in selecting specific motor commands from an infinite number of possible solutions [23]. An approach commonly used for fully-actuated robots involves resetting the model to a stable configuration, zeroing the accelerations, and calculating required joint forces via inverse dynamics for the robot to remain in that position. This method typically yields precise

control forces across all joints and, as a result, precise actuator torques. For example, this method has been used extensively in industrial robots like the Puma 560 and the Stanford Arm, both of which employ a series of actuators at each joint to leverage inverse dynamics for accurate movement and task execution along complex trajectories [43].

However, in Cassie's case, certain joints lack actuation, meaning the model cannot directly match desired forces exactly in all joints, leading to inaccurate control setpoints. To address this, a novel cost function $C(\mathbf{x})$ was proposed to minimise discrepancies in acceleration, velocity, and position, expressed as:

$$C(\mathbf{x}) = \mathbf{A}\,(\delta\mathbf{x})^2 + \mathbf{B}\,(\delta\dot{\mathbf{x}})^2 + \mathbf{C}\,(\delta\ddot{\mathbf{x}})^2 \tag{3.4}$$

where,

- A, B, and $C \in \mathbb{R}^{1\times 32}$ represent the weighting matrices on joint positions, joint velocities, and joint accelerations, respectively.

- The element-wise squaring for the state, velocity, and acceleration deviations are defined as:

$$(\delta\mathbf{x})^2 = \begin{bmatrix} (\delta x_1)^2 \\ (\delta x_2)^2 \\ \vdots \\ (\delta x_{32})^2 \end{bmatrix}_{32\times 1}, \quad (\delta\dot{\mathbf{x}})^2 = \begin{bmatrix} (\delta \dot{x}_1)^2 \\ (\delta \dot{x}_2)^2 \\ \vdots \\ (\delta \dot{x}_{32})^2 \end{bmatrix}_{32\times 1}, \quad (\delta\ddot{\mathbf{x}})^2 = \begin{bmatrix} (\delta \ddot{x}_1)^2 \\ (\delta \ddot{x}_2)^2 \\ \vdots \\ (\delta \ddot{x}_{32})^2 \end{bmatrix}_{32\times 1}$$

in which,

- $\delta\mathbf{x}_i$, $\delta\dot{\mathbf{x}}_i$, and $\delta\ddot{\mathbf{x}}_i$ represent the changes in position, velocity, and acceleration, respectively, of the i-th state variable after stepping the simulation forward in time.

Theoretically, the robot should exhibit no changes in position, velocity, or acceleration if the correct control input is applied at a given position, resulting in a cost function value of zero to maintain its upright stance. However, realistically, in experiments, the cost function is rarely perfect (i.e., greater than zero), necessitating the identification of the control combination that brings the cost function as close to zero as possible. Numerous initial control combinations were tested and refined for each configuration using Newton's Method, as detailed below. To evaluate each control combination, bounds on the maximum and minimum controls were applied to ensure realistic actuator limits as portrayed in Figure 5.4. The simulation is then advanced using MuJoCo's `mj_forward` function, which performs physics computations to update the robot's state based on the applied control inputs [40].

The weighting matrices **A**, **B**, and **C** were calibrated using Bryson's rule, which as described in Section 3.6.1.2, proportionally adjusts state variables based on their maximum expected values. This ensures that each variable is appropriately scaled, preventing any one term from disproportionately affecting the overall cost function. However, analogous to the process of establishing a reference position, the control input at steady state was designated as the initial control input for subsequent simulations. This approach facilitated precise determination of control torques, enabling the robot to achieve stabilisation before activating feedback compensation.

3.5.1 Newton's Method

Newton's method is an optimisation technique that iteratively updates a solution estimate using the gradient, which indicates the direction of the steepest increase, and the Hessian matrix, which provides information on the curvature of the objective function [44]. Initially, Newton's method-based optimisation was employed to minimise the cost function described in Equation 3.4, generating an optimal initial control setpoint that ensured the robot remained in a static position for some time before the control loop was implemented [44]. Theoretically, the robot should experience no change in position, velocity, or acceleration if the correct control input is applied to maintain an upright position, provided the position is stabilisable.

The update step in Newton's method is given by:

$$x_{k+1} = x_k - \lambda_k H(x_k)^{-1} g(x_k) \tag{3.5}$$

where $\lambda_k \in [0, 1]$ is a damping factor applied to ensure smoother convergence during the optimisation process, $H(x_k)$ is the Hessian matrix representing the second-order derivatives of the objective function, and $g(x_k)$ is the gradient indicating the steepest ascent direction [44].

3.6 LQR Formulation and Linearisation of A and B Matrices via Centred Finite Difference Method

In our analysis of the LQR in the literature review presented in Chapter 2, Equations 2.4 and 2.5 describe a straightforward LTI system. Now consider a system defined by a set of differential equations with initial state $\mathbf{x}(0)$ [28].

$$f(\mathbf{x}, \mathbf{u}) = \dot{\mathbf{x}}, \tag{3.6}$$

$$g(\mathbf{x}, \mathbf{u}) = \mathbf{y} \tag{3.7}$$

This nonlinear system can be linearised about an equilibrium. If we consider a stable operating point (x_0, u_0), wherein $f(x_0, u_0) = 0$ then,

$$\delta\dot{\mathbf{x}} \approx \frac{\partial f}{\partial \mathbf{x}}\delta\mathbf{x} + \frac{\partial f}{\partial \mathbf{u}}\delta\mathbf{u}, \tag{3.8}$$

$$\delta\mathbf{y} \approx \frac{\partial g}{\partial \mathbf{x}}\delta\mathbf{x} + \frac{\partial g}{\partial \mathbf{u}}\delta\mathbf{u}. \tag{3.9}$$

where

$$\delta\mathbf{x} = \mathbf{x} - \mathbf{x}_e \tag{3.10}$$

$$\delta\mathbf{u} = \mathbf{u} - \mathbf{u}_e \tag{3.11}$$

$$\delta\mathbf{y} = \mathbf{y} - \mathbf{y}_e \tag{3.12}$$

This is an LTI system with

$$\mathbf{A}(t) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}(0),\mathbf{u}(0)}, \quad \mathbf{B}(t) = \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\mathbf{x}(0),\mathbf{u}(0)} \tag{3.13}$$

The Centred Finite Difference Method (CFDM) is employed to linearise the $\mathbf{A}$ and $\mathbf{B}$ matrices around $\mathbf{x}(0), \mathbf{u}(0)$, accurately approximating the Cassie robot's nonlinear dynamics. This method involves evaluating the Jacobian matrices of the system's state equations with respect to the state and input variables at the previously defined equilibrium point [45]. The CFDM provides a second-order accurate spatial approximation, which enhances the accuracy of the linearised model over forward or backward difference schemes [45]. Specifically, the partial derivative of the system function $f(\mathbf{x}, \mathbf{u})$ with respect to a state variable $\mathbf{x}_i$ is approximated as follows:

$$\frac{\partial f}{\partial \mathbf{x}_i} \approx \frac{f(\mathbf{x} + \delta\mathbf{x}_i, \mathbf{u}) - f(\mathbf{x} - \delta\mathbf{x}_i, \mathbf{u})}{2\delta\mathbf{x}_i} \tag{3.14}$$

Similar results hold for $\frac{\partial f}{\partial \mathbf{u}_i}$. Substituting these partial derivatives into the system equations yields a linearised approximation of the system dynamics around the equilibrium [45].

3.6.1 Choice of Q

3.6.1.1 Bryson's Rule

A standard approach for selecting the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ in optimal control design is guided by Bryson's rule, which provides a practical method for normalisation of state and control variables. According to this rule, the maximum allowable values for each state and control signal under typical operation are identified, denoted as $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ for the state variables and $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m$ for the control inputs [46]. By applying these maximum values, the costs become dimensionless, simplifying the weight selection process compared to traditional pole placement techniques. Now, the weights of each state and control signal determine the relative importance of minimising that state, thereby influencing the controller's response [46]. The weighting matrices are then defined as:

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{\bar{x}_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\bar{x}_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\bar{x}_{64}^2} \end{bmatrix} \in \mathbb{R}^{64\times 64}, \quad \mathbf{R} = \begin{bmatrix} \frac{1}{\bar{u}_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\bar{u}_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\bar{u}_{10}^2} \end{bmatrix} \in \mathbb{R}^{10\times 10} \tag{3.15}$$

The above weighting matrix $\mathbf{Q}$ can be further partitioned into position-related and velocity-related components [46].

First, we define $\mathbf{Q}_{\text{pos}}$ for the positional states based on the allowable maximum positional deviations. Similarly, $\mathbf{Q}_{\text{vel}}$ is designed for the velocity-related states following the same logic.

$$\mathbf{Q}_{\text{pos}} = \begin{bmatrix} \frac{1}{\bar{x}_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\bar{x}_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\bar{x}_{32}^2} \end{bmatrix} \in \mathbb{R}^{32\times32}, \quad \mathbf{Q}_{\text{vel}} = \begin{bmatrix} \frac{1}{\dot{\bar{x}}_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\dot{\bar{x}}_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\dot{\bar{x}}_{10}^2} \end{bmatrix} \in \mathbb{R}^{32\times32} \tag{3.16}$$

$$\Rightarrow \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{\text{pos}} & 0 \\ 0 & \mathbf{Q}_{\text{vel}} \end{bmatrix} \tag{3.17}$$

3.6.1.2 Balance Cost

To account for balance in the system, we introduce the difference between the Jacobian $\mathbf{J}_{\text{CoM}}$, representing the whole connected body CoM, and the foot Jacobian $\mathbf{J}_{\text{foot}}$ [46, 47]:

$$\mathbf{J}_d = \mathbf{J}_{\text{CoM}} - \mathbf{J}_{\text{foot}} \tag{3.18}$$

where,

$$\mathbf{J}_{\text{CoM}} \in \mathbb{R}^{3\times32}, \mathbf{J}_{\text{foot}} \quad \in \mathbb{R}^{3\times32}, \text{and} \mathbf{J}_{\text{d}} \in \mathbb{R}^{3\times32}$$

The balance-related component of $\mathbf{Q}$, denoted as $\mathbf{Q}_{\text{balance}}$, is given by:

$$\mathbf{Q}_{\text{balance}} = \mathbf{J}_d^T \mathbf{J}_d \in \mathbb{R}^{32\times32} \tag{3.19}$$

Finally, the overall weighting matrix $\mathbf{Q}$ combines the position, velocity, and balance components:

$$\Rightarrow \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{\text{pos}} + \mathbf{Q}_{\text{balance}} & 0 \\ 0 & \mathbf{Q}_{\text{vel}} \end{bmatrix} \in \mathbb{R}^{64\times64} \tag{3.20}$$

In this way, the whole **Q** cost matrix, as shown in Equation 3.21 above, is designed to regulate the robot's position, ensuring that the CoM of the entire connected body remains over the grounded foot [46, 47].

To simplify notation, **Q** will be defined as:

$$\Rightarrow \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & 0 \\ 0 & \mathbf{Q}_2 \end{bmatrix} \in \mathbb{R}^{64\times 64} \tag{3.21}$$

3.6.2 LQR Implementation

A short code snippet below demonstrates the practical implementation of the LQR controller specifically tailored for the Cassie robot. The initial LQR code script, adapted from an online implementation designed for a 27 DoF humanoid robot, was further customised to suit the Cassie model [48]. The algorithm begins by resetting the Cassie robot model to a predefined two-legged stance and calculating the initial control input required to maintain zero acceleration, ensuring stability. It then constructs the weighting matrices **Q** and **R**, along with the system matrices **A** and **B**, which are crucial for solving the discrete Riccati equation to derive the matrix **P**. Using **P**, the LQR feedback gain matrix **K** was computed, establishing the foundation for determining control actions. In the main loop, the algorithm continuously collects the robot's velocity data, calculates position deviations, and forms a comprehensive state vector $\delta\mathbf{x}$. The LQR control law $\mathbf{u} = \mathbf{u}_0 - \mathbf{K} \cdot \delta\mathbf{x}$ is applied to dynamically adjust the control input based on state errors, with the simulation progressing step-by-step.

Algorithm 1: Main Thread for Cassie LQR Control [48]

Data: Initial forces and control setpoints

Result: Simulate and control Cassie using LQR

Reset model to desired position

Calculate initial control $\mathbf{u}_0$

Compute $\mathbf{Q}$ and $\mathbf{R}$

Compute $\mathbf{A}$ and $\mathbf{B}$

Solve discrete Riccati equation for $\mathbf{P}$: $\mathbf{A}^\top\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}R^{-1}\mathbf{B}^\top\mathbf{P} + \mathbf{Q} = 0$

Solve LQR feedback gain matrix for $\mathbf{K}$: $\mathbf{K} = R^{-1}\mathbf{B}^\top\mathbf{P}$

while *running* **do**

- Get velocity elements (nv)
- Compute position difference ($\delta\mathbf{q}$)
- Differentiate for active DoF's
- Form full state vector $\delta\mathbf{x}$ from $\delta\mathbf{v}$ and $\delta\mathbf{x}$
- Apply LQR control: $\mathbf{u} = \mathbf{u}_0 - \mathbf{K} \cdot dx$
- Render frames

Close video writer

Chapter 4

Results

This chapter presents the results obtained by implementing the methodology described in Chapter 3. The analysis combines simulation renders and MATLAB plots to provide a comprehensive evaluation. An overview of the procedure is included below for clarity and context.

4.1 Procedure

The results are roughly divided into the two-leg and one-leg balancing cases.

1. **Section 4.2**: This section begins with balancing the robot on two legs, demonstrating performance across various keyframe transitions. The initial simulations test the robot on two legs, starting from a height of $z_p = 0.90 - 1.00m$ (keyframe 1-0), with the $\mathbf{Q}$ and $\mathbf{R}$ cost matrices computed using Bryson's rule.

2. Additional tests are performed from keyframes 2-0 ($z_p = 0.82 - 1.00m$), 3-0 ($z_p = 0.74 - 1.00m$), and 4-0 ($z_p = 0.66 - 1.00m$), varying the robot's pelvis height from the ground to observe the effects on stability.

3. Control implementation is then extended to the one-leg case. Initially, $\mathbf{Q}$ matrix values are adjusted to explore the widest operating region.

4. The controller's **Q** and **R** matrices are subsequently fine-tuned for optimal performance, using the largest successful keyframe transition.

5. The final controller is tested using the largest successful keyframe transition (keyframe 6 - 5: z_p = 0.906m - 0.921m), with results demonstrating its effectiveness.

6. **Section 4.3**: This section provides a deeper analysis by modifying design parameters such as mass, external disturbances, and friction to understand their impact on the robot's stability and control.

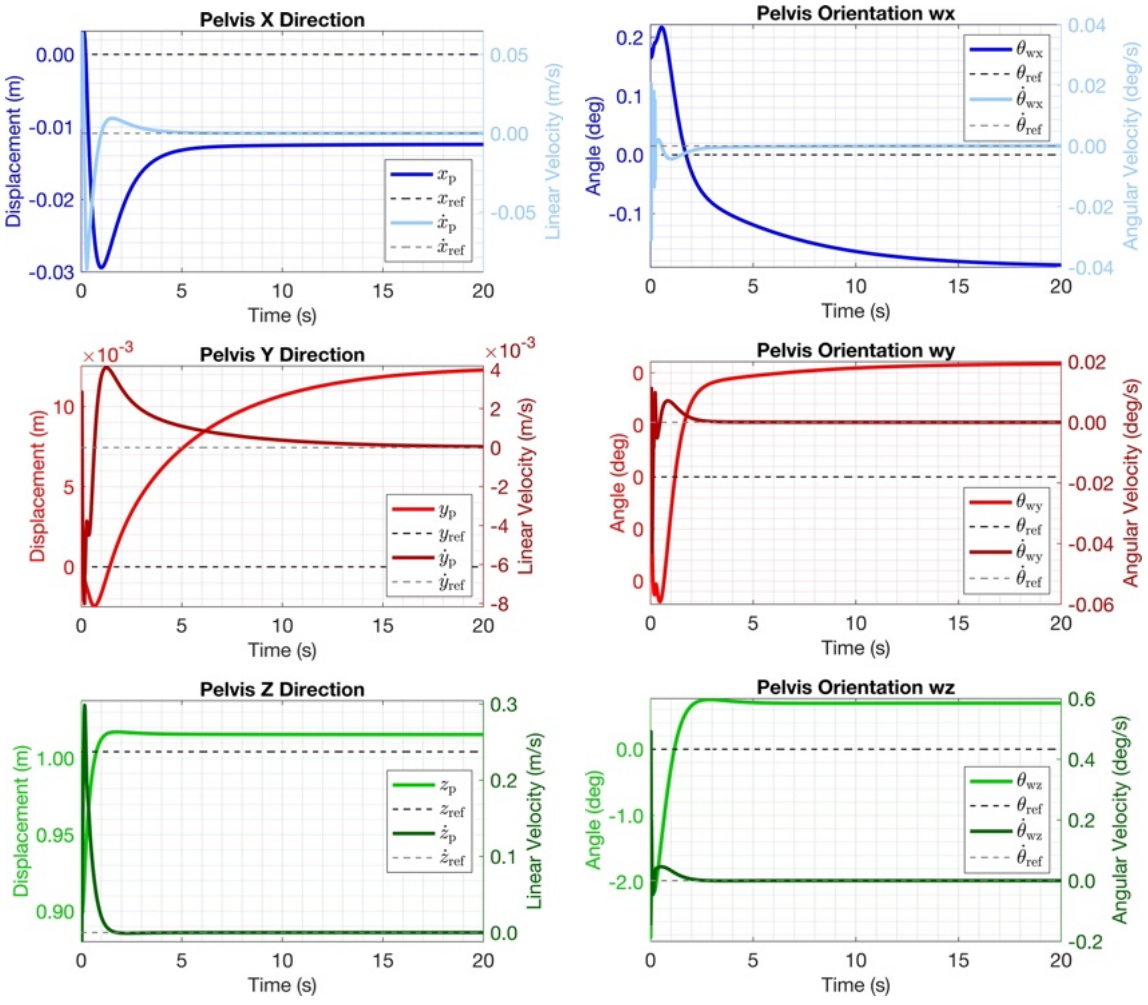
Keyframe 0 $z_p = 1.00$m | **Keyframe 1** $z_p = 0.90$m | **Keyframe 2** $z_p = 0.82$m | **Keyframe 3** $z_p = 0.74$m | **Keyframe 4** $z_p = 0.66$m

FIGURE 4.1: Two-Legged Keyframe Variations with Pelvis Height Adjustment

4.2 Balancing on Two Legs

Figures 4.2 and 4.3 illustrate the performance of the fine-tuned controller, stabilising the robot as it transitions from a height of z_p = 0.90m - 1.00m off the floor. The left axis of each subfigure displays the state positions, while the right axis, in lighter colours, represents the corresponding velocities. The simulation was executed for a total of 20 seconds, and it can be seen that all states converge to a stable solution (see Github video).

Figure 4.4 portrays a demonstration from a pelvis height of z_p = 0.74m - 1.00m off the ground. Note that the controller is able to balance the robot indefinitely with the simulation (as shown in the last frame) running for 20 seconds.



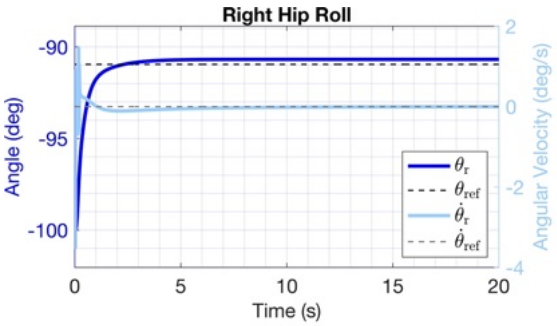
(A) Pelvis Position (B) Pelvis Orientation

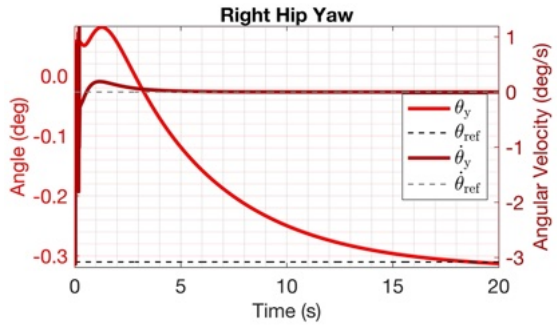
FIGURE 4.2: 20s Simulation of Fine-tuned Two-leg LQR Controller from $z_p = 0.9m$ to $z_p = 1.0m$ (Pelvis Position and Orientation)

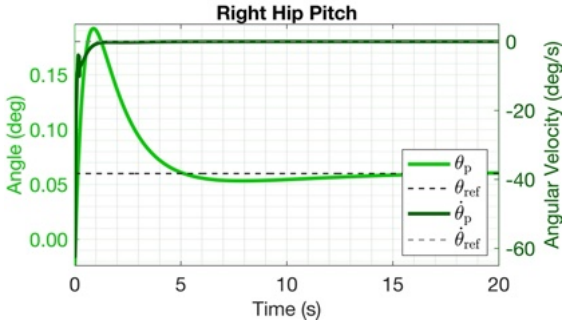
4.2.1 Varying Initial Robot Position

The images in Figure 4.1 illustrate the robot's starting positions at varying heights during distinct keyframes of the simulation. Each keyframe represents a different initial stance or configuration of the robot, showcasing changes in its posture as it transitions between crouching and straightening positions, highlighting variations in height relative to the ground.

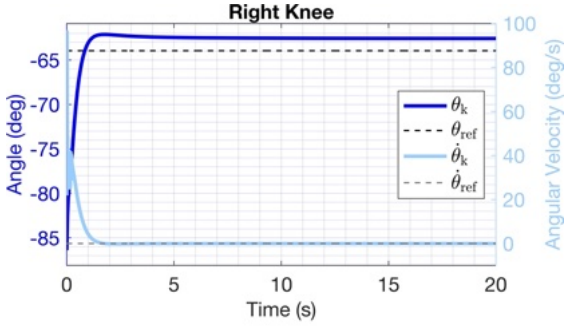
To validate and further fine-tune the controller, simulations were run in 8cm height increments from $z_p = 0.66$m, to a reference height of $z_p = 1.00$m. As shown in Figures 4.5, the controller successfully transitions between states from heights of $z_p = 0.74$m, 0.82m and 0.90m to 1.00m but fails when starting at a height of just

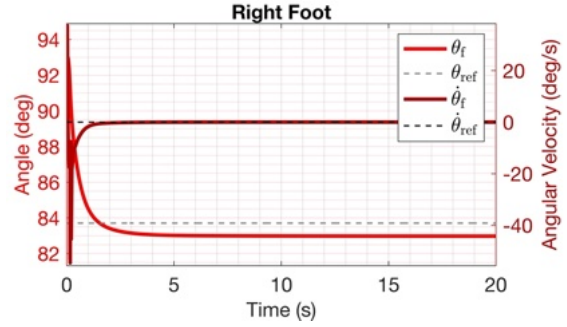






(A) Right Hip





(B) Knee and Foot

FIGURE 4.3: 20s Simulation of Fine-tuned Two-leg LQR Controller from $z_p = 0.9m$ to $z_p = 1.0m$ (Additional States)

$z_p = 0.66$m. Based on the results, the controller was fine-tuned using metrics such as overshoot and rise time. The process was then repeated for the one-legged case, starting from keyframe 5-4.

4.3 Balancing on One Leg

After successfully implementing the controller for two-legged stability, it was subsequently adapted to the one-leg case, requiring certain modifications compared to the two-legged configuration.

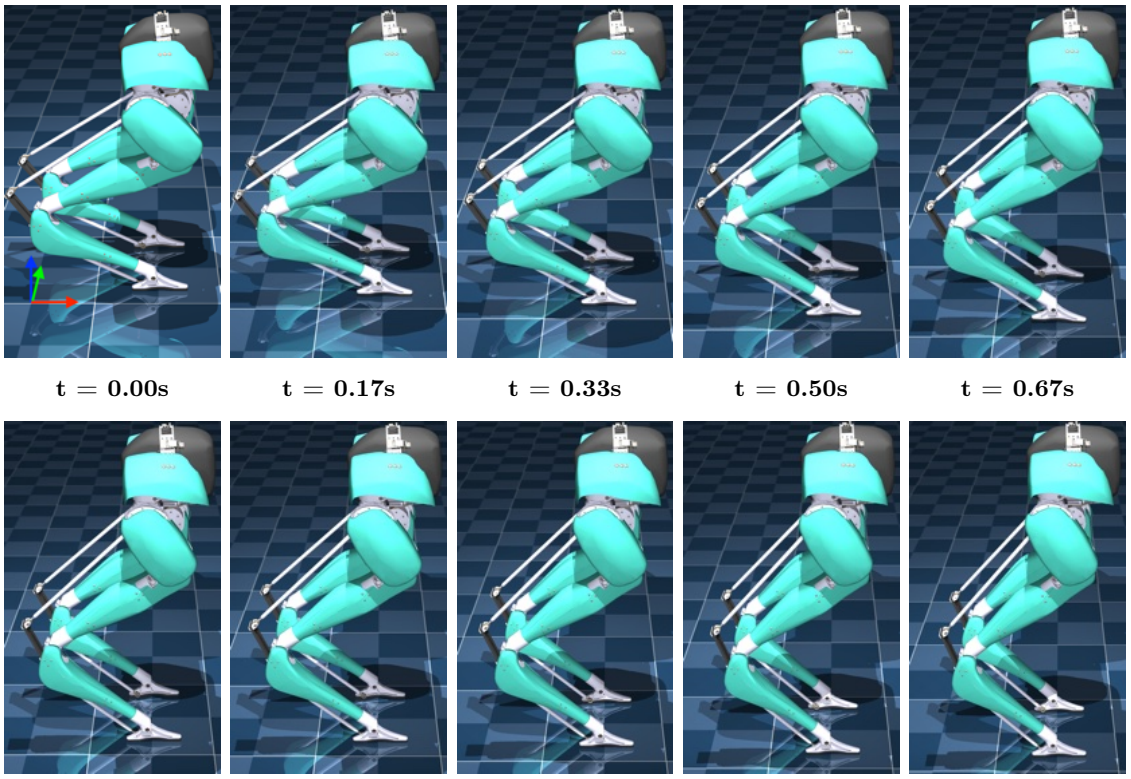
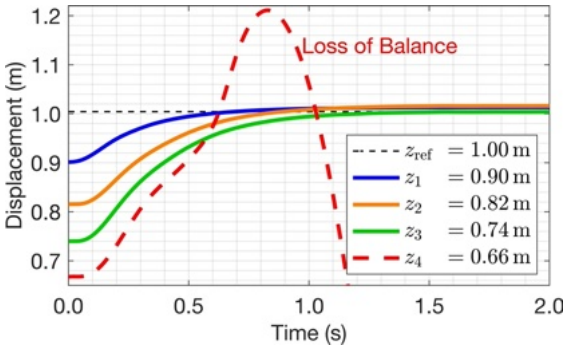
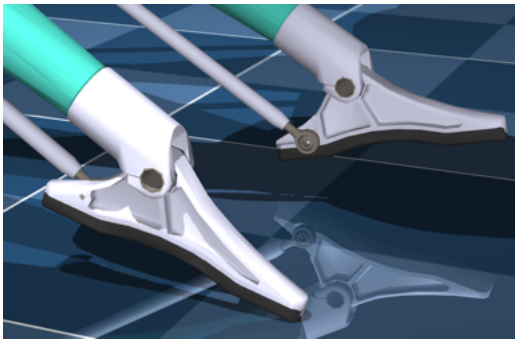


FIGURE 4.4: Transition from Keyframe 3-0 ($z_p = 0.74 - 1.00$m)



(A) Varying Height of Pelvis from $z_p =$ 0.9m to $z_p = 1.00$m



(B) Controller Failure at $z_4 =$ 0.66m

FIGURE 4.5: Simulation Results of Two-leg LQR Controller by Varying Initial Conditions

4.3.1 Optimising Q for Maximum Stable Operating Region

As outlined in Section 1.2, a primary objective of this thesis is to analyse the effectiveness of the proposed control method and assess its feasibility on the physical model. To achieve these objectives, tests were conducted to maximise the robot's stable operating region (measured by the change in CoM) in the x, y, and z directions, as well as evaluating the controller's ability to stabilise from various initial configurations by modifying $\mathbf{Q}$. A larger stable operating region increases the likelihood of successful implementation on the real model, as it accounts for scenarios where the robot may not start from an ideal position.

The approach used involves adjusting the trailing leg's hip roll (for the y direction) or hip pitch (for the x direction). The key strategy is to modify the $\mathbf{Q}$ cost matrix of the LQR controller to enhance stability across a wider range of motion, enabling the robot to maintain balance even when its hip pitch or hip roll angle deviates significantly from the nominal position.

In the y direction, suitable values for $\mathbf{Q}_{\text{CoM}}$ and $\mathbf{Q}_{\text{LHP}}$ are sought to enable the robot to start from a position as far offset from the CoM in the y direction as possible. Similarly, in the x direction, the parameters $\mathbf{Q}_{\text{CoM}}$ and $\mathbf{Q}_{\text{LHR}}$ are adjusted to maximise the initial offset distance from the CoM. Finally, in the z direction, appropriate values for $\mathbf{Q}_{\text{CoM}}$ and $\mathbf{Q}_{\text{RK}}$ are identified to allow the robot to begin from a lower position and transition upward towards the reference trajectory.

Five video demonstrations are provided on Github to visually demonstrate the robot's maximum stable operating region in all directions.

4.4 Controller Fine-Tuning

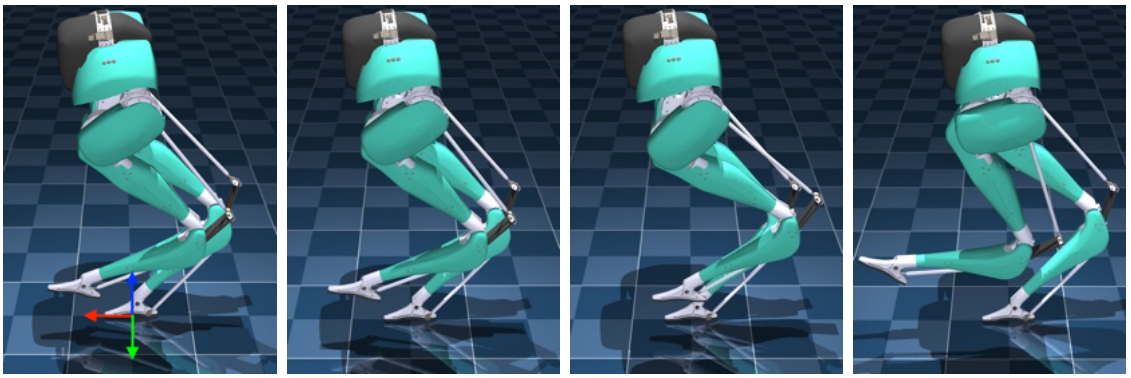
We can further fine-tune the controller to achieve optimum performance after identifying the values in $\mathbf{A}$ and $\mathbf{B}$ that yield the greatest operational margin. As illustrated in Figure 4.8, the coefficient scalers were systematically varied from $\mathbf{Q}_{\text{CoM}} = 500\mathbf{x}$, to $1000\mathbf{x}$, and $2000\mathbf{x}$ to analyse the system response around the CoM. Similarly, the

TABLE 4.1: Optimising Operational Margins in the X-Y-Z Directions by Adjusting the $\mathbf{Q}_{\text{CoM}}$ and Specific Joint Costs. This Table Identifies the Furthest Achievable Starting Positions in Each Direction by Modifying $\mathbf{Q}_1$ (see Section 3.6.1.2)

Direction	**Q Value**	**CoM Difference**	**Furthest Angle (°)**
Positive y (Left Hip Roll Angle)	$\mathbf{Q}_{\text{CoM}} = 1500$	1.86 mm	0.42
	$\mathbf{Q}_{\text{CoM}} = 2000$	1.89 mm	0.46
	$\mathbf{Q}_{\text{CoM}} = 2500$	1.90 mm	0.46
	$\mathbf{Q}_{\text{HR}} = 3$	1.89 mm	0.46
	$\mathbf{Q}_{\text{HR}} = 9$	1.83 mm	0.40
	$\mathbf{Q}_{\text{HR}} = 15$	1.50 mm	0.07
Negative y (Left Hip Roll Angle)	$\mathbf{Q}_{\text{CoM}} = 1500$	-2.54 mm	-3.84
	$\mathbf{Q}_{\text{CoM}} = 2000$	-2.66 mm	-3.95
	$\mathbf{Q}_{\text{CoM}} = 2500$	-2.70 mm	-4.00
	$\mathbf{Q}_{\text{RHR}} = 3$	-2.66 mm	-3.95
	$\mathbf{Q}_{\text{RHR}} = 9$	-2.70 mm	-4.00
	$\mathbf{Q}_{\text{RHR}} = 15$	-2.71 mm	-4.00
Positive x (Left Hip Pitch Angle)	$\mathbf{Q}_{\text{CoM}} = 1500$	11.86 mm	80.21
	$\mathbf{Q}_{\text{CoM}} = 2000$	11.86 mm	80.21
	$\mathbf{Q}_{\text{CoM}} = 2500$	11.86 mm	80.21
	$\mathbf{Q}_{\text{RHR}} = 2$	11.86 mm	80.21
	$\mathbf{Q}_{\text{RHR}} = 8$	11.86 mm	80.21
	$\mathbf{Q}_{\text{RHR}} = 14$	10.24 mm	77.35
Negative x (Left Hip Pitch Angle)	$\mathbf{Q}_{\text{CoM}} = 1500$	-7.54 mm	48.70
	$\mathbf{Q}_{\text{CoM}} = 2000$	-7.90 mm	48.13
	$\mathbf{Q}_{\text{CoM}} = 2500$	-7.65 mm	48.53
	$\mathbf{Q}_{\text{RHR}} = 2$	-7.90 mm	48.13
	$\mathbf{Q}_{\text{RHR}} = 8$	-6.98 mm	49.56
	$\mathbf{Q}_{\text{RHR}} = 14$	-5.69 mm	51.57
Negative z (Right Knee Angle)	$\mathbf{Q}_{\text{CoM}} = 1500$	-1.42 cm	-69.34
	$\mathbf{Q}_{\text{CoM}} = 2000$	-1.47 cm	-70.20
	$\mathbf{Q}_{\text{CoM}} = 2500$	-1.47 cm	-70.20
	$\mathbf{Q}_{\text{RK}} = 32$	-1.23 cm	-69.84
	$\mathbf{Q}_{\text{RK}} = 40$	-1.47 cm	-70.20
	$\mathbf{Q}_{\text{RK}} = 48$	-0.65 cm	-68.92

velocity scalers were adjusted to examine the dynamics of velocity around the CoM, starting from $\mathbf{Q}_{\text{vel}} = 0.01\mathbf{x}$, to $0.1\mathbf{x}$, and $1\mathbf{x}$.

Furthermore, the influence of the trailing leg hip roll weighting is a critical design decision, as this joint primarily contributes to balancing the robot in the y-direction. The results illustrate the $\mathbf{Q}_{\text{HR}}$ cost being scaled by factors of 0x, 3x, 6x, and 9x. The effect of changing the influence of velocity-weighted elements on the $\mathbf{Q}$ cost matrix is



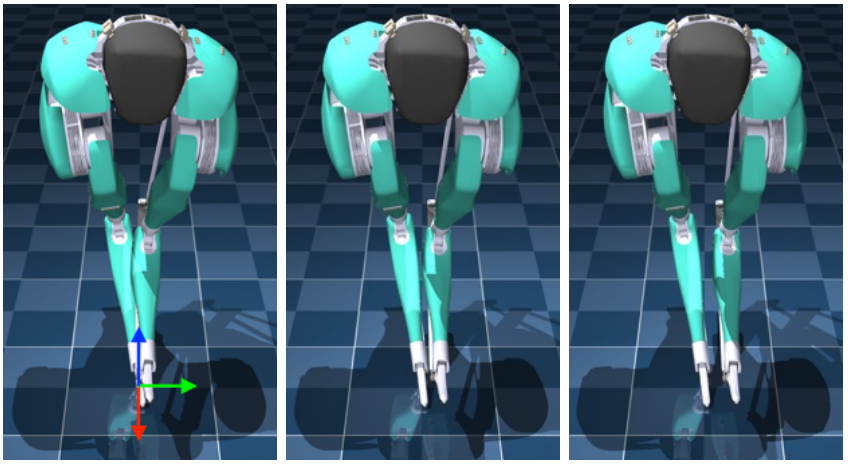
$\boldsymbol{\delta z}_{\text{CoM}} = \mathbf{-1.47}$cm
RK $= -1.91°$

$\boldsymbol{\delta z}_{\text{CoM}} = \mathbf{0}$cm
Target Position

$\boldsymbol{\delta x}_{\text{CoM}} = \mathbf{-0.27}$cm
LHP $= -12.33°$

$\boldsymbol{\delta x}_{\text{CoM}} = \mathbf{1.19}$cm
LHP $= +19.75°$

FIGURE 4.6: Visualisation of Maximum Operating Margin in the X and Z Direction



$\boldsymbol{\delta y}_{\text{CoM}} = \mathbf{-3.21}$mm
LHR $= -2.63°$

$\boldsymbol{\delta y}_{\text{CoM}} = \mathbf{0}$mm
Target Position

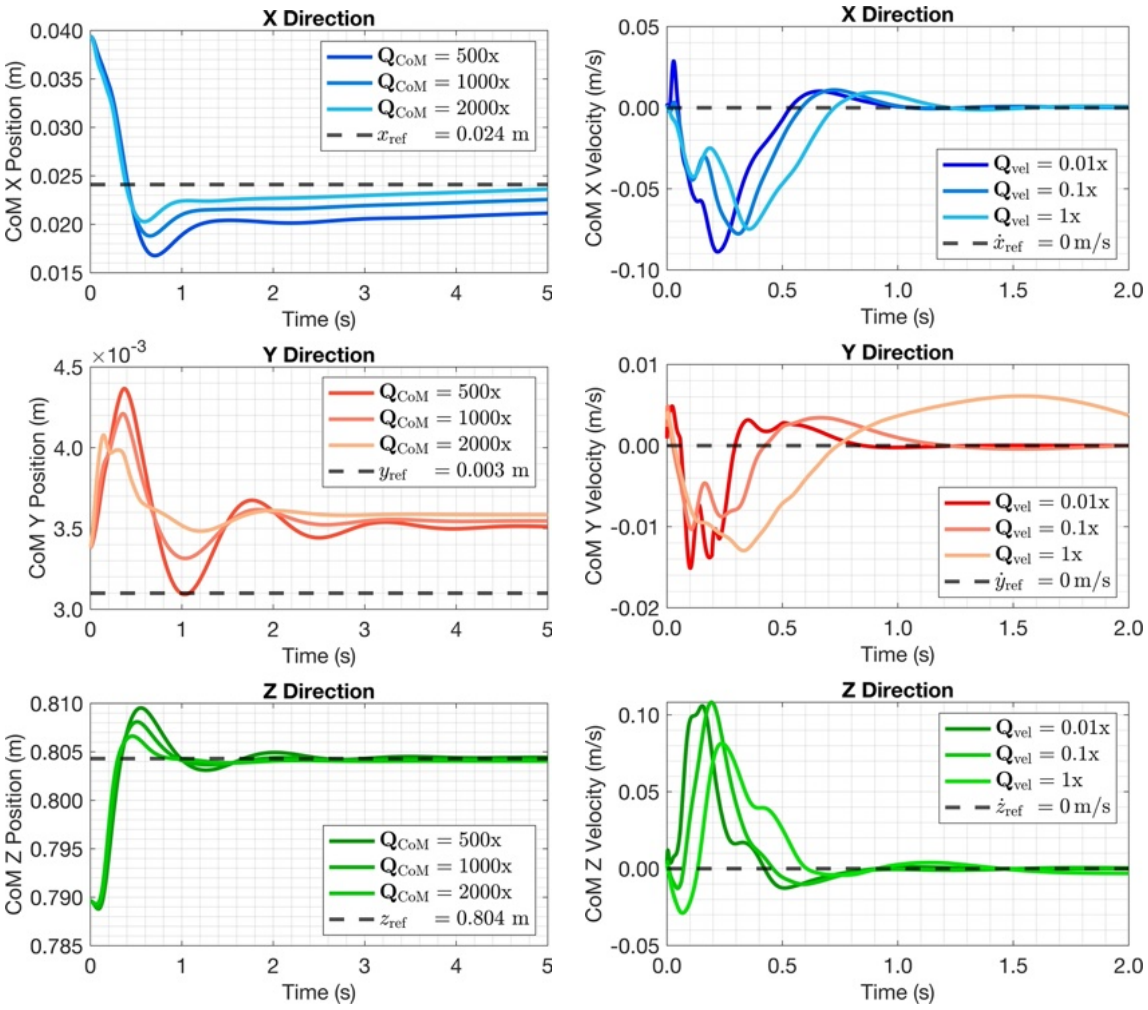
$\boldsymbol{\delta y}_{\text{CoM}} = \mathbf{+5.93}$mm
LHR $= +1.83°$

FIGURE 4.7: Visualisation of Maximum Operating Margin in the Y Direction

displayed in figure 4.9. Figure 4.9a shows the hip roll angle over time, while Figure 4.9b illustrates the right hip roll velocity over the same period.

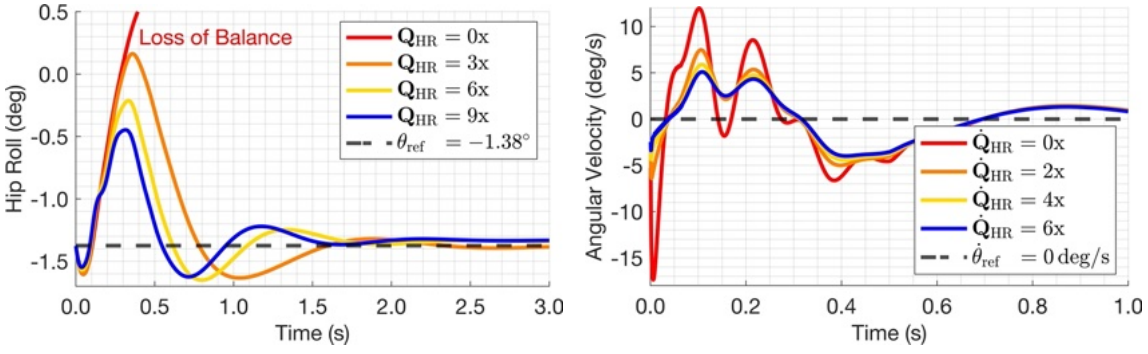
4.5 Controller Design Considerations for Real-World Performance

When designing a robust controller for the Cassie robot, it is essential to account for various real-world factors that could impact its performance. These factors include disturbances from the environment, variations in robot mass, and discrepancies in



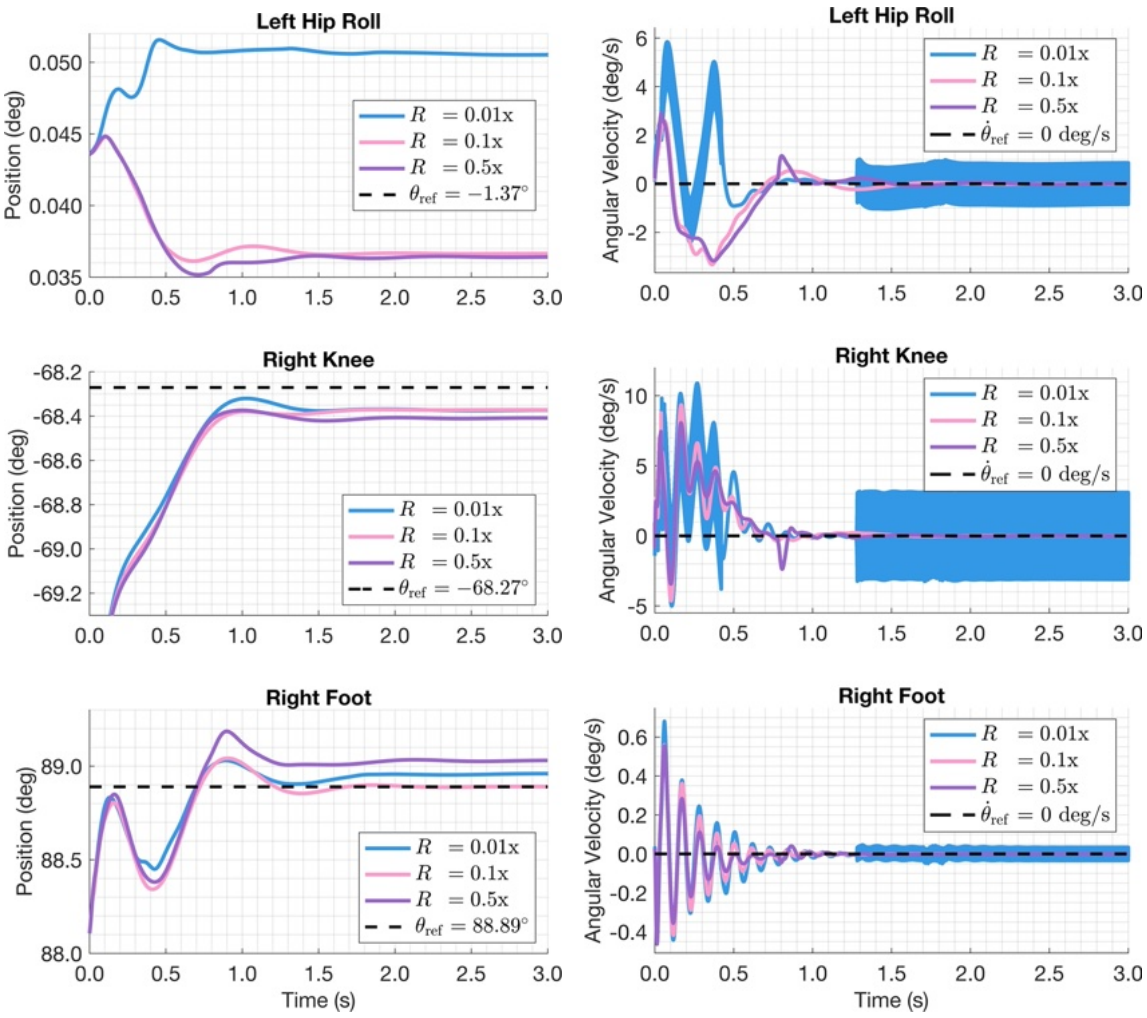
(A) CoM Position (B) CoM Linear Velocity

FIGURE 4.8: Altering **Q** to Analyse Impact upon CoM



(A) Right Hip Roll Position Zoom (B) Right Hip Roll Velocity Zoom

FIGURE 4.9: Right Hip Roll **Q** Weighting Decisions

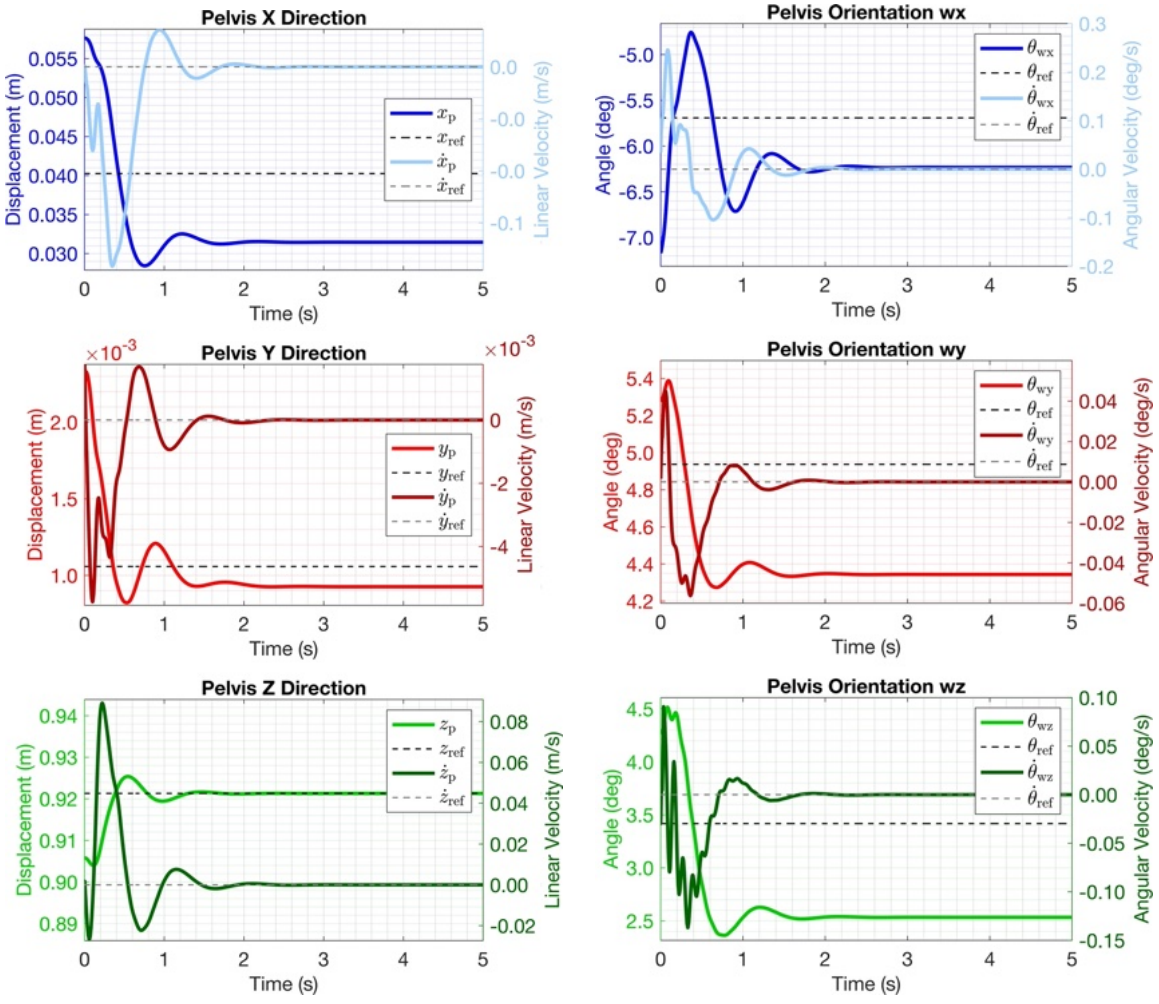


(A) Position of Key States (B) Angular Velocity of Key States

FIGURE 4.10: Effect of Altering **R** Cost Matrix Scaler

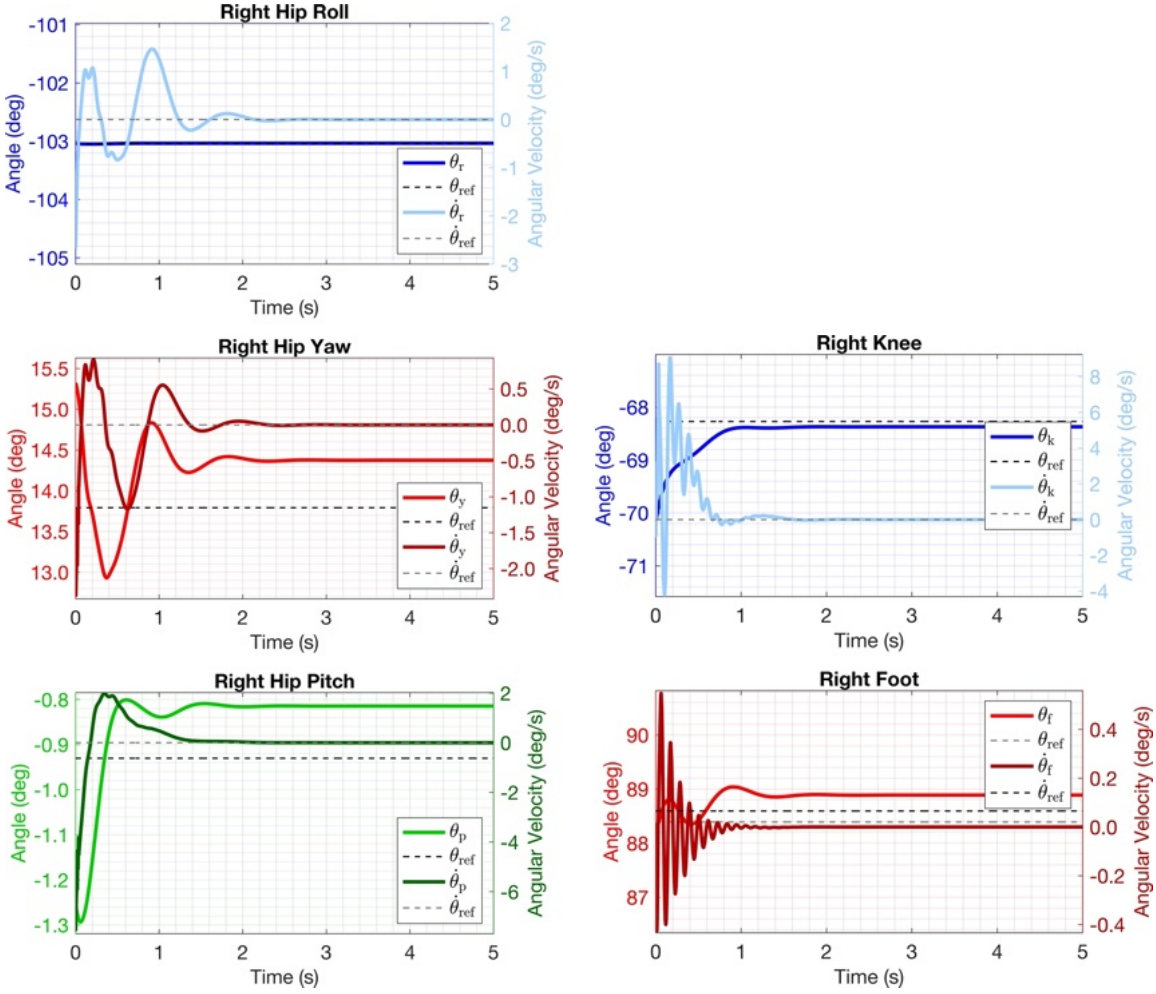
joint friction. Each of these elements can influence Cassie's stability and responsiveness, especially when navigating challenging or unpredictable terrain [7].

In the simulation, forces were repeatedly applied in the positive and negative x and y directions through a trial-and-error process, with force magnitudes gradually increasing until the robot failed to maintain balance. This approach allowed us to identify Cassie's stability limits under different directional forces.



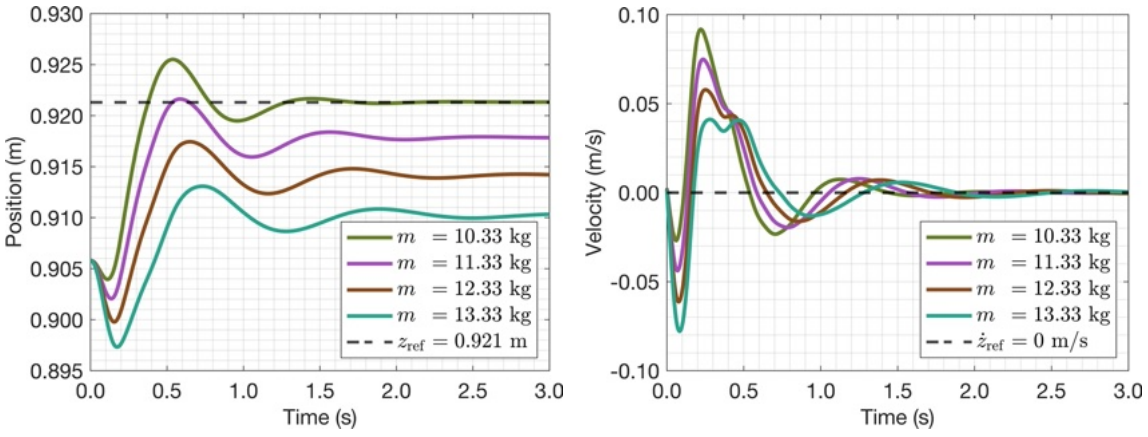
(A) Pelvis (B) Pelvis Orientation

FIGURE 4.11: 5s Simulation of Fine-Tuned One-leg LQR Controller from $z_p = 0.90m$ to $z_p = 0.92m$ (Pelvis Position and Orientation)



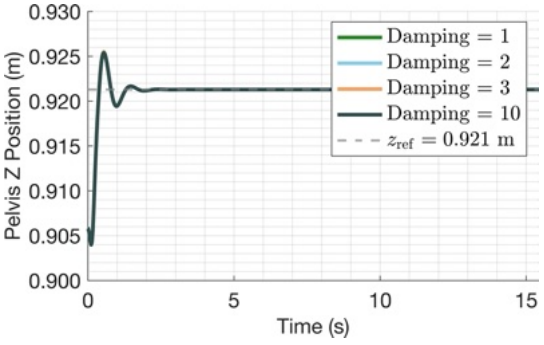
(A) Right Hip (B) Knee and Foot

FIGURE 4.12: 5s Simulation of Fine-Tuned One-leg LQR Controller from $z_p = 0.90m$ to $z_p = 0.92m$ (Additional States)

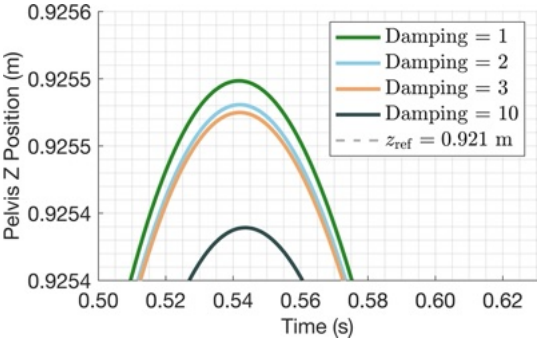


(A) Pelvis Z Position (B) Pelvis Z Velocity

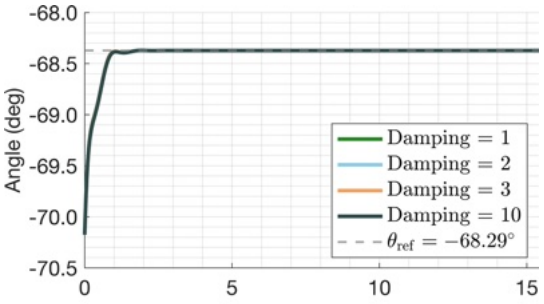
FIGURE 4.13: Effect of Changing Mass on Pelvis

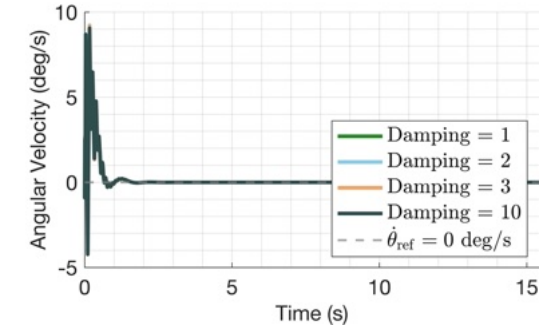


(A) Pelvis Z Position

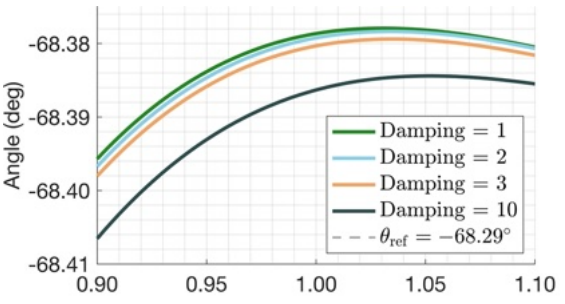


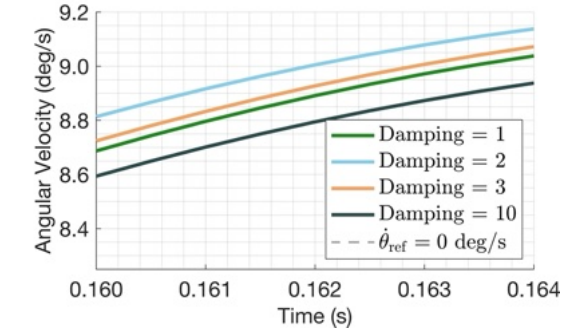
(B) Pelvis Z Position (Zoom)





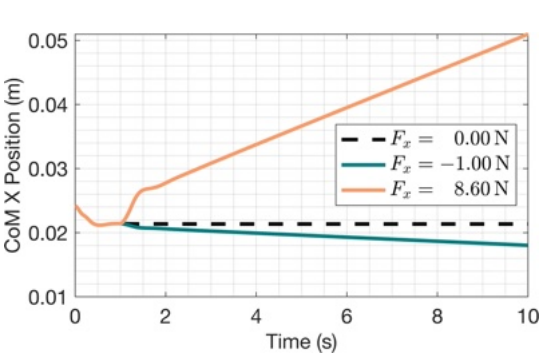
(C) Right Knee Position and Angular Velocity



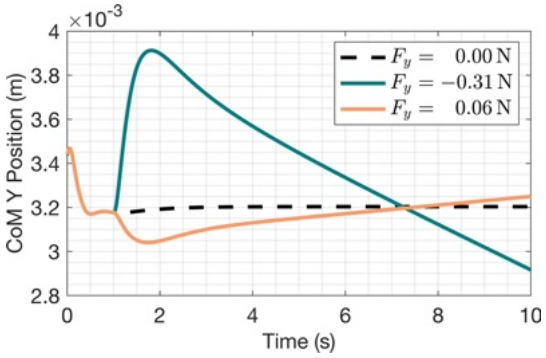


(D) Right Knee Position and Angular Velocity (Zoom)

FIGURE 4.14: Impact of Varying Knee Damping Values on Pelvis and Knee Positions



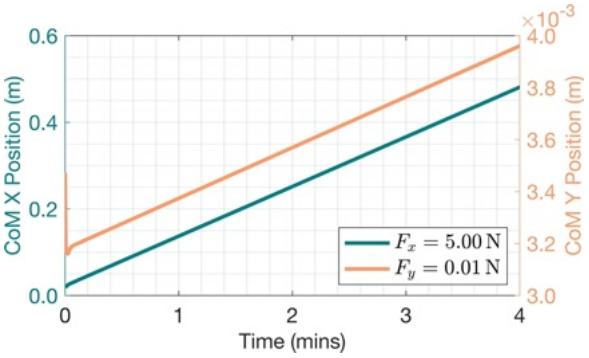
(A) Disturbances Along Positive/Negative X-Direction



(B) Disturbances Along Positive/Negative Y-Direction

FIGURE 4.15: Disturbances Applied on the Whole Body CoM in X and Y-Direction During a Short Time Frame

(A) Force Visualisation on the Robot



(B) Disturbances Along Positive/Negative Y-Direction (Long Time Frame)

FIGURE 4.16: Force Visualisation and Maximum Stabilising Disturbances Applied to the Whole Connected Body CoM During a Long Time Trame in the Y-Direction. Under Small Loads, the Controller Eventually Fails.

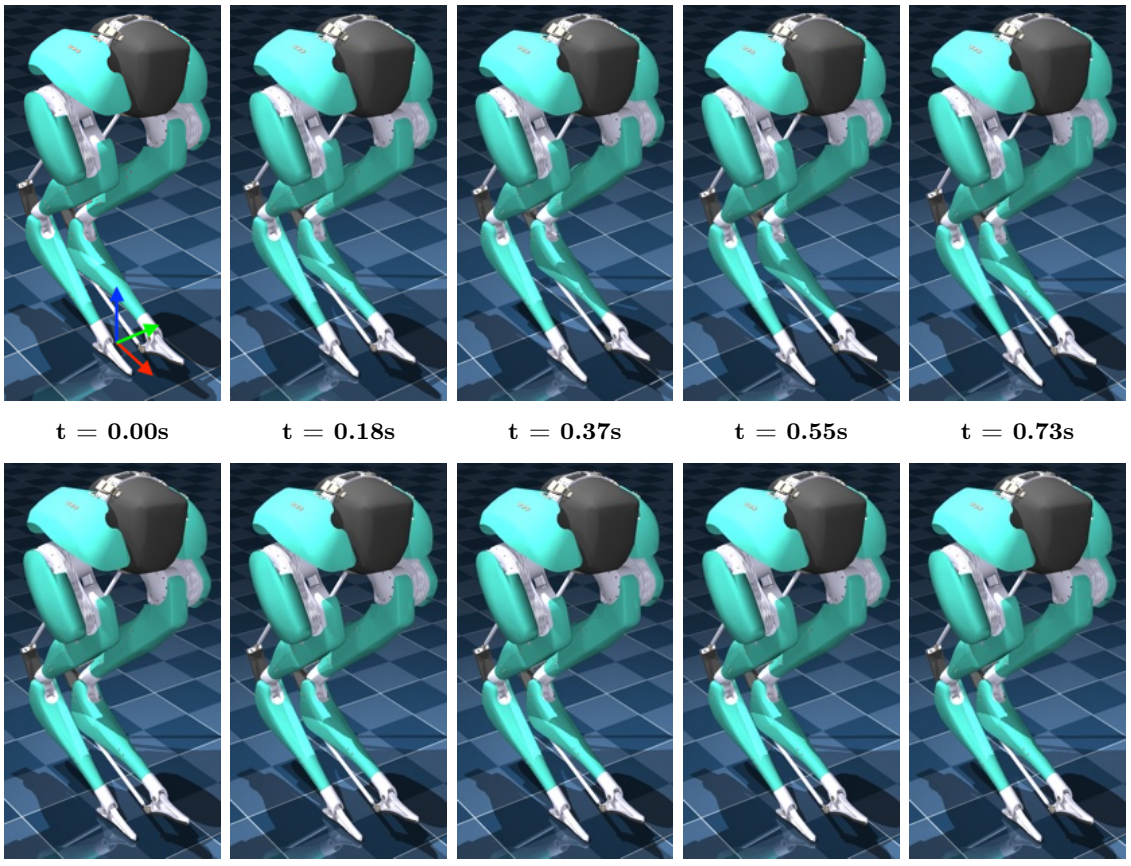


FIGURE 4.17: Simulation of Transition in One-leg Balance Over Time, Showcasing Greatest Height Transition From 0.906 m - 0.921 m (Keyframe 6 - 5). Note That the Last Frame, at $t = 20s$, Demonstrates Successful Indefinite Balancing of the Robot.

Chapter 5

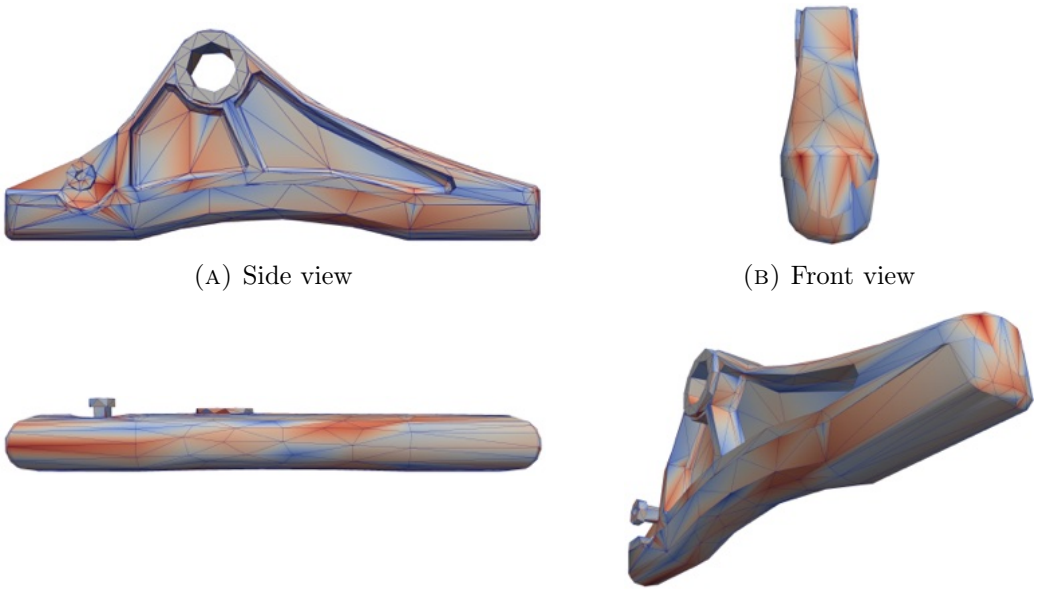
Discussion

This Chapter provides an in-depth analysis of the findings presented in previous chapters, directly evaluating them against the thesis objectives. The focus is on highlighting the unique contributions of this research within the broader academic landscape established in the Literature Review. Additionally, this Chapter critically examines the practicality, feasibility, and potential limitations of the proposed methodologies and findings.

5.1 Finding Robot Reference Position

The simulation was conducted numerous times to identify an appropriate initial state for the robot, with the ultimate goal of achieving convergence to an equilibrium position. This challenge arises primarily due to Cassie's narrow support polygon; its foot design is notably rounded and slender, lacking a traditional 'ankle' joint, which limits its ability to rest flat on the ground (see Figure 5.1).

In the two-leg configuration, it is essential for both feet to maintain consistent contact with the ground during transitions; penalising costs associated with pelvis x-y-z movements at times led to undesirable lateral shifts with contrasting torques from either foot (see Figure 4.5b).



(A) Side view

(B) Front view

(C) Bottom view

(D) Bottom Angled view

FIGURE 5.1: Foot Visualisation to Demonstrate the Narrow Support Base and Emphasise the Difficulty of One-legged Balancing

In the one-leg simulation, a narrow gap of just 2.03cm between the two legs is evident, underscoring limited lateral displacement. This gap at steady state emphasises the precise nature of the task and the final position around which the system is linearised.

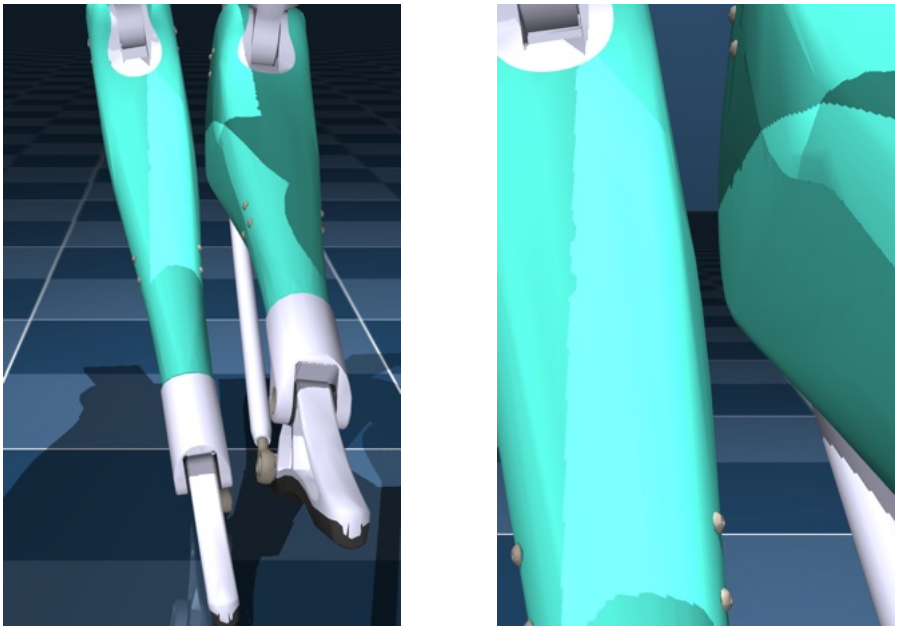


FIGURE 5.2: Gap Between Legs at steady-state

5.2 Evaluation of Initial Control Cost Function

The novel control function introduced in Section 3.5 demonstrated success in most simulation scenarios, and its effectiveness was evaluated across various initial configurations. The functions effectiveness was assessed qualitatively by witnessing if the robot remained unmoved for a short period before falling (after rendering frames), and quantitatively by examining the final number generated from the cost function. As shown in Table 5.1, lower costs likely indicate a greater chance of success.

TABLE 5.1: Keyframe Analysis: Cost Evaluation, Success Rate, and Pelvis Height

Keyframe	**Cost**	**Success**	**Pelvis Height (m)**
0 (Two-Leg)	980.34	Yes	0.90
1 (Two-Leg)	1023.45	Yes	0.80
2 (Two-Leg)	1350.78	No	0.72
5 (One-Leg)	1220.56	Yes	0.92
6 (One-Leg)	1349.23	No	0.90

As outlined above, the proposed cost function achieved a success rate of approximately 60% in simulation. While this novel strategy was reasonably successful in simulation, it faced notable limitations, such as inaccuracies in the weighting matrices **A**, **B**, and **C**. These imprecisions highlight the need for more sophisticated methods to ensure reliable real-world implementation. For example, in this paper, an output-zeroing controller was successfully used to balance an acrobat Inverted Double Pendulum (IDP), requiring the calculation of the third derivative of angular momentum to determine the initial torque [49]. Despite these challenges, the control input generated by the feedback control law was adopted in simulation as a refined initial setpoint after the robot stabilised at the reference position.

5.3 Two-leg Balance

5.3.1 Q Cost Matrix Discussion for Two-leg Balance

As detailed in Section 3.6.1.2, Bryson's Rule was employed to construct the **Q** cost matrix, targeting both control inputs and state variables to align with the stability and mobility requirements of the Cassie robot. The final diagonal elements are presented in the Appendix (see Equation 3.13), and calculations are provided in this Excel file on GitHub. Significant weightings were assigned to the pelvis translational positions, such as Pelvis x and y, with values set at 625.00 for the joint states, reflecting a maximum movement of just 4cm. These high weights prioritised minimising deviations to preserve balance, ensuring the CoM remained stable and correctly aligned [1].

Conversely, the rotational components of the pelvis (q_x, q_y, q_z) were assigned lower weights (32.83) to balance stability and flexibility, as minor rotations minimally affect the overall balance. Similarly, reduced weights (1.62) for the hip and knee joints prioritise flexibility for dynamic leg movements like transitions and agile tasks [1].

5.3.2 Two-leg Simulation Discussion

Table 5.2 presents the performance metrics for a final 20s two-leg demonstration from a pelvis height of $z_p = 0.90m - 1.00m$, after the controller was fine-tuned. The rise time for various metrics, which indicates how long it takes for a system to steer its motion from $0.1 \cdot \mathbf{x}(\infty)$ to $0.9 \cdot \mathbf{x}(\infty)$, varies to ensure a balance between rapid response and controlled stabilisation [50]. Notably, the pelvis x and y positions exhibit longer rise times (4.8s and 8.1s), allowing for deliberate horizontal realignment, while the pelvis z position and right knee angle achieve quick vertical alignment with shorter rise times (0.9s) to achieve the desired height.

TABLE 5.2: Final Two-Leg Performance Metrics: 20s simulation

Metric	**Rise Time (s)**	**Overshoot (%)**
Pelvis x Position	4.8	149.1
Pelvis y Position	8.1	0
Pelvis z Position	0.9	0.3
Pelvis w_x Orientation	8.9	0.0
Pelvis w_y Orientation	6.1	0.0
Pelvis w_z Orientation	1.6	5.8
Right Hip Roll	2.1	0.0
Right Hip Yaw	12.5	0.0
Right Hip Pitch	4.6	16.7
Right Knee	0.9	1.5
Right Foot	2.4	0.0

The maximum overshoot, defined as the highest percentage by which the system exceeds its final value, appears large in the case of the pelvis x position (149.1%) and right hip pitch (16.7%). However, this is primarily because the reference states for these variables are minor, making even small absolute deviations seem proportionally significant [50]. Conversely, minimal or zero overshoot in metrics like the pelvis y direction and right foot reflect steady and efficient performance. Rotational stability is also evident, with gradual responses in pelvis w_x and w_y, indicating smooth adjustments without abrupt corrections.

5.4 One-leg Balance

5.4.1 Q Cost Matrix Discussion with a Focus on CoM for One-leg Balance

As shown in Section 3.6.1.2, the Q cost matrix for our goal was formulated with a focus on movements around the CoM. However, one promising direction investigated was the use of task-space control that explicitly incorporates both linear and angular momentum into the $\mathbf{Q}$ cost matrix around the CoM, which may improve control accuracy [51]. Studies on human locomotion suggest that momentum regulation is

essential for stable walking, indicating potential benefits for humanoid systems. Recent research has demonstrated successful balancing approaches that regulate both CoM position and momentum simultaneously, showing promising improvements for applications in bipedal balance control [51].

For instance, this approach was applied to the hydraulically actuated, torque-controlled robot Sarcos, where impulses ranging from 4.5Ns to 5.8Ns, with peak forces up to 150N, were successfully countered while balancing on a single support [28]. However, Cassie's design lacks an upper body, making it more challenging to utilise such an approach. In fact, Cassie has a significantly lower mass distribution and is unable to pivot multiple limbs around the torso region compared to robots like Sarcos, which limits the effectiveness of angular momentum adjustments during dynamic movements [28]. The control algorithm for Cassie may also be optimised further for the specific mechanics of its legs and joints, as demonstrated in Section 4.3.1 by varying the hip roll and hip pitch movements. This optimisation makes the integration of angular momentum regulation more complex and potentially less beneficial. For these reasons, it was excluded in the $\mathbf{Q}$ cost matrix, and the focus remained on the robot's CoM as detailed in Section 3.6.1.2.

5.4.1.1 Jacobian Matrices and Their Role in Regulating Stability Around the CoM

As derived in Section 3.6.1.2, Jacobian matrices play a crucial role in regulating the robot's balance by influencing the $\mathbf{Q}$ cost matrix, particularly when managing the velocity relationships between joint movements and the CoM. The Jacobian matrix, J_{CoM}, is used to map joint velocities to the linear and angular velocities of the CoM, enabling precise control of its movement relative to other parts of the robot. Each entry of the Jacobian matrix indicates how a small change in a joint angle affects the velocity of the CoM, making it a vital tool for maintaining stability [51].

In Figure 5.5, the gain matrix for the LQR controller highlights position error terms, referred to as $\mathbf{Q}_1$ in Equation 3.20. The heatmap demonstrates that pelvis rotation states are assigned high weights to prioritise balance. The cost matrix $\mathbf{Q}$ features

values from -1,201 to 1,441, with significant emphasis on reducing positional deviations. This is particularly important because even small positional errors can cause instability; for example, a slight angular deviation in the knee is easily corrected, whereas a lateral shift of 20cm in the CoM from the foot could easily lead to a fall [10, 48].

Jacobian matrices influence how weights are assigned in the cost matrix by translating joint movements into task space adjustments, enabling the determination of effective weighting strategies [1]. Recent research demonstrated that in push recovery for biped robots, the Momentum Jacobian Matrix (MJM) effectively mapped joint velocities to CoM dynamics, facilitating recovery from external disturbances [52]. By incorporating eigenanalysis of the MJM within an optimisation framework, the method enhanced stability across various stances and managed greater impulse forces compared to existing techniques, underscoring the pivotal role of Jacobian matrices in bipedal robot control [52].

5.4.2 Maximising Operating Region via Q Cost Matrix

To enhance robustness, the controller was fine-tuned to ensure stability across a broader operating region. Given Cassie's 32 DoF and 25 distinct parts, multiple strategies were considered for adjusting the CoM in the x-y plane. For instance, the hip pitch angle of the stabilising leg could have been incrementally adjusted while maintaining minimal ground contact force on the foot, creating an offset between the body's CoM and the foot's CoM in the x direction. Similarly, the hip roll angle of the stabilising leg could have been modified to achieve an offset in the y direction. Ultimately, however, it was decided to adjust the movements of the trailing leg to regulate the CoM and the robot's position.

Studies indicate that dynamically coordinating torso and leg motions are particularly effective for one-leg balancing, emphasising the importance of a highly adaptable swinging leg [53]. Furthermore, research highlights the challenges of controlling the kinematic coupling between the CoM and the swinging leg, suggesting that varying

the position of this coupling can lead to improved performance [53]. For example, in 2013, Boston Dynamics demonstrated their humanoid robot, Atlas, using both the swing leg and its arms to adjust posture effectively, highlighting the necessity of coordinated limb movements for balance [23].

Adjusting the robot's left hip roll (LHR) and left hip pitch (LHP) significantly influenced the system's operating point. Through iterative adjustments, modifying elements of the $\mathbf{Q}$ cost matrix—specifically $\mathbf{Q}_{\text{CoM}}$, $\mathbf{Q}_{\text{LHP}}$, and $\mathbf{Q}_{\text{LHR}}$—enabled the identification of configurations that maximised the system's stabilisable range.

Table 4.1 and Figure 4.7 show the widest positions achieved in the y direction. For the positive y direction, a maximum initial CoM difference of 1.896mm was achieved with $\mathbf{Q}_{\text{CoM}} = 2500\mathbf{x}$, corresponding to an LHR angle of 0.464°. In the negative y direction, a maximum CoM difference of -2.707mm was found with $\mathbf{Q}_{\text{HR}} = 15\mathbf{x}$, resulting in an LHR angle of -4.003°. These values indicate that higher $\mathbf{Q}_{\text{CoM}}$ values allow a more extensive initial range in the positive direction, while a higher $\mathbf{Q}_{\text{HR}}$ provides greater range in the negative direction.

For the x direction, as shown in Table 4.1 and Figure 4.6, the widest initial positions were achieved with $\mathbf{Q}_{\text{CoM}} = 1500\mathbf{x}$ for the positive x direction, yielding an CoM difference of 11.856mm and an LHP angle of 80.214°. The negative x direction showed a maximum CoM difference of -7.904mm with $\mathbf{Q}_{\text{CoM}} = 2000\mathbf{x}$ and an LHP angle of 48.128°. These findings suggest that lower values of $\mathbf{Q}_{\text{CoM}}$ in the positive x direction extend the operating range, while in the negative x direction, slightly higher values of $\mathbf{Q}_{\text{CoM}}$ widen the range. Overall, however, tuning $\mathbf{Q}$ did not result in a major increase in the operating range, suggesting inherent limitations in the controller's stabilisable region.

This aligns with real-world applications, where increasing the weighting of certain state variables yields diminishing returns in extending the stabilisable range. For instance, on the lower body of a hydraulic Sacros, researchers found that the stabilisable range of LQR for bipedal systems was significantly limited by sensor noise and restricted control bandwidth, leading to instability at operating points with CoM deviations exceeding 10mm from the nominal position [28].

5.4.3 Time-Varying Re-Linearisation

To enhance the responsiveness of the controller and further broaden the operating range, the matrices **A** and **B** were re-linearised at each time step. This approach allowed the controller to adapt to the updated dynamics instantaneously as the robot moved, ensuring greater precision near the linearisation point, where a linear LQR controller performed optimally. This resulted in the following modifications to Equation 3.13:

$$\mathbf{A}(t) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}(t),\mathbf{u}(t)}, \quad \mathbf{B}(t) = \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\mathbf{x}(t),\mathbf{u}(t)} \tag{5.1}$$

However, an error occurred at approximately 0.4 seconds during the maximum height transition from Keyframe 6-5 (z_p = 0.906-0.921m) due to eigenvalues of the associated symplectic pencil being too close to the unit circle, which disrupted stability. A common method used to address this issue and improve stability was to shift the eigenvalues of the matrix **A** further into the stable region. This was achieved by modifying **A** as follows:

$$\mathbf{A}_{\text{mod}} = \mathbf{A} + \alpha\mathbf{I} \tag{5.2}$$

where $\alpha = \frac{4}{T_s}$ with a settling time of $T_s = 5$. Adding a scaled identity matrix shifted the eigenvalues closer to the origin, improving stability and controller convergence. However, this introduced trade-offs, as increased conservatism slowed the system's response and delayed control inputs. While higher control inputs previously posed no issue within controller limits (see Figure 5.4), the reduced agility and responsiveness prevented the robot from maintaining balance, ultimately compromising performance.

Re-linearising the system at each timestep further increased computational cost. Constant recalculation of the **A** and **B** matrices and updating the controller's dynamics added processing demands, requiring a balance between real-time feasibility and robust performance. At a clock time of 1.0 second, CPU time rose from 12.6 seconds without re-linearisation to 445.7 seconds with it, representing a substantial increase. This trend continued; at 2.0 seconds, CPU time for re-linearisation reached 914.5 seconds, over fifty times that of the non-re-linearised case, leading to noticeable sluggishness in MuJoCo simulations.

TABLE 5.3: CPU Computation Time for Re-linearisation

Clock Time (s)	**No Re-linearisation (s)**	**Re-linearisation (s)**
1.0	12.6	445.7
2.0	16.9	914.5

Consequently, re-linearising was deemed impractical, and the system was linearised once per simulation.

5.4.4 Further Fine-Tuning Controller and Final Weighting Decisions

The controller was tested using the largest height transition adjustment for the following analysis, from $z_p = 0.906\,m$ to $z_p = 0.921\,m$. The simulation is shown in Figure 5.3 below, demonstrating the robots changing CoM over time.

As shown in Figure 4.8, the selection of $\mathbf{Q}_{\text{CoM}}$ had a significant impact on the controller's performance. It was observed that setting $\mathbf{Q}_{\text{CoM}} = 2000\mathbf{x}$ resulted in approximately 52% less overshoot compared to $\mathbf{Q}_{\text{CoM}} = 500\mathbf{x}$ in the x-direction, along with lower rise times and reduced overshoot in all directions. Conversely, the effect of increasing $\mathbf{Q}_{\text{vel}}$ was more ambiguous, though a scaler of $0.01\mathbf{x}$ led to high control effort upon the left hip roll motor, nearly overcoming the formal maximum motor torque as shown in Figure 5.4. Thus, $\mathbf{Q}_{\text{vel}} = 0.1\mathbf{x}$ was selected for a balanced response.

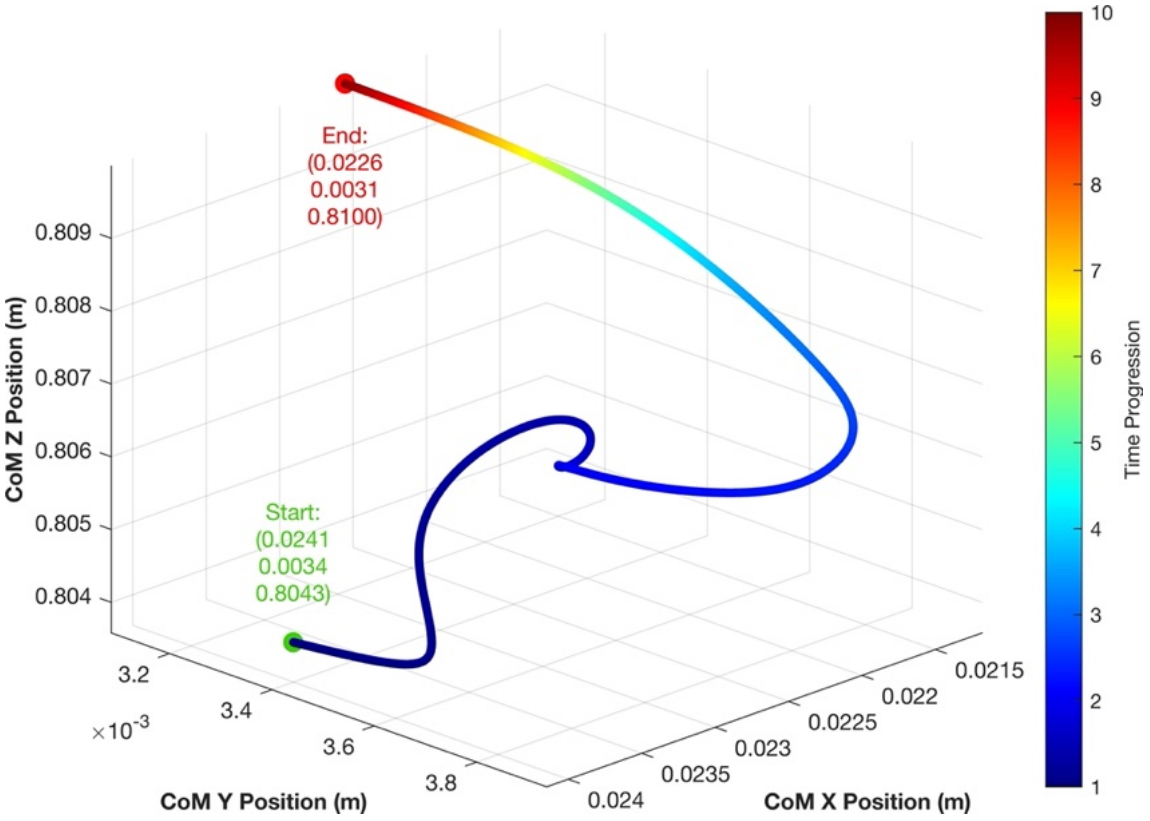


FIGURE 5.3: 3D Plot of Δ CoM for One-Leg Simulation From Lowest Stabilisable Height (Keyframes 5–4: $z_p = 0.906$–0.921 m)

TABLE 5.4: Final Controller Weightings of Key States

Weighting Parameter	**Description**	**Scaler**
$\mathbf{Q}_{\text{CoM}}$	Centre of Mass Position	2000**x**
$\mathbf{Q}_{\text{vel}}$	Velocity (All States)	0.1**x**
$\dot{\mathbf{Q}}_{\text{HR}}$	Right Hip Roll Angular Velocity	2**x**
$\mathbf{R}$	Control Effort	0.1**x**

The absence of any right hip roll weighting led to instability and a loss of balance. As the weighting increased, as shown in Figure 4.9, the hip roll demonstrated a tendency to return to its reference value of -1.38 degrees more rapidly. However, this improvement in responsiveness came at the cost of reduced stability during transitions between different configurations. The increase in $\dot{\mathbf{Q}}_{\text{HR}}$ was observed to effectively dampen the angular velocity. While a slightly elevated angular velocity was deemed acceptable for enhanced balance control, a coefficient of $\dot{\mathbf{Q}}_{\text{HR}} = 2\mathbf{x}$ was selected to achieve an optimal balance between a quicker response and minimisation of abrupt movements.

Finally, the effect of the $\mathbf{R}$ cost scaler on performance was mixed. It had minimal impact on the positions of the left hip roll, right knee, and right foot. However, setting $\mathbf{R} = 0.01\mathbf{x}$ led to severe oscillations for the angular velocities of these states. As shown in Figure 5.4, this resulted in an increase in the maximum control input for the left hip roll from -0.02Nm to 4.44Nm, a level of added motor torque that is only slightly below the formal maximum allowable torque of 4.5Nm. To prevent such oscillations and reduce the torque of the motor while still prioritising the maintenance of the CoM over the foot, $\mathbf{R} = 0.1\mathbf{x}$ was selected.

Table 5.5 identified the maximum and minimum values for each control input, along with the corresponding observed values during the simulation. The formal control limits represented the maximum allowable torques specified for each joint to ensure safe and controlled operation within the mechanical and electrical design limits of the actuators. In contrast, the actual values represented the highest torques exerted by the system throughout the simulation, highlighting the extent to which each joint operated within or close to its formal limits. This comparison provided insight into the demand placed on each actuator, revealing which joints were more frequently or significantly stressed relative to their design constraints.

TABLE 5.5: Control Bounds for Right Leg Compared to Maximum Simulation Values

Control Input	**Units**	**Minimum**	**Maximum**	**Simulation Max**
$\tau_{\text{hip roll}}$	Nm	-4.50	4.50	-1.74
$\tau_{\text{hip yaw}}$	Nm	-4.50	4.50	-0.17
$\tau_{\text{hip pitch}}$	Nm	-12.20	12.20	-0.82
τ_{knee}	Nm	-12.20	12.20	5.97
τ_{foot}	Nm	-0.90	0.90	-0.24

5.4.5 Final Controller Performance

The performance metrics in Table 5.6 highlight the fine-tuned controller's effectiveness in stabilising the one-leg case during a 5-second simulation. Notably, the one-leg scenario achieved quicker stabilisation compared to the two-leg simulation, albeit with a smaller height offset of 1.5cm instead of 26.3cm. This improvement

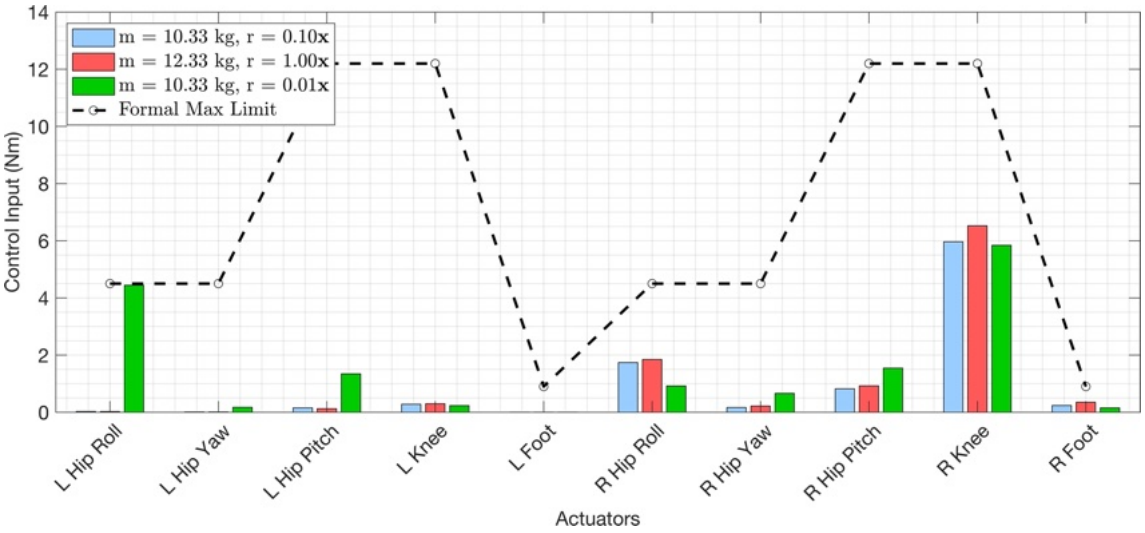


FIGURE 5.4: Absolute Maximum Control Inputs $|\mathbf{u}|_{\max}$ for Differing Simulations

was evident in the lower rise times, such as 1.2 seconds and 1.1 seconds for the pelvis x and y positions, respectively. Overshoot values remained modest, with key states like pelvis orientation w_x showing a manageable 24.1% overshoot, demonstrating a robust and responsive control strategy without compromising stability.

TABLE 5.6: Final One-leg Performance Metrics: 20s simulation

Metric	**Rise Time (s)**	**Overshoot (%)**
Pelvis x Position	1.2	12.5
Pelvis y Position	1.1	15.0
Pelvis z Position	1.0	0.5
Pelvis w_x Orientation	1.6	24.1
Pelvis w_y Orientation	1.2	2.3
Pelvis w_z Orientation	1.0	6.1
Right Hip Roll	0.0	0.0
Right Hip Yaw	1.7	3.1
Right Hip Pitch	0.9	2.4
Right Knee	0.7	0.1
Right Foot	1.0	0.2

The final tuned $\mathbf{Q}$ matrix is shown in Figure 5.5 below. Note that the cost matrix has strong off diagonal elements due to the $\mathbf{Q}_{\text{CoM}}$ cost, as per the design established in Section 3.6.1.2.

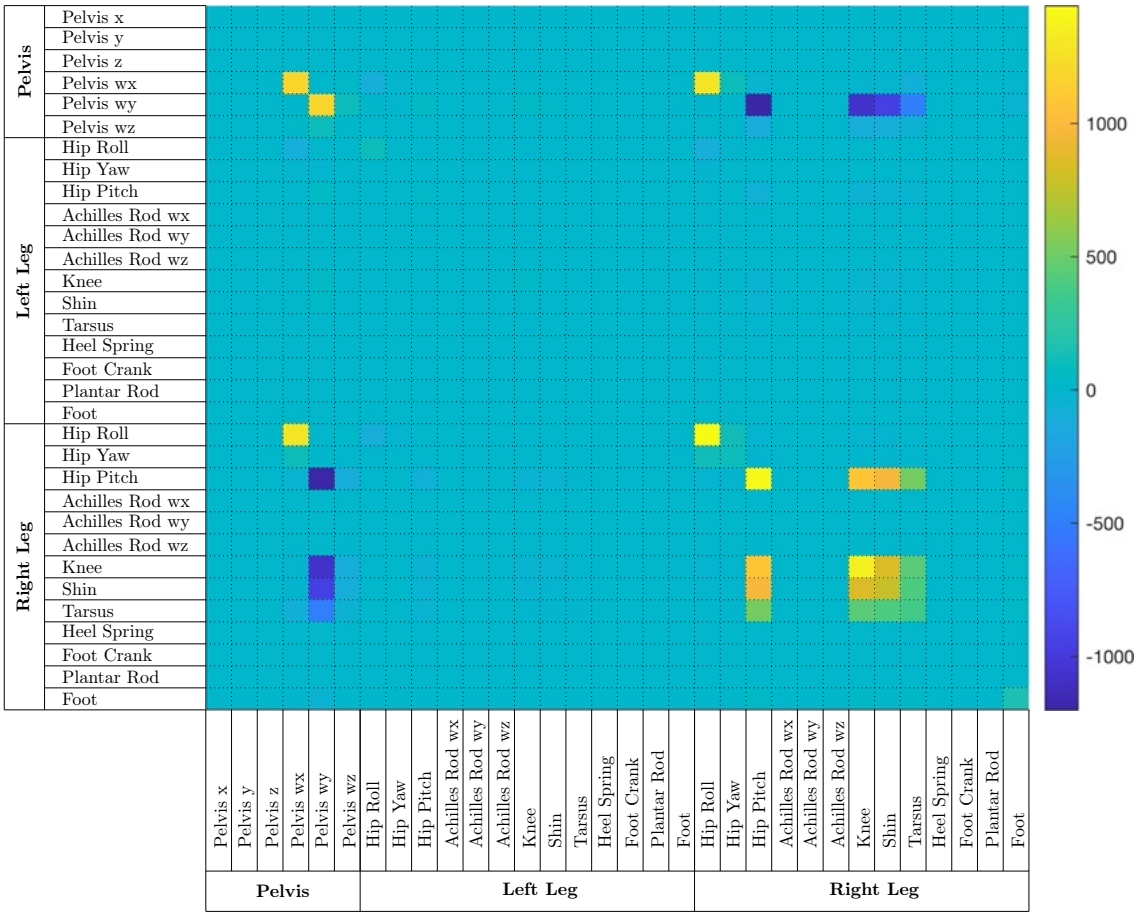


FIGURE 5.5: **Q** Heatmap

5.4.6 Disturbances

External disturbances were critical in evaluating the controller's robustness, as Cassie is likely to face unexpected forces in real-world scenarios. Examples include lateral forces from strong winds in open environments or bumps from interactions with other robots. Disturbances were applied to the robot's CoM at the pelvis, a large surface area particularly susceptible to external forces like wind or collisions with autonomous systems.

Disturbance magnitudes may range from a small nudge to more significant forces that could destabilise the robot if the controller is not well-tuned. The simulation was repeated numerous times, and the robot was manually observed in MuJoCo to determine if it remained balanced. As shown in Figure 4.15, the robot withstood a

maximum force of 0.06N in the positive y-direction and -0.31N in the negative y-direction over a brief 10-second interval (repetitive trials were limited by long rendering times and slow algorithm computation). In the x direction, the robot sustained a maximum force of 8.6N in the positive direction and only 1.0N in the negative direction. Over longer timeframes, however, the performance significantly deteriorated. Regardless of the force magnitude, the robot ultimately failed to maintain balance. Figure 4.16 illustrates the CoM trajectory during a 4-minute simulation under a 5.00N force in the x direction and a 0.01N force in the y direction, where the CoM continuously drifted, indicating poor robustness to sustained disturbances (see Github videos). Attempts to improve performance by tuning parameters such as $\mathbf{Q}_{\text{CoM}}$ and the hip roll and pitch costs were ineffective. This is likely because the controller incurs a higher penalty for attempting to resist the force than for allowing a gradual drift at low angular velocities.

In comparison, hydraulically actuated, torque-controlled robots like the Sarcos humanoid have demonstrated remarkable resilience under external forces. Using two-leg stability, experiments showed that an LQR controller enabled the Sarcos humanoid to resist impacts ranging from 10.4 Ns to 10.7 Ns, with peak forces close to 650 N in the sagittal plane (x-direction), and 5.2 Ns to 6.5 Ns, with peak forces around 200 N in the frontal plane (y-direction) [51]. In single-leg tests, the Cassie robot was able to reject impulses of 4.5 Ns to 5.8 Ns, with peak forces reaching up to 150 N while maintaining balance. This comparison highlights the significant challenge in achieving robust balance on a single leg, particularly for Cassie, whose narrow support base and underactuated design limit its ability to manage external disturbances effectively.

5.4.7 Variations in Robot Mass

The Cassie robot consisted of multiple components, each contributing uniquely to its overall mass and balance. In practical applications, studies showed the weight of each part to vary slightly due to factors such as manufacturing tolerances and material inconsistencies [8]. Additionally, real-world scenarios often required mounting extra

sensors or equipment, such as a laser scanner, camera, or GPU box, may increase weight on the pelvis area.

To evaluate the controller's robustness against variations in mass distribution, the mass of the robot's pelvis was incrementally increased by 1kg across four separate simulations. Initially, the increased payload caused the robot to dip 1.2cm lower before the controller stabilised. Over time, the robot adjusted to these changes, as shown in Figure 4.13, where an additional pelvis mass of 1kg resulted in a final height 0.01m lower than the baseline. The increased mass also led to a slight rise in the time required to reach a stable position, reflecting the controller's adaptive response. Interestingly, the additional weight dampened vertical oscillations in the pelvis, as observed in its z-direction velocity, demonstrating that the controller effectively managed the added load. However, the increased mass also placed higher demands on the actuators. As shown in Figure 5.4, a 2kg increase in pelvis mass led to a rise in the maximum control input for the right knee from 5.97 to 6.53Nm, a level that remains within the robot's manageable torque range.

These results imply that adding equipment like an omnidirectional camera or laser sensor would likely have a minimal impact on controller performance. The Femto-Bolt 3D camera shown in Figure 5.6a and the Leica BLK ARC laser scanner shown in Figure 5.6b represent examples of additional payloads that Cassie could carry, further supporting the model's capacity to handle extra mass without significant degradation in control stability [54, 55].

5.4.8 Effects of Joint Friction

Discrepancies in frictional values between the simulated and real-life Agility Robotics Cassie model are likely, given the complexities of Cassie's mechanical linkages. Such mismatches can significantly impact the controller's performance by altering the robot's response to control inputs, potentially leading to instability or poor transferability when applied to the physical system [56]. The mechanical structure of

(A) FemtoBolt 3D camera [54]

(B) Leica BLK ARC Laser Scanner [55]

FIGURE 5.6: Additional Payload Examples

biped robots like Cassie, which includes transmissions and drive mechanisms, inherently introduces joint friction. This friction can adversely affect performance by causing steady-state errors, limit cycles, and degraded responsiveness, compounding the challenges of controlling the robot's highly coupled degrees of freedom [57].

To evaluate the potential impact of these differences, the damping factor in Cassie's right knee joint was systematically varied for the furthest keyframe height transition from 0.906m to 0.921m. Results in Figure 4.14 show that increasing knee joint damping minimally affects Cassie's pelvis height trajectory. For example, a slight trajectory deviation of 2mm and a minor reduction in angular velocity from 9.23 to 9.05 deg/s was recorded with an extreme damping factor of 10.

A novel approach combining measurement-based and model-based strategies was validated on a 12-DoF biped model, where uncompensated joint friction increased the CoM position error by over 10% [57]. Additionally, compliant joints in biped robots were shown to experience up to 25% higher energy consumption due to frictional forces, particularly in robots utilising a Series Elastic Actuator (SEA), such as COMAN [10, 58, 59]. In the Cassie simulation, however, the impact of friction was mitigated due to the small deviation in the knee angle and the large torque

magnitude. The control torque magnitude of 5.97 Nm was sufficient to counteract frictional effects (see Figure 5.4). Consequently, these results suggest that Cassie's control performance in the simulation would remain consistent and reliable when transitioning to real-world operations, addressing Research Objective (3), as outlined in Section 1.2.

5.5 Practical Considerations and Limitations

5.5.1 Observability and Challenges with Imprecise Velocity Measurements in the Real-World Model

The Cassie robot produces imprecise velocity measurements on the physical model, making it challenging to use velocity elements directly as states in the LQR controller. These inaccuracies can introduce significant errors into the feedback loop, reducing the overall performance and stability of the control system. To mitigate this issue, state estimation techniques such as a Kalman filter or Extended Kalman Filter (EKF) can be employed to provide more reliable estimates of velocity states by fusing sensor data with the robot's dynamics model (as described in Section 6.2.1). However, these methods introduce additional computational overhead and may require careful tuning to ensure robust performance in noisy environments [60]. Balancing the trade-off between estimation accuracy and computational efficiency is therefore a critical consideration for implementing LQR on real-world biped robots like Cassie.

5.5.2 Limited Operating Region

In simulation, it was found that the controller operated effectively only within a narrow range of operating points, primarily due to the robot's narrow feet. When applied to the physical system, this limitation will make it challenging to initiate the robot in a pose suitable for the controller to engage. To address this, a separate

controller could be used to raise the robot using its leading leg, or the robot might need to be suspended mid-air to achieve an appropriate starting position. In separate experiments conducted at USYD, a rope or strap was attached from an overhead frame to the top of the robot's pelvis to prevent damage in the event of a fall. Additional auxiliary supports, such as retractable stabilisers, temporary harnesses, or external frames, could also be employed to maintain balance during initialisation, providing the stability required for the primary controller to function effectively [57].

5.5.3 Imperfect Control Cost Function

The control approach outlined in Equation 3.4 produced mixed results, largely due to the system's underactuated nature. The lack of actuation in certain joints limited the model's ability to achieve the desired control forces, resulting in imperfect stabilisation. To address this, future work should focus on developing a more robust method that enables the model to determine an optimal control setpoint directly, eliminating the need for adjustments after the system converges to a reference value. This refinement will be crucial for real-world implementation, where the method will need to perform reliably on the first attempt.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis has explored advanced control methodologies that enable the Cassie biped robot to balance on one leg. After selecting an LQR approach instead of options such as MPC, H^∞ control, and RL, the problem was formulated and executed using a state-space model. The main findings of this thesis are outlined below.

The LQR approach was chosen due to its simplicity, reliability, and the relatively low computational cost compared to other methodologies such as MPC and RL [61]. While these alternative methods offer advantages in handling constraints and learning-based adaptation, their significantly higher computational requirements made them less suitable for this application, where real-time performance is critical.

6.1.1 Summary of Contribution

Using an Agility Robotics-based model in MuJoCo, this work achieves Research Objective (2) as outlined in Section 1.2, demonstrating the successful balancing of the Cassie robot on both one and two legs in a simulated environment. The findings indicate that through the use of an LQR controller, Cassie can maintain stability

even under challenging conditions. In both cases, the LQR controller stabilised around designated operating points, with the pelvis height maintained 0.92m above the ground.

The controller's performance was evaluated across a range of initial conditions, revealing a defined operational margin. The results showed Cassie could sustain stability with CoM deviations from -3.21 to +5.93mm in the y direction. Additionally, in the x-direction, the controller managed to stabilise within a narrow operational range between -7.90mm to +11.86mm. In the z-direction, Cassie remained stable with a larger downward deviation of -1.47cm.

However, the controller's limitations were also evident, particularly in response to external disturbances. Due to the narrow design of Cassie's foot, the controller lacks robustness against forces in the y-direction, tolerating only between -0.31 N and -0.06 N over a 10-second interval. On the other hand, the x-axis exhibited greater resilience, with the controller enduring forces between -1.00 to +8.60N. The controller demonstrates strong effectiveness in response to changes in pelvis weight of up to 3 kg, and variations in friction have non-negligible effects, indicating that the controller is suitable for application on the physical model.

6.2 Future Work

Future work could expand on current findings by exploring advanced control strategies to address remaining challenges for enhanced stability and robustness.

6.2.1 EKF Combined with a LQR

To address the challenges posed by imprecise velocity measurements in the real-world Cassie model, an EKF can be employed alongside an LQR controller. The EKF is well-suited for Cassie's highly nonlinear dynamics, as it linearises the system around the current state estimate, enabling more accurate state predictions despite sensor noise and model uncertainties. These refined state estimates can then be fed into

the LQR controller, which generates optimal control inputs based on a quadratic cost function.

The EKF operates in two main phases: a prediction step, where the system's state is projected forward using the nonlinear dynamics, and an update step, where sensor data is incorporated to correct the prediction. This iterative process provides a robust estimate of the robot's state, even under noisy or incomplete measurements. By integrating the EKF with the LQR controller, the system can achieve a balance between effective state estimation and optimal control.

For instance, a contact-aided invariant EKF has been demonstrated to provide highly accurate odometry for bipedal robots like Cassie. In a long-term odometry experiment, Cassie was able to walk 200 meters in 7 minutes and 45 seconds while maintaining a low drift rate, with the final position estimate deviating only a few metres from the true position [60]. This approach relied on fusing inertial, contact, and kinematic data without requiring vision-based systems, showcasing its robustness under various environmental conditions.

While this approach can improve Cassie's ability to maintain stability and respond to disturbances, the performance of the EKF heavily depends on accurate modelling of the system dynamics and noise characteristics. Additionally, the need for frequent linearisation and computational overhead may introduce practical challenges, particularly during high-speed or highly dynamic tasks [60].

6.2.2 MPC

Although the current control approach demonstrated stable performance across a range of operating points and transitions, certain scenarios posed control challenges, as discussed in Section 4. In successful simulations, the control inputs remained within their defined limits, allowing for smooth operation. However, a MPC framework could offer a more structured way to enforce critical constraints, such as maintaining continuous foot contact with the ground and preventing the trailing leg from colliding with the supporting leg by limiting hip roll.

For instance, a study involving a bipedal robot demonstrated the effectiveness of an MPC framework by successfully resisting disturbances equivalent to an 11kg mass acting in the x-direction, resulting in a force of approximately 107.8N. This robot, which weighs 45 kg—comprising a 22kg torso and legs weighing 11.5kg each and featuring ten actuated joints similar to Cassie—had dimensions of 0.22m by 0.1m [1]. Consequently, future work could focus on integrating MPC into Cassie to enhance constraint management and overall robustness in diverse environments.

6.2.3 RL

RL has proven effective for handling complex control tasks, making it a promising approach for balancing the robot on one leg. Given the long experimentation times and numerous manual adjustments involved in tuning traditional controllers, RL offers an adaptive solution that could automate these processes. Despite being computationally intensive, RL can significantly speed up simulations when paired with parallel computing, especially by bypassing rendering and visualisation, which can reduce computational overhead and accelerate calculation. This approach could streamline the training phase, enabling rapid policy optimisation wherein the robot could teach itself to balance on one leg [62].

For example, researchers at the University of California, Berkeley have already employed RL techniques to enable Cassie to run 400 metres across various terrains and perform standing long jumps, all without explicit training on each individual movement [6]. In a similar study, it was reported that the RL-trained Atlas robot developed by Boston Dynamics achieved stable locomotion over diverse terrains with a success rate of 90% in simulation, employing a combination of policy gradients and value function approximation [62].

Bibliography

[1] Y. Yang, J. Shi, S. Huang, Y. Ge, W. Cai, Q. Li, X. Chen, X. Li, and M. Zhao, "Balanced Standing on One Foot of Biped Robot Based on Three-Particle Model Predictive Control," *Biomimetics*, vol. 7, no. 4, p. 244, Dec. 2022. [Online]. Available: https://www.mdpi.com/2313-7673/7/4/244

[2] T. Mikolajczyk, E. Mikołajewska, H. F. N. Al-Shuka, T. Malinowski, A. Kłodowski, D. Y. Pimenov, T. Paczkowski, F. Hu, K. Giasin, D. Mikołajewski, and M. Macko, "Recent Advances in Bipedal Walking Robots: Review of Gait, Drive, Sensors and Control Systems," *Sensors*, vol. 22, no. 12, p. 4440, Jan. 2022, number: 12 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/22/12/4440

[3] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-Based Control for Dynamic Legged Robots," Nov. 2022, arXiv:2211.11644 [cs]. [Online]. Available: http://arxiv.org/abs/2211.11644

[4] Y. Ding, C. Li, and H.-W. Park, "Single Leg Dynamic Motion Planning with Mixed-Integer Convex Optimization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, Oct. 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8594161/

[5] J. H. Bong, S. Jung, J. Kim, and S. Park, "Standing Balance Control of a Bipedal Robot Based on Behavior Cloning," *Biomimetics*, vol. 7, no. 4, p. 232, Dec. 2022. [Online]. Available: https://www.mdpi.com/2313-7673/7/4/232

[6] "How AI taught Cassie the two-legged robot to run and jump." [Online]. Available: https://www.technologyreview.com/2024/03/18/1089899/how-ai-taught-cassie-the-two-legged-robot-to-run-and-jump/

[7] A. Goswami, *Humanoid robotics: a reference*. New York, NY: Springer Berlin Heidelberg, 2018. [Online]. Available: https://www.google.com.au/books/edition/Humanoid_Robotics_A_Reference/wlAHnwEACAAJ?hl=en

[8] J. Reher, W.-L. Ma, and A. D. Ames, "Dynamic Walking with Compliance on a Cassie Bipedal Robot," Apr. 2019, arXiv:1904.11104 [cs]. [Online]. Available: http://arxiv.org/abs/1904.11104

[9] "Elektro, the Most Famous Robot of the 1930s : History of Information." [Online]. Available: https://historyofinformation.com/detail.php?id=3129

[10] J. Denny, M. Elyas, S. A. D'costa, and R. D. D'Souza, "Humanoid Robots – Past, Present and the Future," *ResearchGate*, 2016. [Online]. Available: https://www.researchgate.net/publication/303806185_Humanoid_Robots_-_Past_Present_and_the_Future

[11] "Ameca." [Online]. Available: https://www.engineeredarts.co.uk/robot/ameca/

[12] "Robonaut 2 (NASA/General Motors) - Google Search." [Online]. Available: https://www.nasa.gov/robonaut2/nasa-and-gm-take-a-giant-leap-forward-in-robotics/

[13] "Asimo." [Online]. Available: https://robotsguide.com/robots/asimo

[14] "ResearchGate." [Online]. Available: https://www.researchgate.net/figure/A-picture-of-the-iCub_fig1_351368654/download?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6Il9kaXJlY3QiLCJwYWdlIjoiX2RpcmVjdCJ9fQ

[15] "DRC-Hubo+." [Online]. Available: https://robotsguide.com/robots/drchubo

[16] "HD Atlas." [Online]. Available: https://robotsguide.com/robots/atlas2016

[17] "CyberOne." [Online]. Available: https://robotsguide.com/robots/cyberone

[18] P. Sohi, "Meet TOCABI: A Humanoid Robot Designed with Fusion 360," Oct. 2021. [Online]. Available: https://www.autodesk.com/products/fusion-360/blog/tocabi-dyros-njit/

[19] "Bipedal robot developed at Oregon State achieves Guinness World Record in 100 meters," Sep. 2022. [Online]. Available: https://today.oregonstate.edu/news/bipedal-robot-developed-oregon-state-achieves-guinness-world-record-100-meters

[20] "Cassie." [Online]. Available: https://robotsguide.com/robots/cassie

[21] "Cassie Sets a Guinness World Record." [Online]. Available: https://agilityrobotics.com/news/2022/cassie-sets-a-guinness-world-record

[22] "UBCMOCCA/Cassie_robot_resources," Apr. 2024, original-date: 2019-04-28T23:23:43Z. [Online]. Available: https://github.com/UBCMOCCA/Cassie_Robot_Resources

[23] G. Venture, J.-P. Laumond, and B. Watier, *Biomechanics of anthropomorphic systems*, ser. Springer tracts in advanced robotics. Cham: Springer international publishing, 2019, no. 124.

[24] P. Sardain and G. Bessonnet, "Forces Acting on a Biped Robot. Center of Pressure—Zero Moment Point," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 34, no. 5, pp. 630–637, Sep. 2004. [Online]. Available: http://ieeexplore.ieee.org/document/1325327/

[25] "What is center of mass? (article) | Khan Academy." [Online]. Available: https://www.khanacademy.org/science/physics/linear-momentum/center-of-mass/a/what-is-center-of-mass

[26] K. Xu, H. Chen, A. Mueller, and X. Ding, "Kinematics of the center of mass for robotic mechanisms based on lie group theory," *Mechanism and Machine Theory*, vol. 175, p. 104933, Sep. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0094114X22001902

[27] "Overview - MuJoCo Documentation." [Online]. Available: https://mujoco.readthedocs.io/en/latest/overview.html

[28] S. Mason, N. Rotella, S. Schaal, and L. Righetti, "Balancing and Walking Using Full Dynamics LQR Control With Contact Constraints," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov. 2016, pp. 63–68, arXiv:1701.08179 [cs]. [Online]. Available: http://arxiv.org/abs/1701.08179

[29] J. P. Hespanha, *Linear Systems Theory: Second Edition*. Princeton University Press, Feb. 2018. [Online]. Available: http://www.jstor.org/stable/10.2307/j.ctvc772kp

[30] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos, *Optimal control*, 3rd ed. Hoboken, NJ: Wiley, 2012.

[31] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, Dec. 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0005109814005160

[32] K. Glover, "H-Infinity Control," in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. London: Springer London, 2013, pp. 1–9. [Online]. Available: https://link.springer.com/10.1007/978-1-4471-5102-9_166-1

[33] G. Gibson, O. Dosunmu-Ogunbi, Y. Gong, and J. Grizzle, "Terrain-Adaptive, ALIP-Based Bipedal Locomotion Controller via Model Predictive Control and Virtual Constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, Oct. 2022, pp. 6724–6731. [Online]. Available: https://ieeexplore.ieee.org/document/9981969/

[34] "What is Reinforcement Learning? - Reinforcement Learning Explained - AWS." [Online]. Available: https://aws.amazon.com/what-is/reinforcement-learning/

[35] E. Todorov, "Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, May 2014, pp. 6054–6061. [Online]. Available: http://ieeexplore.ieee.org/document/6907751/

[36] "BeginnersGuide/Overview - Python Wiki." [Online]. Available: https://wiki.python.org/moin/BeginnersGuide/Overview

[37] "Welcome to Python.org," May 2024. [Online]. Available: https://www.python.org/about/

[38] "MATLAB." [Online]. Available: https://au.mathworks.com/products/matlab.html

[39] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors, "Mujoco menagerie: A collection of high-quality simulation models for mujoco," 2022. [Online]. Available: https://github.com/google-deepmind/mujoco_menagerie

[40] "Releases · google-deepmind/mujoco." [Online]. Available: https://github.com/google-deepmind/mujoco/releases

[41] "Kinematic Model · agilityrobotics/cassie-doc Wiki," 2018. [Online]. Available: https://github.com/agilityrobotics/cassie-doc/wiki/Kinematic-Model

[42] "What Are State-Space Models? - MATLAB & Simulink - MathWorks Australia." [Online]. Available: https://au.mathworks.com/help/ident/ug/what-are-state-space-models.html

[43] "titania/Papers/[2008 Featherstone] Rigid Body Dynamics Algorithms.pdf at master · create3000/titania." [Online]. Available: https://github.com/create3000/titania/blob/master/Papers/%5B2008%20Featherstone%5D%20Rigid%20Body%20Dynamics%20Algorithms.pdf

[44] "Calculus I - Newton's Method." [Online]. Available: https://tutorial.math.lamar.edu/classes/calci/newtonsmethod.aspx

[45] "Central Difference - an overview | ScienceDirect Topics." [Online]. Available: https://www.sciencedirect.com/topics/engineering/central-difference

[46] A. Sir Elkhatem and S. Naci Engin, "Robust LQR and LQR-PI control strategies based on adaptive weighting matrix selection for a UAV position and attitude tracking control," *Alexandria Engineering Journal*, vol. 61, no. 8, pp. 6275–6292, Aug. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016821007900

[47] B. J. van Hofslot, R. Griffin, S. Bertrand, and J. Pratt, "Balancing Using Vertical Center-of-Mass Motion: A 2-D Analysis From Model to Robot," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3247–3254, Oct. 2019, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/document/8746154

[48] "Google Colab." [Online]. Available: https://colab.research.google.com/github/deepmind/mujoco/blob/main/python/LQR.ipynb#scrollTo=OeHYWdQdm-vE

[49] M. Azad and R. Featherstone, "Angular Momentum Based Controller for Balancing an Inverted Double Pendulum," in *Romansy 19 – Robot Design, Dynamics and Control*, F. Pfeiffer, F. Rammerstorfer, J. Salençon, B. Schrefler, P. Serafini, V. Padois, P. Bidaud, and O. Khatib, Eds. Vienna: Springer Vienna, 2013, vol. 544, pp. 251–258, series Title: CISM International Centre for Mechanical Sciences. [Online]. Available: http://link.springer.com/10.1007/978-3-7091-1379-0_31

[50] A. Haber, "Transient response specifications: Peak time, settling time, rise time, overshoot, and percent overshoot," fusion of Engineering, Control, Coding, Machine Learning, and Science. [Online]. Available: https://aleksandarhaber.com/transient-response-specifications-peak-time-settling-time-rise-time-and-percent-overshoot/

[51] S. Mason, L. Righetti, and S. Schaal, "Full dynamics LQR control of a humanoid robot: An experimental study on balancing and squatting," in

2014 IEEE-RAS International Conference on Humanoid Robots. Madrid, Spain: IEEE, Nov. 2014, pp. 374–379. [Online]. Available: http://ieeexplore.ieee.org/document/7041387/

[52] A. Dan, S. K. Saha, K. Rama Krishna, A. Sawant, G. S. Bhullar, and T. Perween, "Push Recovery of a Biped in Different Stance Scenarios," *Journal of Mechanisms and Robotics*, vol. 17, no. 031015, Oct. 2024. [Online]. Available: https://doi.org/10.1115/1.4066443

[53] M. Yan, K. Liu, R. M. Sohel, R. Ji, and H. Ye, "A Study of Friction Nonlinearity and Compensation for Turntable Servo Systems," *Applied Sciences*, vol. 14, no. 17, p. 8002, Jan. 2024, number: 17 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2076-3417/14/17/8002

[54] "Femto Bolt." [Online]. Available: https://www.orbbec.com/products/tof-camera/femto-bolt/

[55] "BLK ARC | Leica Geosystems." [Online]. Available: https://shop.leica-geosystems.com/global/leica-blk/blk-arc/overview

[56] "Implementing torque control-based biped walking of humanoid robots with high reduction gear and no joint torque feedback," in *IEEE Conference Publication*. IEEE, 2024, iEEE Conference Publication.

[57] I. Hashlamon and K. Erbatur, "Joint friction estimation for walking bipeds," *Robotica*, vol. 34, no. 7, pp. 1610–1629, Jul. 2016. [Online]. Available: https://www.cambridge.org/core/product/identifier/S0263574714002471/type/journal_article

[58] A. Maiorino and G. G. Muscolo, "Biped Robots With Compliant Joints for Walking and Running Performance Growing," *Frontiers in Mechanical Engineering*, vol. 6, Mar. 2020, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/journals/mechanical-engineering/articles/10.3389/fmech.2020.00011/full

[59] H. Yuan, L. Zhou, and W. Xu, "A comprehensive static model of cable-driven multi-section continuum robots considering friction effect," *Mechanism and Machine Theory*, vol. 135, pp. 130–149, May 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X1831468X

[60] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-Aided Invariant Extended Kalman Filtering for Robot State Estimation," Nov. 2019, arXiv:1904.09251 [cs]. [Online]. Available: http://arxiv.org/abs/1904.09251

[61] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback Control of a Cassie Bipedal Robot: Walking, Standing, and Riding a Segway," in *2019 American Control Conference (ACC)*. Philadelphia, PA, USA: IEEE, Jul. 2019, pp. 4559–4566. [Online]. Available: https://ieeexplore.ieee.org/document/8814833/

[62] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to Walk via Deep Reinforcement Learning," Jun. 2019, arXiv:1812.11103. [Online]. Available: http://arxiv.org/abs/1812.11103

Appendix

6.3 Loading Model Into MuJoCo

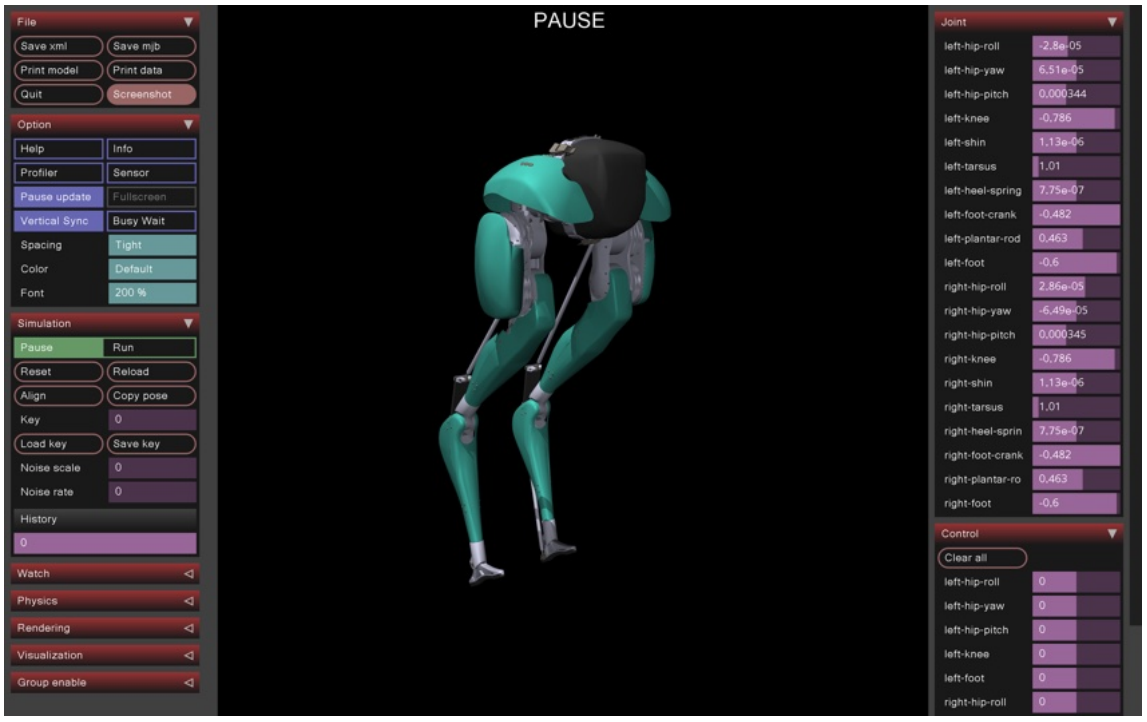


FIGURE 6.1: Initialising Cassie in MuJoCo

6.4 Nonlinear Dynamic Equation of the Cassie Robot

This equation was taken from [51].

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{B}\mathbf{u} + \mathbf{J}_s^T(\mathbf{q})\boldsymbol{\tau}_s + \mathbf{J}_c^T(\mathbf{q})\mathbf{F}_c \tag{6.1}$$

where:

- $\mathbf{D}(\mathbf{q})$: Inertia matrix of the system.
- $\mathbf{H}(\mathbf{q}, \dot{\mathbf{q}})$: Vector containing Coriolis, centrifugal, and gravitational forces.
- $\mathbf{B}$: Input selection matrix mapping control inputs to the system.
- $\mathbf{J}_s(\mathbf{q})$: Jacobian matrix for soft constraints (e.g., springs).
- $\boldsymbol{\tau}_s$: Forces/torques from soft constraints.
- $\mathbf{J}_c(\mathbf{q})$: Jacobian matrix for contact constraints.
- $\mathbf{F}_c$: Contact forces acting on the system.
- $\mathbf{u}$: Control input vector.
- $\mathbf{q}$: Generalised position coordinates of the system.
- $\dot{\mathbf{q}}$: Generalised velocity coordinates of the system.
- $\ddot{\mathbf{q}}$: Generalised acceleration coordinates of the system.

6.5 Bryson's Rule Calculations of Q and R

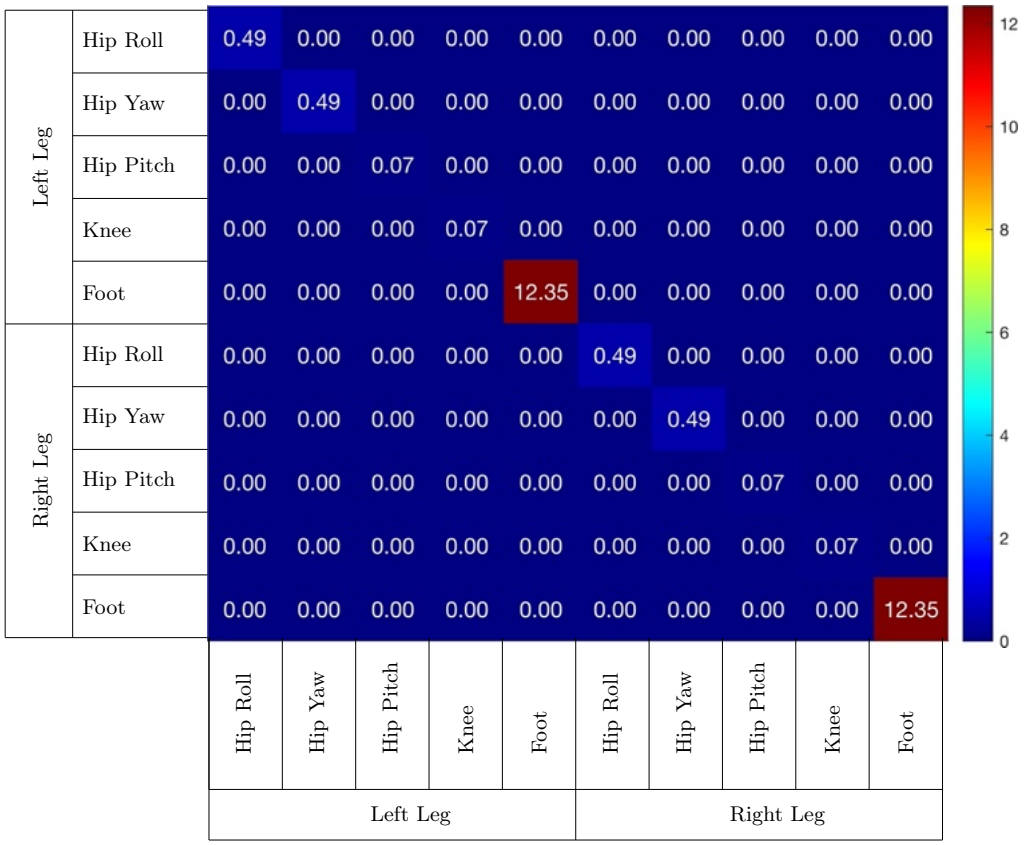


FIGURE 6.2: R Matrix Heatmap - Two-legged Stance

TABLE 6.1: Bryson's Calculation for Positions (Two-leg)

State	**Unit**	**Max Value**	**Bryson's Calculation**
Pelvis x	m	0.04	625.00
Pelvis y	m	0.04	625.00
Pelvis z	m	0.10	100.00
Pelvis orientation wx	deg	10.00	32.83
Pelvis orientation wy	deg	10.00	32.83
Pelvis orientation wz	deg	10.00	32.83
Left hip roll	deg	22.50	6.48
Left hip yaw	deg	22.50	6.48
Left hip pitch	deg	45.00	1.62
Left achilles rod wx	deg	0.00	0.00
Left achilles rod wy	deg	0.00	0.00
Left achilles rod wz	deg	0.00	0.00
Left knee	deg	45.00	1.62
Left shin	deg	0.00	0.00
Left tarsus	deg	0.00	0.00
Left heel spring	deg	0.00	0.00
Left foot crank	deg	0.00	0.00
Left plantar rod	deg	0.00	0.00
Left foot	deg	10.00	32.83
Right hip roll	deg	22.50	6.48
Right hip yaw	deg	22.50	6.48
Right hip pitch	deg	45.00	1.62
Right achilles rod wx	deg	0.00	0.00
Right achilles rod wy	deg	0.00	0.00
Right achilles rod wz	deg	0.00	0.00
Right knee	deg	45.00	1.62
Right shin	deg	0.00	0.00
Right tarsus	deg	0.00	0.00
Right heel spring	deg	0.00	0.00
Right foot crank	deg	0.00	0.00
Right plantar rod	deg	0.00	0.00
Right foot	deg	10.00	32.83

TABLE 6.2: Bryson's Calculation for Velocities (Two-leg)

State	**Unit**	**Max Value**	**Bryson's Calculation**
Pelvis x	m/s	0.10	100.00
Pelvis y	m/s	0.10	100.00
Pelvis z	m/s	0.10	100.00
Pelvis orientation qx	deg/s	25.00	5.25
Pelvis orientation qy	deg/s	25.00	5.25
Pelvis orientation qz	deg/s	25.00	5.25
Left hip roll	deg/s	10.00	32.83
Left hip yaw	deg/s	10.00	32.83
Left hip pitch	deg/s	10.00	32.83
Left achilles rod wx	deg/s	0.00	0.00
Left achilles rod wy	deg/s	0.00	0.00
Left achilles rod wz	deg/s	0.00	0.00
Left knee	deg/s	25.00	5.25
Left shin	deg/s	0.00	0.00
Left tarsus	deg/s	0.00	0.00
Left heel spring	deg/s	0.00	0.00
Left foot crank	deg/s	0.00	0.00
Left plantar rod	deg/s	0.00	0.00
Left foot	deg/s	7.50	58.36
Right hip roll	deg/s	10.00	32.83
Right hip yaw	deg/s	10.00	32.83
Right hip pitch	deg/s	10.00	32.83
Right achilles rod wx	deg/s	0.00	0.00
Right achilles rod wy	deg/s	0.00	0.00
Right achilles rod wz	deg/s	0.00	0.00
Right knee	deg/s	25.00	5.25
Right shin	deg/s	0.00	0.00
Right tarsus	deg/s	0.00	0.00
Right heel spring	deg/s	0.00	0.00
Right foot crank	deg/s	0.00	0.00
Right plantar rod	deg/s	0.00	0.00
Right foot	deg/s	7.50	58.36

TABLE 6.3: State Vector **x** for Keyframes in the Two-leg Case

State	**Key 0**	**Key 1**	**Key 2**	**Key 3**	**Key 4**
Pelvis x	0.0	-0.0026	-0.0034	0.0006	-0.0012
Pelvis y	0.0	0.00002	0.00004	-0.0027	0.0001
Pelvis z	1.0042	0.8983	0.8118	0.6627	0.9055
Pelvis orientation qw	1.0	0.9997	0.9988	0.9866	0.9950
Pelvis orientation qx	0.0	0.0016	0.0030	-0.0229	0.0023
Pelvis orientation qy	0.0	-0.0247	-0.0494	-0.1613	-0.0321
Pelvis orientation qz	0.0	-0.0006	-0.0010	0.0019	-0.0009
Left hip roll	0.0038	0.0010	-0.0013	0.0551	0.0028
Left hip yaw	-0.0011	-0.0002	0.0010	-0.0097	-0.0015
Left hip pitch	0.4264	0.5691	0.6335	0.5131	0.5770
Left achilles rod qw	0.9831	0.9331	0.8828	0.7845	0.9200
Left achilles rod qx	-0.0166	-0.0147	-0.0122	-0.0063	-0.0134
Left achilles rod qy	0.0151	0.0351	0.0501	0.0740	0.0410
Left achilles rod qz	-0.1819	-0.3575	-0.4669	-0.6157	-0.4520
Left knee	-1.1161	-1.5065	-1.7539	-2.1098	-1.7654
Left shin	-0.0404	-0.0198	-0.0160	-0.0261	-0.0187
Left tarsus	1.4460	1.7787	2.0137	2.4032	1.9500
Left heel spring	-0.0297	-0.0176	-0.0142	-0.0254	-0.0203
Left foot crank	-1.4793	-1.6299	-1.7350	-1.8715	-1.7820
Left plantar rod	1.4609	1.6115	1.7167	1.8530	1.7000
Left foot	-1.5874	-1.7424	-1.8475	-1.9772	-1.8650
Right hip roll	0.0038	0.0010	-0.0014	0.0540	0.0030
Right hip yaw	-0.0011	-0.0004	-0.0004	-0.0231	-0.0008
Right hip pitch	0.4264	0.5702	0.6355	0.5039	0.5900
Right achilles rod qw	0.9831	0.9327	0.8819	0.7929	0.9100
Right achilles rod qx	0.0166	0.0147	0.0122	0.0069	0.0138
Right achilles rod qy	-0.0151	-0.0353	-0.0503	-0.0722	-0.0385
Right achilles rod qz	-0.1819	-0.3585	-0.4685	-0.6050	-0.4600
Right knee	-1.1161	-1.5082	-1.7571	-2.0812	-1.7700
Right shin	-0.0404	-0.0204	-0.0167	-0.0272	-0.0190
Right tarsus	1.4460	1.7811	2.0177	2.3753	1.9600
Right heel spring	-0.0297	-0.0176	-0.0142	-0.0252	-0.0210
Right foot crank	-1.4793	-1.6308	-1.7370	-1.8611	-1.7805
Right plantar rod	1.4609	1.6124	1.7187	1.8426	1.7100
Right foot	-1.5874	-1.7434	-1.8496	-1.9669	-1.8700

TABLE 6.4: State Vector **x** for Keyframes in the One-leg Case

State	**Key 5**	**Key 6**	**Key 7**	**Key 8**	**Key 9**
Pelvis x	0.0403	0.0576	0.0403	0.0403	0.0403
Pelvis y	0.0011	0.0023	0.0011	0.0011	0.0011
Pelvis z	0.9213	0.9058	0.9213	0.9213	0.9213
Pelvis orientation qw	0.9975	0.9964	0.9975	0.9975	0.9975
Pelvis orientation qx	-0.0483	-0.0607	-0.0483	-0.0483	-0.0483
Pelvis orientation qy	0.0319	0.0401	0.0319	0.0319	0.0319
Pelvis orientation qz	0.0415	0.0436	0.0415	0.0415	0.0415
Left hip roll	-0.0240	-0.0240	-0.0240	-0.0240	-0.0240
Left hip yaw	-0.0217	-0.0217	-0.0217	-0.0217	-0.0217
Left hip pitch	1.0552	1.0552	1.0552	1.0552	1.0552
Left achilles rod qw	0.8862	0.8862	0.8862	0.8862	0.8862
Left achilles rod qx	-0.0120	-0.0120	-0.0120	-0.0120	-0.0120
Left achilles rod qy	0.0492	0.0492	0.0492	0.0492	0.0492
Left achilles rod qz	-0.4605	-0.4605	-0.4605	-0.4605	-0.4605
Left knee	-1.7548	-1.7548	-1.7548	-1.7548	-1.7548
Left shin	0.0033	0.0033	0.0033	0.0033	0.0033
Left tarsus	1.9648	1.9648	1.9648	1.9648	1.9648
Left heel spring	0.0011	0.0011	0.0011	0.0011	0.0011
Left foot crank	-1.6919	-1.6919	-1.6919	-1.6919	-1.6919
Left plantar rod	1.6735	1.6735	1.6735	1.6735	1.6735
Left foot	-1.7984	-1.7984	-1.7984	-1.7984	-1.7984
Right hip roll	0.2407	0.2675	0.2407	0.2407	0.2407
Right hip yaw	-0.0162	-0.0219	-0.0162	-0.0162	-0.0162
Right hip pitch	0.4995	0.5122	0.4995	0.4995	0.4995
Right achilles rod qw	0.9708	0.9667	0.9708	0.9708	0.9708
Right achilles rod qx	0.0039	0.0038	0.0039	0.0039	0.0039
Right achilles rod qy	-0.0175	-0.0190	-0.0175	-0.0175	-0.0175
Right achilles rod qz	-0.2393	-0.2552	-0.2393	-0.2393	-0.2393
Right knee	-1.1916	-1.2249	-1.1916	-1.1916	-1.1916
Right shin	-0.0892	-0.0896	-0.0892	-0.0892	-0.0892
Right tarsus	1.6468	1.6807	1.6468	1.6468	1.6468
Right heel spring	-0.0687	-0.0697	-0.0687	-0.0687	-0.0687
Right foot crank	-1.5612	-1.5561	-1.5612	-1.5612	-1.5612
Right plantar rod	1.5428	1.5376	1.5428	1.5428	1.5428
Right foot	-1.6688	-1.6637	-1.6688	-1.6688	-1.6688