Balancing and Hopping Motion Control Algorithms for an Under-actuated Robot

Morteza Azad

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

June 2014

Declaration

This thesis is the result of my own original work under the guidance and supervision of Doctor Roy Featherstone during the period of my PhD degree at the Australian National University. Most of the results in this thesis have been published in refereed journals and at international conferences. These results include:

1. M. Azad and R. Featherstone, Modeling the contact between a rolling sphere and a compliant ground plane. *Australasian Conference on Robotics and Automation (ACRA 2010)*, Brisbane, Australia, 1-3 December 2010.

2. M. Azad and R. Featherstone, Angular momentum based controller for balancing an inverted double pendulum. *19th CISM-IFToMM Symposium on Robot Design, Dynamics and Control (ROMANSY2012)*, Paris, France, 12-15 June 2012.

3. M. Azad and R. Featherstone, Balancing and hopping motion of a planar hopper with one actuator, *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 6-10 May 2013.

4. M. Azad and R. Featherstone, A new nonlinear model of contact normal force, accepted November 21, 2013 to be published in the *IEEE Transactions on Robotics*.

The idea of using a decomposed bend-swivel controller to control the 3D balancer robot, which is discussed in chapter 5, was proposed by Dr. Roy Featherstone. He also invented the idea of using a constant velocity joint in the knee joint of the 3D balancer to decouple its motion instantaneously into bend and swivel motions.

Morteza Azad
22 June 2014

to my wife and parents

Acknowledgments

This thesis would not have been possible without the support and assistance of many individuals and some organizations. I would like to thank all those who have made this thesis possible.

First and foremost, I would like to express my very great appreciation to my supervisor Dr. Roy Featehrstone. I am so proud to had the chance to work with Roy from whom I have learnt a lot. I am very thankful to him for his incredible support, bright ideas and great supervision during my candidature.

I would like to express my gratitude to Dr. Jochen Trumpf and Prof. Robert Mahony for their comments and support. In particular, I am very grateful to Jochen for spending hours to review all my thesis. Also I would like to thank the Australian National University and the School of Engineering for their financial support.

During my time as a student I had a chance to visit one of the best robotics laboratory in Italy. I would like to thank Dr. Roy Featherstone and Prof. Thushara Abhayapala, the director of the School of Engineering, who provided this visit for me. I would also like to thank Prof. Darwin Caldwell, Dr. Claudio Semini, Dr. Jonas Buchli and the students and the staff of the Department of Advanced Robotics at the Istituto Italiano di Tecnologia, Genova, Italy, who supported me with a very welcoming environment during the visit.

I would like to thank all my fellow students and the staff of the School of Engineering who provided me with a pleasant working environment at the Australian National University.

Last but not least, I would like to gratefully thank my family and friends who were kindly supporting me during my candidature. This thesis would not be possible without the continuous love and support of my family. My special thanks go to my wife, Sahba, who has been very supportive, patient and encouraging during these years.

Abstract

The main contribution of this thesis is a new and simple angular momentum based control algorithm for balancing an under-actuated planar robot. The proposed controller is able to stabilize the robot at any unstable balanced configuration and to control the robot to follow motion trajectory commands. This controller is also capable of balancing the robot on a rolling contact. A modified version of the controller is used to control the robot during a single hop motion. The proposed planar controller is then used as a part of a 3D balancing controller for a spatial under-actuated robot. The 3D control algorithm, which is based on the angular momentum of the robot, can balance the robot within any vertical plane and also rotate the robot from a vertical plane to any other one. Performance of the control algorithm in planar balancing, spatial balancing and planar hopping motions is demonstrated by simulation results. Although hopping motion of a spatial under-actuated robot is not considered in this thesis, it is shown in simulations that, starting from an upright configuration, the 3D controller is able to move the robot within any arbitrary vertical plane. Therefore, by confining the robot's motion to a vertical plane, the robot is potentially able to perform hopping motion in 3D using the same algorithm that is proposed for planar hopping motion.

Another major contribution of this thesis is a new nonlinear model for the contact normal force. This new model accurately predicts the measured values of the coefficient of restitution between spheres and plates of various materials. The new contact model is used in this thesis to model the contact between the robot's foot and the ground during a hopping motion.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Balancing is always a crucial problem in legged robots. This problem is much more challenging when it comes to monopod robots which have only one leg. It is also an indispensable and the most important part of the hopping motion of a monopod robot. This thesis concerns the problems of balancing and hopping motions of a planar and a spatial under-actuated monopod robots with a minimum possible ratio between the number of actuators and degrees of freedom (DoF).

1.1 Background

The robot that is considered in this study is in fact a monopod robot which is a kind of legged robots with only one leg. Although legged robots are much more difficult to control, rather than wheeled ones, they are more applicable in real environments. They are usually capable of moving in rough terrains much better than wheeled ones and also have more mobility and flexibility in moving. Having legs instead of wheels, allows them to move in environments with discontinuous support like a flight of stairs or rungs of a ladder.

In general, walking and running are the two main types of locomotion for legged robots. The difference is that during walking there is always at least one leg, supporting the robot's body, in contact with the ground whereas running includes a flight phase in which the robot is not in contact with the ground. Hopping is essentially a special case of running. Hopping robots have only one leg or if they have more than one leg then all legs do the same thing at the same time (like a kangaroo).

Hopping motion of a monopod can be simply generalised to running motion of a biped or a multi-legged robot. Comparing with walking robots, hopping and running robots usually have simpler dynamic models and control algorithms. Also, studying hopping motion can be regarded as a basic study on legged robots. So the literature is reviewed with focus on hopping robot mechanisms and their control algorithms. Also balancing control algorithms of under-actuated robots are considered because of the necessity of balancing the robot as a part of its hopping motion.

1.1.1 Hopping Robots

Raibert was a pioneer researcher in the field of hopping robots. More than twenty years ago, he and his colleagues at CMU proposed a control algorithm for a planar hopping robot [Raibert and Brown Jr, 1984] that amazed robotics and control community because of its simplicity. They verified their proposed control algorithm by implementing it on a real monopod planar hopper. The robot showed good performance at hopping in place, hopping at various speeds, and leaping over small obstacles. Fig. 1.1 shows a schematic diagram of the hopper which is built by Raibert and his colleagues. Raibert's planar hopper consists of a torso and a leg connected to each other by a hinge-type hip joint. The leg, which is in fact an air cylinder, acts like a spring (springy leg) and the torso carries sensors, valves and actuators [Raibert, 1986]. This hopper, which has five motion DoF, is controlled by using two actuators one in the hip joint and the other in the leg.

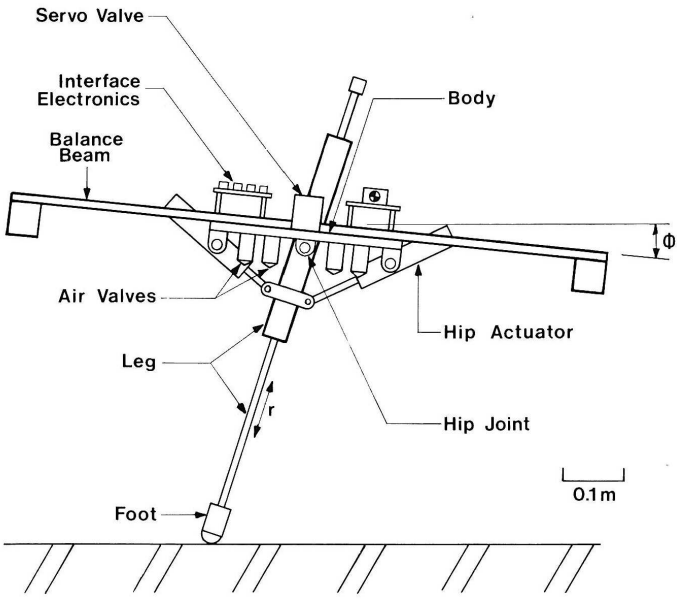


Figure 1.1: Raibert's 2D hopper (reproduction of Figure 2.1 in [Raibert, 1986])

Raibert claimed that control of such a monopod hopping robot can be decomposed into three separate parts, which are controlling the hopping height, regulating the forward speed and correcting the attitude of the torso. Raibert further developed his control algorithm and successfully controlled the running motion of a biped robot [Raibert, 1986].

Raibert then extended his planar control algorithm to 3D and built a spatial monopod robot and controlled its hopping motion [Raibert et al., 1984]. Fig. 1.2 shows a schematic diagram of Raibert's spatial hopper. Like the planar version, it has a torso (carrying sensors, valves, etc.) and a leg (pneumatic cylinder). This mechanism has nine DoF, and is controlled by using three actuators (two in the hip joint and one in the leg).

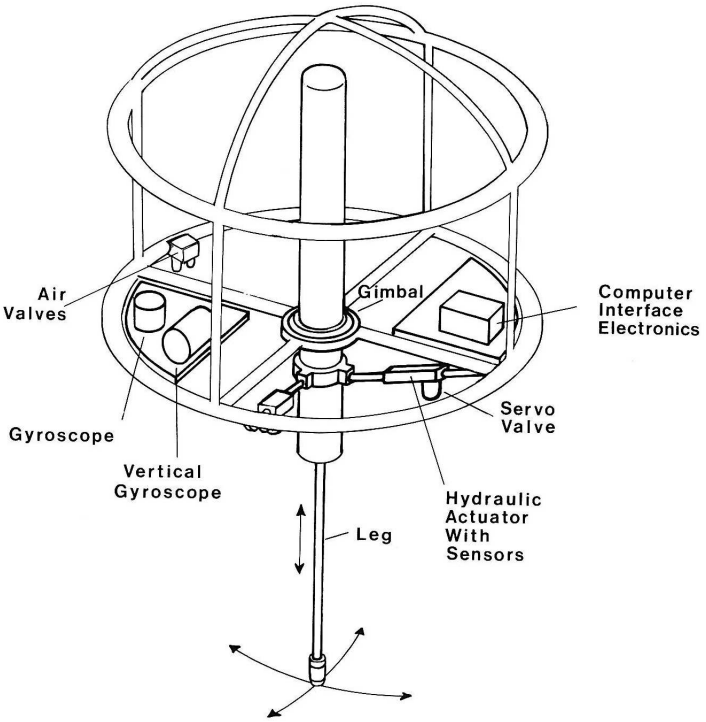


Figure 1.2: Raibert's 3D hopper (reproduction of Figure 3.5 in [Raibert, 1986])

Raibert also studied running on four legs [Raibert et al., 1986], gymnastics with a biped robot [Hodgins and Raibert, 1990] and somersault of a biped in 3D [Playter and Raibert, 1992].

1.1.1.1 Classifications of Hopping Robot Mechanisms

After Raibert, many researchers studied dynamics and control of hopping and running robots using different kinds of mechanisms which are reviewed in [Sayyad et al., 2007]. Fig. 1.3 shows a classification of these mechanisms based on their dimensions and leg mechanisms. In general, a planar or a spatial hopping (or running) robot consists of a torso and a leg (or two or more legs). The leg could be either a Raibert-style leg (i.e. a link containing a prismatic joint which is connected to the torso by a revolute joint) or a leg with a knee (i.e. two links connected to each other by an actuated revolute joint). These kinds of legs are referred as "prismatic" and "knee" legs in this chapter, respectively.

For 1D mechanisms there is only one possible motion which is hopping at a place. Some researchers looked into this type of mechanism to study vertical motion of the robot and control its hopping height [Li and He, 1990; Vakakis et al., 1991; Michalska et al., 1996; Harbick and Sukhatme, 2001a; Fernandes et al., 2009].

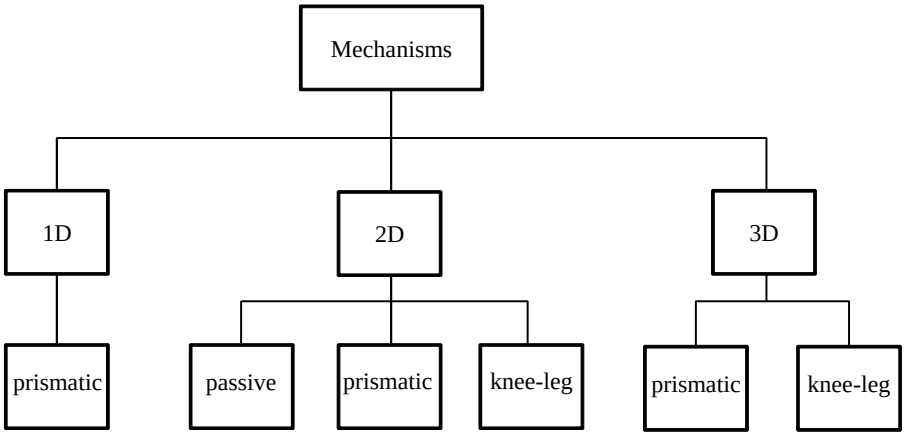


Figure 1.3: Classifications of hopping and running robot's mechanisms

1.1.1.2 Planar Hoppers

Most of the studies on hopping robots are focused on planar robots. A few of them have considered passive prismatic leg hopping robots. These robots have either passive prismatic and revolute joints by using passive springs [Ahmadi and Buehler, 1995, 1997], or a passive prismatic joint and a compliant revolute joint [Hyon and Emura, 2004b; Ahmadi and Buehler, 2006].

Some of the studies, in the category of prismatic leg mechanisms, include designing and controlling a hopping machine [Rad et al., 1993], proposing a periodic hopping motion controller [M'Closkey and Burdick, 1991, 1993], introducing a new energy efficient controller for the hopping motion [François and Samson, 1998], studying flight phase control [Li and Montgomery, 1990; Ur-Rehman, 2005], forward velocity control [Schwind and Koditschek, 1995], forward velocity control on rough terrain [Cherouvim and Papadopoulos, 2009] and height control [Harbick and Sukhatme, 2001b], analyzing stability of different controllers [Altendorfer et al., 2004; Ankaralı and Saranlı, 2010] and proposing new torso mechanisms on a prismatic leg mechanism [Iida et al., 2002; He et al., 2008]. Also there have been a few studies on controlling running motions of planar prismatic biped robots [Raibert, 1986; Hyon and Emura, 2004a; Abdallah and Waldron, 2007, 2009].

Comparing with prismatic leg hoppers, knee-leg ones have more complicated dynamics and therefore they are more difficult to control. However, their similarities to human legs and also their simple mechanisms (to design and build) have encouraged some researchers to work on this type of hopping robot [Vermeulen et al., 2003; Fujimoto, 2004; Takahashi et al., 2006; Ohashi and Ohnishi, 2006; Sung and Youm, 2007; He and Geng, 2009]. Studying hopping motion of knee-leg robots can also help better understanding of human-like motion which might be useful to make faster and more agile humanoid or legged robots.

The most advanced study on knee-leg hopping machines is recently accomplished by Grizzle and his colleagues at the University of Michigan and Carnegie Mellon

University [Poulakakis and Grizzle, 2007, 2009b,a]. They proposed a hybrid zero dynamics controller for hopping motion of a knee-leg robot. They also designed and built a robot, which is called "Thumper", and experimentally verified their proposed control algorithm. Then they extended the controller to control running motion of a biped knee-leg robot [Grizzle et al., 2009]. They built a biped version of thumper, which is named "Mabel", and successfully implemented the running control algorithm on the robot. Running with biped knee-leg robots have been studied by some other researchers as well [Chevallereau and Aoustin, 2001; Chevallereau et al., 2005; Morris et al., 2006; Shimizu et al., 2006; Amagata et al., 2008].

Among the studies on planar knee-leg hoppers, the work which is done by Berkemeier and Fearing [1998] is fundamentally different from other ones in this category in the sense of the ratio between the number of actuators and DoF of the robot. Berkemeier and Fearing [1998] introduced a four DoF hopping mechanism, which was in fact a knee-leg mechanism without a torso, and proposed a control algorithm for its sliding and hopping motions using only one actuator in the knee joint.

A schematic diagram of the hopping robot mechanism that they used in their studies is shown in Fig. 1.4. Their proposed mechanism consists of two links which are connected to each other via an actuated revolute joint. This joint is referred as the knee joint in rest of this thesis. The actuated joint could also be considered as a hip joint connecting a leg to a torso. The tip of the lower leg acts as a foot and it is the only point that makes contact with the ground. Configurations of the robot is characterized by its generalized coordinates which are x_1 and y_1 (horizontal and vertical coordinates of the foot), θ_1 (angle between the lower link and vertical) and θ_2 (angle between the links).

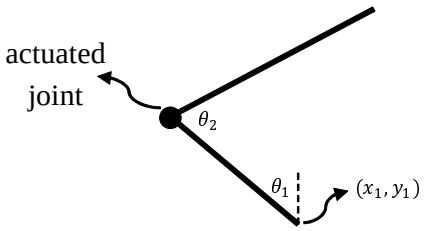


Figure 1.4: Berkemeier's planar hopper with one actuator

Berkemeier and Fearing divided the hopping control problem into two parts: stance phase control and flight phase control. In the stance phase, if the robot is not sliding on the ground, the dynamics of the hopper are identical to those of a two DoF inverted double pendulum robot with one degree of under-actuation. To control the robot in the stance phase, they designed a controller to balance the robot [Berkemeier and Fearing, 1999] which was able to tolerate a little sliding (it presents small disturbances to the system) and maintain its balance while it accelerates its center of mass (CoM) upward. They used this controller to oscillate the robot to obtain enough energy for taking off the ground by tracking a prescribed trajectory during stance phase. In the flight phase, they found a desired trajectory for the

hop and derived a controller to track that trajectory during the flight. Their control strategy in the flight phase was to rotate the lower leg, an integral number of times, to enable the robot to land in the same configuration as it took off the ground. They were able to show a good performance of sliding and hopping motions of the robot in simulation. Fig 1.5 shows the frames of their robot's hopping gait.

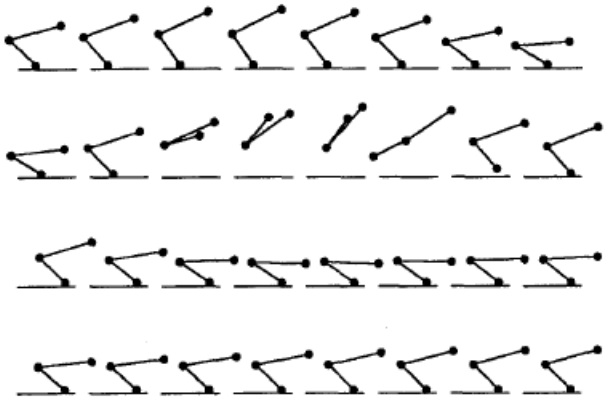


Figure 1.5: Frames of a hopping gait of Berkemeier's hopping robot (reproduction of Figure 8 in Berkemeier and Fearing [1992])

After Berkemeier and Fearing's work, there have been a number of studies on posture control of the same robot during its flying motion [Mita et al., 2001; Hattori et al., 2004; Xin et al., 2004]. The most remarkable work among them is done by Grizzle et al. [2005]. They introduced a nonlinear control algorithm for a general case of under-actuated planar mechanical systems with one degree of under-actuation during stance and flight phases. They showed ballistic motion of a knee-leg hopper with initial angular momentum in simulation.

1.1.1.3 Spatial Hoppers

After Raibert's 3D hopper [Raibert et al., 1984; Raibert, 1986; Playter and Raibert, 1992], there have been only a few studies on 3D hoppers with a prismatic leg mechanism [Seipel, 2005; Seipel and Holmes, 2005a,b, 2006; Wu and Geyer, 2013].

In the category of spatial knee-leg robots, there are several of them that can hop, run or walk. They usually have many DoF and also many actuators [Kajita et al., 2007b,a; Kawamura and Zhu, 2006; Park and Kwon, 2003; Cho and Oh, 2008, 2009]. The problem with this kind of robots is that they are clumsy, slow, heavy and inefficient in the sense of energy consumption.

Unfortunately, there has not been any study on knee-leg 3D hopping robots with a minimum possible number of actuators to DoF ratio. In fact, inspired by the work of Berkemeier and Fearing [1998] (and also Raibert's 3D hopper), this thesis is aiming to propose a solution for the hopping motion of such robots. Such a 3D knee-leg hopping robot would have eight DoF with only two actuators. The lower leg of the robot would have full six DoF in the space and the upper leg would be attached to

the lower one by a two-DoF actuated joint. This thesis introduces a control algorithm which is able to balance the robot and more importantly to confine its motion to an arbitrary vertical plane. Therefore, the problem of its hopping motion in 3D will be simplified to the planar hopping motion which is well-studied in this thesis.

1.1.2 Balancing Robots

Studying balancing motion control is an unavoidable part of a study on hopping motion specially in the case of knee-leg hopping robots. The knee-leg hopper that is studied in this thesis, which is the same as the one that is proposed by Berkemeier and Fearing [1998], is in fact an inverted double pendulum robot. Balancing control problem of such a robot with one degree of under-actuation is first considered by Hauser and Murray [1990]. They proposed a non-linear controller for balancing the acrobot (acrobatic robot). The acrobot is an inverted double pendulum robot which consists of two bodies connected to each other by an actuated revolute joint (knee joint). The lower body is attached to the ground by a passive revolute joint. Balancing of the acrobot has been studied by some other researchers as well [Olfati-Saber and Megretski, 1998; Berkemeier and Fearing, 1999; Olfati-Saber, 2000, 2002; Yamakita et al., 2002].

Balancing motion of the acrobot is also investigated by some researchers as a part of its swinging-up motion control. The swinging-up controller moves the robot from its downward stable configuration via a series of swings and makes it balance at its upward unstable configuration. Some researchers linearized the system about its upright balanced configuration and used an LQR controller for the balancing part [Spong, 1994, 1995; Brown and Passino, 1997; Xin and Kaneda, 2001; Mahindrakar and Banavar, 2005; Lai et al., 2005; Inoue et al., 2007]. However, some others used a different approach for the balancing part of the swinging-up motion controller. Nam et al. [2002]; Yonemura and Yamakita [2004] proposed an output zeroing controller for balancing the acrobot. Their proposed controller uses an output function which is defined by the angular momentum and one other new state. To work out the value of the torque, they had to calculate the third derivative of the angular momentum. This approach has been used by Okawa et al. [2009]; Kawaguchi and Yamakita [2011] to balance a bicycle or a bike robot on its rear wheel.

Berkemeier and Fearing [1999] introduced a controller based on zero dynamics for trajectory tracking of the acrobot. They found a class of interesting feasible trajectories for the acrobot and achieved theoretically accurate trajectory tracking performance using their proposed controller. They also used a simple linear feedback controller to balance the robot in its upright configuration.

Grizzle et al. [2005] considered the general case and designed a nonlinear controller for mechanical systems with one degree of under-actuation. Their output function becomes identical to the one that mentioned in [Olfati-Saber and Megretski, 1998; Olfati-Saber, 2000, 2002; Nam et al., 2002; Yamakita et al., 2002; Yonemura and Yamakita, 2004] for the acrobot case. This approach also needs to work out the third derivative of the angular momentum.

1.2 Thesis Outline

The thesis consists of six chapters including the first chapter (i.e. introduction).

Chapter 2 considers the contact between a sphere and the ground and proposes a new non-linear model to calculate the contact force. The contact model consists of a normal force model and a friction force model. The normal force model differs from previous models in the literature by a single term (the damping term). This small difference in the model results in a substantial difference in its performance compared to previous models. Simulation results show that only by using the new normal force model (not the previous ones), the calculated coefficient of restitution between spheres and plates fits the empirical data for a variety of materials.

Chapter 3 studies the balancing control problem of a planar under-actuated robot. The robot is in fact an inverted double-pendulum robot with one degree of under-actuation. The stability analysis of the controller shows that it is able to balance the robot in any unstable but controllable balanced configuration. Also it is shown in simulation that the proposed controller can follow motion trajectory commands with reasonable accuracy. Simulation results show that imperfections in the system do not substantially affect the performance of the controller. The last section of this chapter investigates the balancing motion of the planar robot with rolling contact using the same controller.

Chapter 4 introduces a control strategy for a single-hop motion of a planar under-actuated robot. The robot is the same as the one that is studied in chapter 3. During a single-hop motion, the robot starts in its upright balanced configuration and ends up in the same configuration at the end of the hop. This hopping motion includes four phases which are crouch-and-launch, flight, landing and rebalancing. The proposed balancing controller in chapter 3 is used to rebalance the robot in the last phase. A modified version of this controller is used to track reference trajectories during the crouch-and-launch and flight phases.

In chapter 5, an angular momentum based controller is introduced to perform balancing motion of a spatial under-actuated robot. The robot is essentially a two-link robot with five DoF and three degrees of under-actuation. The links are connected to each other by a two-DoF actuated joint. The actuated joint is designed to be a constant velocity joint which decouples the robot's motion instantaneously into bending and swivelling motions. The proposed controller controls each of these motions separately. Simulation results show that the controller is able to stabilize the robot in its upright balanced configuration as well as any other balanced configuration in any arbitrary vertical plane. It is also able to change the orientation of the robot by rotating the vertical plane of the robot about the vertical axis.

The last chapter includes the concluding remarks and outlines some possible extensions and directions for future research.

Chapter 2

Modelling the Contact between a Sphere and the Ground

Since robots usually make contact with their environment during the execution of their tasks (e.g. grasping, walking, rolling, etc.) modelling this contact is an indispensable part of most studies in this field. In particular, in the case of legged robots, modelling the contact between the robot's feet and the ground is one of the most important parts of the study. Two major forces appear during contact: the normal force and the friction force. It is desirable to have a complete model for the contact which is able to model both normal and friction forces.

In this chapter, a new non-linear normal force model is derived which conforms to Hertz's theory [Johnson, 1977] for contact between a sphere and a plate. The new normal force model differs from a well-known existing model by only a single term. The advantage of the new model is that it accurately predicts the measured values of the coefficient of restitution between spheres and plates of various materials, whereas other models do not.

In section 2.2 a new non-linear 2D model for friction force is introduced which computes the pre-sliding movement by a non-linear equation. Combining these two new models for normal and friction forces, gives a complete 3D model for the contact between a sphere and a plate. A 2D version of this contact model is used later in chapter 4 for simulating the contact between a robot's foot and the ground.

Finally, the rolling motion of a sphere on a compliant ground plane is simulated, and the results are presented. Rolling motion is an integral part of simulating the general 3D motion of a robot's foot while in contact with the ground.

Most of the results presented in this chapter are published in [Azad and Featherstone, 2010].

2.1 Contact Normal Force

In general, contact normal force models can be classified into two types: rigid and compliant [Gilardi and Sharf, 2002]. Rigid models assume that both contacting surfaces remain fully rigid during contact whereas compliant ones assume that at least one of the bodies is compliant and therefore deforms locally at the contact area.

Given this assumption, the normal force of a compliant model can be expressed as a function of the local deformation and its rate of change. This chapter presents a new compliant model to be used in dynamics simulators to estimate the contact normal force.

Most previous studies of compliant contact models have considered the contact between two spheres, or between a sphere and a flat plate [Hunt and Crossley, 1975; Lankarani and Nikravesh, 1990; Falcon et al., 1998; Marhefka and Orin, 1999]. For example, Hunt and Crossley [1975] modelled the ground as a nonlinear spring-damper pair at the contact point with the sphere, and introduced a nonlinear equation for the normal force between a sphere and the ground as

$$F = \kappa z^n + \lambda z^p \dot{z}^q \,, \tag{2.1}$$

where z is the deformation variable, which is defined as the penetration distance of the undeformed sphere into the undeformed ground, $\dot{z}$ is the rate of deformation, κ and λ are the coefficients of the spring and damper, respectively, and n, p and q are constant parameters. They chose the values of these parameters as $n = \frac{3}{2}$ (to get similar results to Hertz's theory), and $p = \frac{3}{2}$ and $q = 1$ (to be able to determine the value of λ with respect to κ conveniently), resulting in the equation

$$F = \kappa z^{\frac{3}{2}} + \lambda z^{\frac{3}{2}} \dot{z} \,. \tag{2.2}$$

This model also has been studied by Lankarani and Nikravesh [1990] and Marhefka and Orin [1999], and has become well known in the robotics community. This model is referred to as the Hunt/Crossley model in this chapter.

According to the literature, the coefficient of restitution of a contact is an experimental criterion to verify a normal force model and test its accuracy. There are some studies that report experimentally measured values of the coefficient of restitution between spheres and plates of various materials [Goldsmith, 1960; Kawabara and Kono, 1987]. In the remainder of this section, the new model is derived and its accuracy is tested against the experimental data. Using the new model and the Hunt/Crossley model, respectively, and comparing the simulation results to the empirical data shows that the new model is accurate whereas the Hunt/Crossley model is not.

2.1.1 The New Model

The contact normal force in (2.1) is the sum of a non-linear elastic component and a non-linear damping component. The elastic term $\kappa z^{\frac{3}{2}}$ that is proposed by Hunt and Crossley [1975] in (2.2) is consistent with Hertz's theory for contact between a sphere and a plate, and is known to be correct. However, there is not any strong theoretical or experimental background for the damping term in (2.2). In this chapter a different approach is employed to model the contact between a sphere and a plate and it results in a new model that agrees with Hertz's theory and differs from (2.2) only in the value of p, which is shown to be $p = \frac{1}{2}$.

For modelling the contact between a sphere and a plate, a simplifying assumption is made that one material is much harder than the other, so that substantially all of the compression takes place in the softer body. Thus, it is considered that the harder body is rigid and the softer body contains a uniform distribution of infinitely many nonlinear spring-damper pairs. Therefore the contact normal force is the resultant of the spring and damper forces

$$F_n = F_K + F_D\,, \tag{2.3}$$

where F_K and F_D are the forces due to the springs and dampers, respectively. Let k and b denote the stiffness and damping coefficients of each individual spring-damper pair, respectively. Both k and b are functions of local deformation. So for each spring-damper pair, that is engaged in the contact, the spring force is

$$f_{K_i} = \int_0^{\delta_i} k\,\mathrm{d}\xi \tag{2.4}$$

and the damper force is

$$f_{D_i} = b\dot{\delta}_i\,, \tag{2.5}$$

where δ_i and $\dot{\delta}_i$ are the deformation and the rate of the deformation of the i^{th} spring-damper pair, respectively. Consequently, total spring and damper forces would be

$$F_K = \sum_{i=1}^{\infty} f_{K_i} = \sum_{i=1}^{\infty} \int_0^{\delta_i} k\,\mathrm{d}\xi = \int_A \int_0^{\delta_i} k\,\mathrm{d}\xi\,\mathrm{d}A \tag{2.6}$$

and

$$F_D = \sum_{i=1}^{\infty} f_{D_i} = \sum_{i=1}^{\infty} b\dot{\delta}_i = \int_A b\dot{\delta}_i \mathrm{d}A\,, \tag{2.7}$$

where A is the surface of the contact area. Let R and l denote the radius of the sphere and the radius of the contact area, respectively (see Fig. 2.1). Because of the

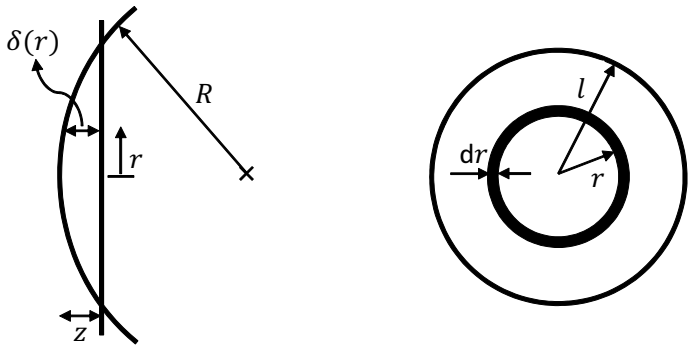


Figure 2.1: Contact between a sphere and a plate (a) local deformation, (b) contact area

symmetry of the deformations about the center of the contact area, all spring-damper

pairs at a distance r of the center have the same local deformation which is denoted by $\delta(r)$ (with $\delta(0) = z$). By defining $\mathrm{d}A = 2\pi r \mathrm{d}r$ as the surface of a ring element with inner radius r and outer radius $r + \mathrm{d}r$ that is centered around the center of the contact area (see Fig. 2.1(b)), F_K and F_D are

$$F_K = 2\pi \int_0^l r(\int_0^{\delta(r)} k \mathrm{d}\xi)\mathrm{d}r \tag{2.8}$$

and

$$F_D = 2\pi \int_0^l r b \dot{\delta}(r)\mathrm{d}r\,. \tag{2.9}$$

As can be seen from Fig. 2.1(a), $\delta(r)$ can be calculated as

$$\delta(r) = -(R - z) + \sqrt{R^2 - r^2}\,. \tag{2.10}$$

By using binomial approximations and assuming that r is very small and negligible with respect to R then (2.10) simplifies to

$$\delta(r) = z - \frac{r^2}{2R}\,. \tag{2.11}$$

To conform with Hertz's theory, k is chosen to be a function of $\zeta^{-\frac{1}{2}}$ in the form of $k(\zeta) = \beta\,\zeta^{-\frac{1}{2}}$, where ζ is the local deformation of each individual spring-damper pair and β is a constant parameter depending on the mechanical properties of the materials and the radius of the sphere. Assuming the same function for b (i.e. $b(\zeta) = \alpha\,\zeta^{-\frac{1}{2}}$), the spring and damper forces are

$$F_K = 4\pi\beta \int_0^l r\delta^{\frac{1}{2}}(r)\,\mathrm{d}r \tag{2.12}$$

and

$$F_D = 2\pi\alpha \int_0^l r\delta^{-\frac{1}{2}}(r)\dot{\delta}(r)\,\mathrm{d}r\,. \tag{2.13}$$

From (2.11), $\dot{\delta}(r) = \dot{\delta}(0) = \dot{z}$. Substituting (2.11) into (2.12) and (2.13), yields

$$F_K = \frac{8}{3}\pi\beta R z^{\frac{3}{2}} = K_n z^{\frac{3}{2}} \tag{2.14}$$

and

$$F_D = 4\pi\alpha R z^{\frac{1}{2}}\dot{z} = D_n z^{\frac{1}{2}}\dot{z}\,, \tag{2.15}$$

where K_n and D_n are the coefficients of total stiffness and total damping, respectively. So the total normal contact force can be expressed as

$$F_n = K_n z^{\frac{3}{2}} + D_n z^{\frac{1}{2}}\dot{z}\,, \tag{2.16}$$

which is different from the Hunt/Crossley model in the power of z in the damping

term. Comparing with Hertz's theory,

$$K_n = \frac{4}{3}E^*\sqrt{R} \implies \beta = \frac{E^*}{2\pi\sqrt{R}}, \tag{2.17}$$

where E^* is determined by

$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}, \tag{2.18}$$

and E_1 and E_2 are the moduli of elasticity, and ν_1 and ν_2 are the Poisson's ratios of the two contacting surfaces [Johnson, 1977].

As was mentioned at the beginning of this section, the only simplifying assumption that has been made in this derivation is that one material is much harder than the other, so that substantially all of the deformation occurs in the softer body. This assumption is theoretically essential for (2.16) to be physically valid. However, comparing the simulation results and the empirical data, in the cases where this assumption does not hold, shows that the new model gives good results and is in particular more accurate than the Hunt/Crossley model.

2.1.1.1 Equivalent Radius

Now a compliant contact between two spheres having radii R_1 and R_2 is considered (see Fig. 2.2). Let $\delta'(r)$ denote the local deformation of the contact area at a distance

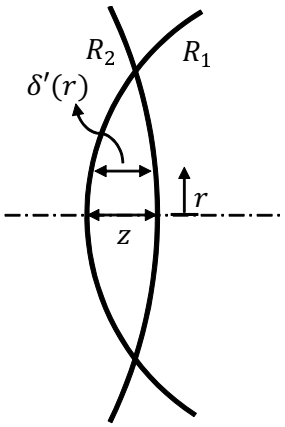


Figure 2.2: Contact between two spheres

r from the center, then

$$\delta'(r) = -(R_1 + R_2 - z) + \sqrt{R_1^2 - r^2} + \sqrt{R_2^2 - r^2}. \tag{2.19}$$

Again by assuming that r is very small and negligible compared to both radii, we have

$$\delta'(r) = z - \frac{r^2}{2}(\frac{1}{R_1} + \frac{1}{R_2}) = z - \frac{r^2}{2R}\,, \tag{2.20}$$

where

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}\,. \tag{2.21}$$

Therefore R is called the equivalent radius of the contact. By comparing (2.20) with (2.11), it is evident that the local deformation in the contact area (δ') is the same as the local deformation in a compliant contact between a sphere of radius R and a flat plate (δ).

Thus, the contact normal force model in (2.16) can be used to calculate the normal force in a sphere-plate contact as well as in a sphere-sphere contact. In the latter case, the equivalent radius (2.17) should be used.

2.1.1.2 Implementation Algorithm

According to (2.16), K_n and D_n are the two parameters of the new contact normal force model. The coefficient of total stiffness (K_n), which is a function of the mechanical properties of the contacting bodies and the radius of the contact, can be calculated using (2.17). The coefficient of total damping (D_n) needs to be estimated using the coefficient of restitution of the contact (see section 2.1.2). The new model also needs z and $\dot{z}$ as inputs to compute the normal force. Obviously, these values can be calculated from the relative position and velocity of the two bodies making the contact. The value of z will be negative whenever the undeformed shapes of these two bodies do not touch (see Fig. 2.2).

To calculate the normal force, using z and $\dot{z}$ as inputs, the simplest way is to use the following equation:

$$F_n = \begin{cases} 0 & \text{if } z \leq 0 \\ \max(0, \sqrt{z}\,(K_n z + D_n \dot{z})) & \text{if } z > 0 \end{cases} \tag{2.22}$$

where F_n is the output. So if $F_n > 0$ then the bodies are in contact and the friction force needs to be computed in the next stage (which will be described later in this chapter) and otherwise (i.e. if $F_n = 0$) there is no contact and the contact force is zero.

Equation (2.22) correctly predicts that contact will be lost before z drops to zero, because of the negative value of $D_n\dot{z}$ when the two bodies are moving apart, but this introduces one error into the model (see Fig. 2.3): if the sphere bounces twice in quick succession, so that the ground has not fully recovered from the first bounce at the time of the second landing, then (2.22) fails to model the second landing and bounce correctly. Physically, this happens on the last bounce before the two bodies settle into continuous contact.

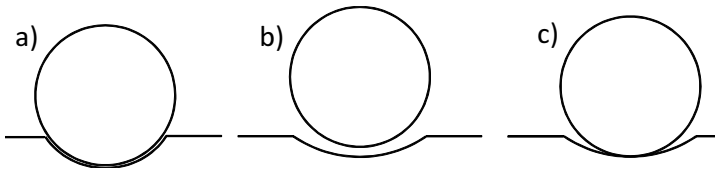


Figure 2.3: a) end of first bounce b) ground is recovering c) re-collision with the ground with undetermined shape.

2.1.2 Coefficient of Restitution

In this subsection, the coefficient of restitution is calculated using different contact models and the results are compared with each other and with published experimental results. Accurate dynamics simulations implemented in Simulink are employed to model the contact and compute the coefficient of restitution.

According to experiments reported by Goldsmith [1960] on measuring the coefficient of restitution for impacts between spheres and thick plates, the value should decrease as the impact velocity increases. Fig. 2.4 shows the values of the coefficient of restitution at different impact velocities, as calculated by the new model in (2.16), the Hunt/Crossley model in (2.2), and a linear spring-damper model ((2.1) with $n = q = 1$ and $p = 0$). In every case, gravity is set to zero. (With nonzero gravity, impacts below a threshold velocity do not result in a bounce, implying a coefficient of restitution equal to zero.) As can be seen, the linear model predicts a constant coefficient of restitution, which is not physically realistic. The two nonlinear models each correctly predict a decrease in the coefficient of restitution with increasing impact velocity, but they predict curves with different shapes. The difference between these curves can be observed more clearly by changing the scale of the horizontal axis to a logarithmic scale, as shown in Fig. 2.5. In this figure different curves are obtained by using different values of λ for the Hunt/Crossley model and D_n for the new model. The curves related to the Hunt/Crossley model in Fig. 2.5 are essentially the same as some of the curves in Fig. 2 in [Marhefka and Orin, 1999] only with a change in the scale of the horizontal axis. It can now be seen that the new model predicts an almost linear relationship between the coefficient of restitution and the logarithm of impact velocity, whereas the Hunt/Crossley model does not.

Figure 2.6 compares the calculated values of the coefficient of restitution according to the new model with empirical results published in [Goldsmith, 1960; Kawabara and Kono, 1987], all on a logarithmic scale. The empirical data were obtained by hand measurements from Fig. 172 in [Goldsmith, 1960] and Fig. 3 in [Kawabara and Kono, 1987]. From this graph it can immediately be seen that the data follow approximately straight lines, which means that the new model predicts the correct relationship between the coefficient of restitution and impact velocity, whereas the Hunt/Crossley model does not. Furthermore, in cases (A), (C), (D) and (F) the new model fits the data very well.

Best possible fits of the Hunt/Crossley model to the experimental data have been added to the graph in Fig. 2.7. The least-squares method is utilized to obtain the

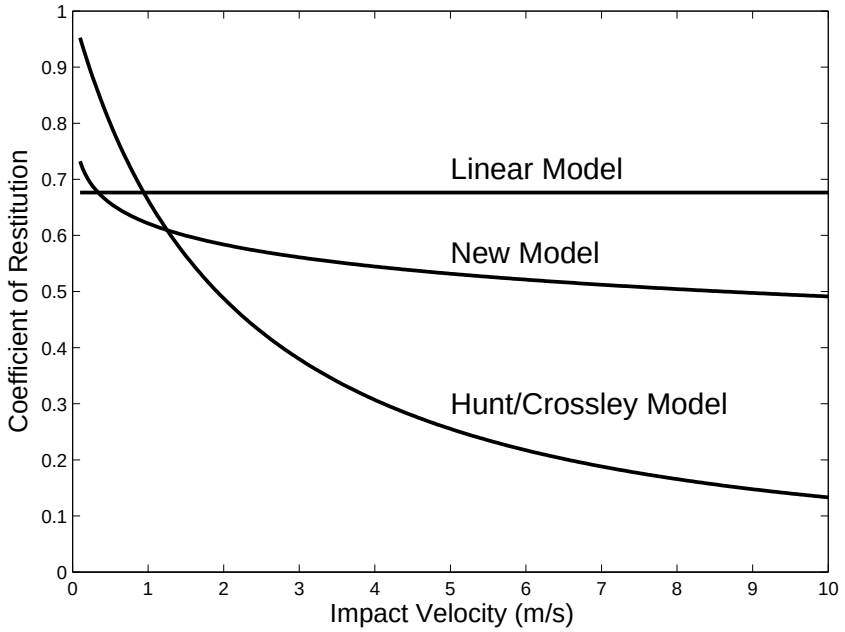


Figure 2.4: Coefficient of restitution vs. impact velocity calculated by three different models

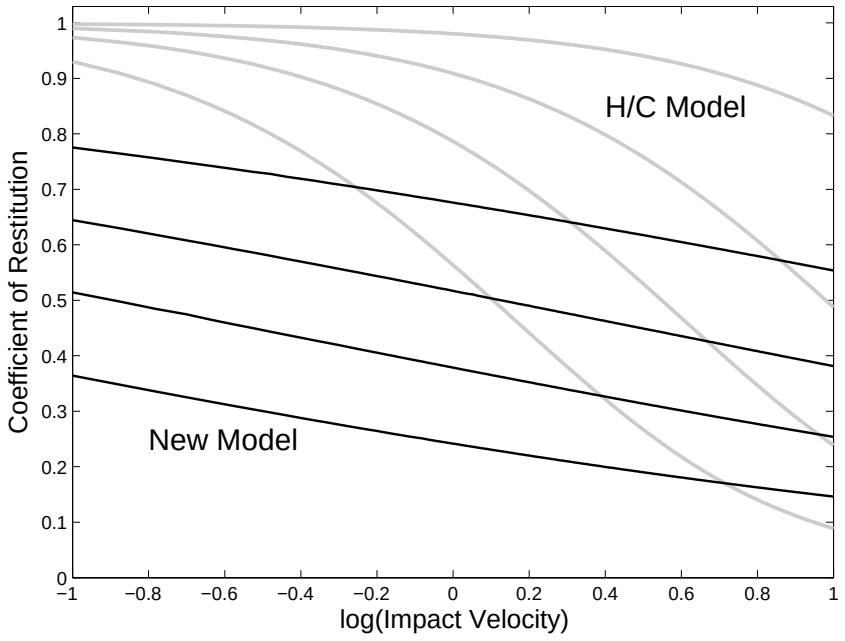


Figure 2.5: Coefficient of restitution vs. logarithm of impact velocity calculated by the Hunt/Crossley model and the new model for a range of model parameters

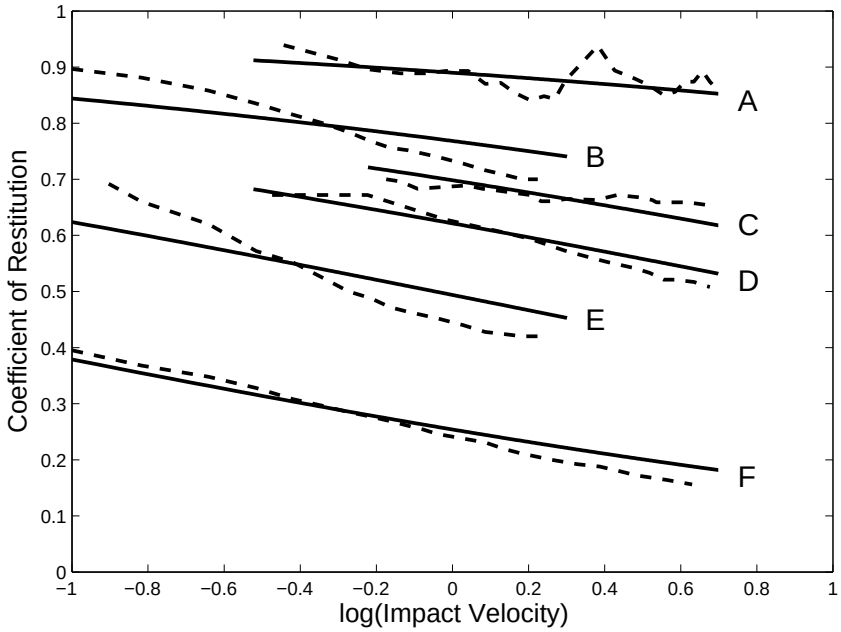


Figure 2.6: Coefficient of restitution vs. logarithm of impact velocity. Dashed curves show the experimental results and solid curves the calculated values using the new model. Results are for the contact between (A) brass spheres ($R_1 = R_2 = 1.5$cm), (B) a steel sphere ($R = 1.27$cm) and a cast iron plate, (C) cork spheres ($R_1 = R_2 = 1.66$cm), (D) a steel sphere ($R = 1.65$cm) and a cork plate, (E) a steel sphere ($R = 1.27$cm) and a brass plate, and (F) a steel sphere ($R = 1.27$cm) and a cold-worked lead plate.

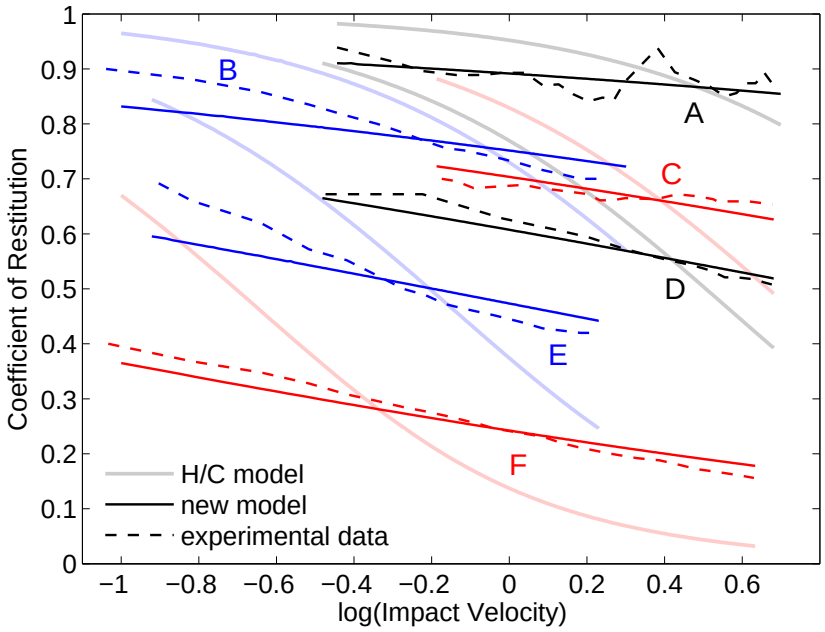


Figure 2.7: Coefficient of restitution vs. logarithm of impact velocity. Cases (A) to (F) are described in Fig. 2.6.

curves of the new model and the Hunt/Crossley model. According to this figure, it is clear that in all cases the new model fits the data much better than the Hunt/Crossley model does, although the fit is not so good in cases (B) and (E). The reason for the less accurate fit could be that in these two cases the sphere and the plate are made of different materials having similar stiffnesses, so the assumption in subsection 2.1.1 does not hold.

In obtaining the fits shown in Figs. 2.6 and 2.7, each K_n has been calculated from the appropriate material properties and the sphere radius using (2.17), and only D_n has been adjusted to fit the data. So in each case, there is only one parameter that needs to be tuned to acquire the proper curve.

2.2 Friction Force Model

There is a large number of research articles on modelling the friction force, such as Dupont et al. [2000]; Haessig Jr and Friedland [1991]; Bliman and Sorine [1995]; Dupont et al. [2002]; Dahl [1968]; Canudas de Wit et al. [1995], most of which are reviewed by Olsson et al. [1998]. In fact, friction is a more complicated phenomenon than contact, and many effects are included in friction. Armstrong-Hélouvry et al. [1994] have introduced a nearly complete model which is called the seven parameter friction model. This model is able to capture almost all friction effects, like pre-sliding displacement, coulomb+viscous+stribeck curve, frictional memory and rising static friction. This is a quite complicated model which is hard to implement in simulation. Also, there are not enough experimental results in the published literature for determining the parameters.

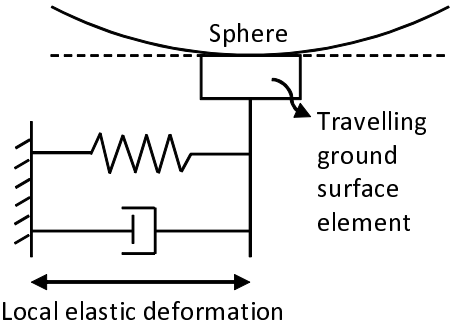


Figure 2.8: Friction model incorporating pre-sliding through local deformation of a tangential spring-damper pair

In this section, a simplified model for predicting the friction force during the contact between a sphere and a plate is derived. It is, essentially, a nonlinear, 2D version of the friction model in §11.8 of [Featherstone, 2008]. A physical interpretation of the model, which consists of a non-linear spring-damper pair, is shown in Fig. 2.8. The major difference between this model and previous friction force models in the literature, is the modelling of stiction force in the pre-sliding regime with a non-linear

equation. The non-linearity in the equation comes from the assumption that is also used in section 2.1.1 namely that the softer body contains a uniform distribution of infinitely many non-linear spring-damper pairs.

To work out the correct friction force, the first step is to work out what the friction force would be if the ground reaction force was inside the friction cone (i.e., the sphere is not slipping on the plate). This force is called $\boldsymbol{f}_{stick}$ and is calculated from

$$\boldsymbol{f}_{stick} = -k_t \boldsymbol{u} - b_t \boldsymbol{V}_{sph}\,, \tag{2.23}$$

where $\boldsymbol{u}$ is the tangential deformation of the ground at the contact point, $\boldsymbol{V}_{sph}$ is the tangential velocity of the bottom point of the sphere, and k_t and b_t are stiffness and damping coefficients of the tangential non-linear spring-damper pair, respectively. As the tangent plane is two-dimensional, $\boldsymbol{f}_{stick}$, $\boldsymbol{u}$ and $\boldsymbol{V}_{sph}$ are all 2D vectors. The coefficients k_t and b_t are functions of the contact area (A), and the stiffness and damping coefficients of each individual spring-damper pair in the normal direction (k and b). Assuming that the contacting surfaces are isotropic, k_t and b_t are calculated by

$$k_t = \int_A k(\zeta)\,\mathrm{d}A = 2\pi\beta \int_0^l r\delta^{-\frac{1}{2}}(r)\mathrm{d}r = 4\pi\beta R z^{\frac{1}{2}} \tag{2.24}$$

and

$$b_t = \int_A b(\zeta)\,\mathrm{d}A = 2\pi\alpha \int_0^l r\delta^{-\frac{1}{2}}(r)\mathrm{d}r = 4\pi\alpha R z^{\frac{1}{2}}\,. \tag{2.25}$$

Hence, the stiction force can be written as:

$$\boldsymbol{f}_{stick} = -K_t z^{\frac{1}{2}} \boldsymbol{u} - D_t z^{\frac{1}{2}} \boldsymbol{V}_{sph}\,, \tag{2.26}$$

where according to (2.14) and (2.15) K_t and D_t are given by

$$K_t = \frac{3}{2} K_n = 2E^* \sqrt{R} \tag{2.27}$$

and

$$D_t = D_n = 4\pi R\,\alpha\,. \tag{2.28}$$

The correct friction force will either be $\boldsymbol{f}_{stick}$, if it lies inside the friction cone, or else a force lying on the friction cone and having the same direction as $\boldsymbol{f}_{stick}$. The latter is called $\boldsymbol{f}_{slip}$, and is calculated from

$$\boldsymbol{f}_{slip} = \boldsymbol{f}_{stick} \times \frac{\mu F_n}{|\boldsymbol{f}_{stick}|}\,, \tag{2.29}$$

where μ is the coefficient of friction and F_n is the contact normal force. To avoid a divide-by-zero error, this equation is only calculated if $|\boldsymbol{f}_{stick}| > \mu F_n$. The correct

friction force, $\boldsymbol{F}_f$, is now given by

$$\boldsymbol{F}_f = \begin{cases} \boldsymbol{f}_{slip} & |\boldsymbol{f}_{stick}| > \mu F_n \\ \boldsymbol{f}_{stick} & \text{otherwise.} \end{cases} \tag{2.30}$$

2.3 Example: Rolling Motion of a Sphere

The rolling motion of a sphere on the ground is simulated in this section as an example of implementing the new 3D contact model. Using Simulink in MATLAB, a simulation is done for the rolling motion of a steel sphere on a cork plate (i.e. case (D) in Figs. 2.6 and 2.7). The sphere has a radius of $R = 1.65$cm and its mass is $m = 154$g. It is released from an initial height of 10cm of its CoM with $0.5\frac{\mathrm{m}}{\mathrm{s}}$ initial velocity in both x and y directions. The coefficient of friction is set to $\mu = 0.2$. The stiffness coefficients are calculated using mechanical properties of steel and cork as $K_n = 8.5 \times 10^6$ and $K_t = 12.75 \times 10^6$ and damping coefficients are $D_n = 3.1 \times 10^3$ and $D_t = 3.1 \times 10^3$.

2.3.1 Equations of Motion

The motion equations for a rolling sphere on the ground in 3D space are

$$\boldsymbol{f} = m\ddot{\boldsymbol{c}} \tag{2.31}$$

and

$$\boldsymbol{n}_C = \bar{\boldsymbol{I}}_c\, \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \bar{\boldsymbol{I}}_c\, \boldsymbol{\omega}\,, \tag{2.32}$$

where $\boldsymbol{f}$ is the resultant force acting on the sphere, $\boldsymbol{n}_C$ is the moment around the center of the sphere, $\boldsymbol{c}$ is a vector giving the position of the center of mass, $\bar{\boldsymbol{I}}_c$ is the moment of inertia of the sphere about its center of mass, and $\boldsymbol{\omega}$ is the angular velocity of the sphere. The velocity of the contact point is calculated by

$$\boldsymbol{V}_P = \dot{\boldsymbol{c}} + \boldsymbol{\omega} \times \overrightarrow{CP}\,, \tag{2.33}$$

in which $\overrightarrow{CP}$ is a vector from the CoM of the sphere to the contact point.

2.3.2 Results

Figure 2.9 shows the height of the lowest point of the sphere against time and Fig. 2.10 shows the normal force exerted by the ground on the sphere.

Figure 2.11 shows the contact force exerted on the sphere at the first bounce. It can be seen from this figure that the force is continuous and starts from zero at the beginning of the contact and comes back to zero smoothly. Also it is not sticky (negative) during the contact.

Figure 2.12 shows the friction force in the x direction. Because of the initial conditions, friction forces in both x and y directions are the same. Fig. 2.13 shows

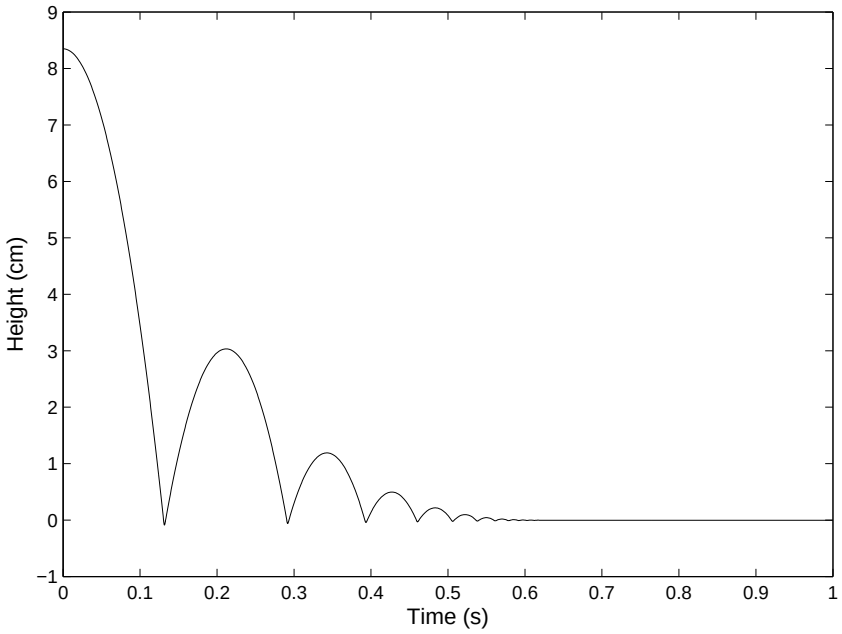


Figure 2.9: Vertical position of the lowest point of the sphere

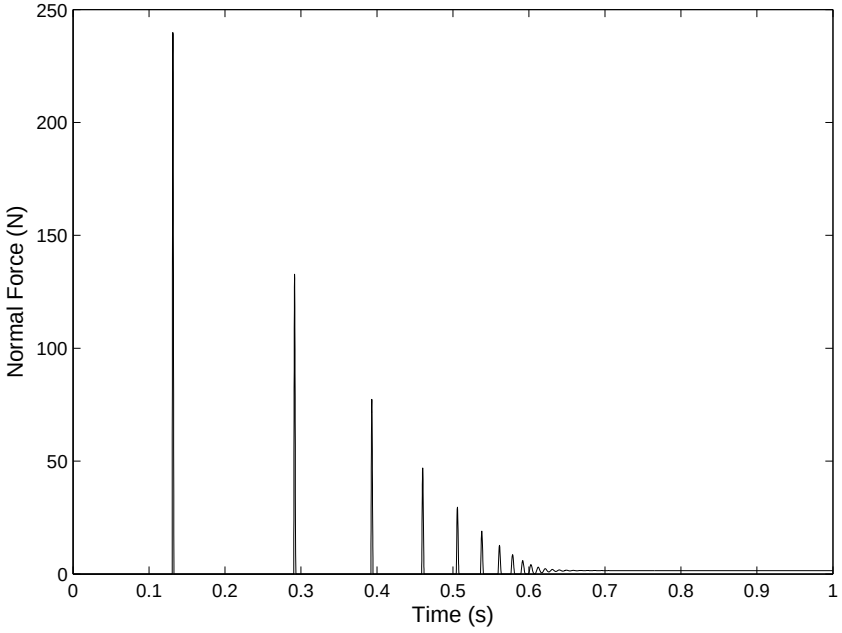


Figure 2.10: Contact normal force during the rolling motion of the sphere

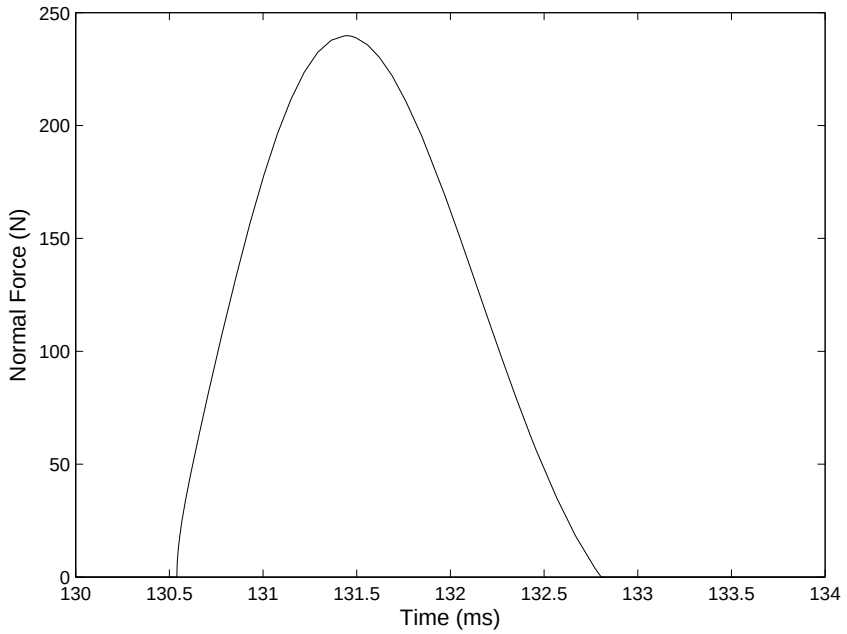


Figure 2.11: Enlargement of the first bounce in Fig. 2.10

the friction force in the x direction at the first bounce. It can be seen from this figure that the friction force is completely continuous and smooth. It is also shown that the contact starts with slipping of the sphere on the ground and after that, as the normal force approaches its maximum value (according to Fig. 2.11), the sphere sticks on the ground for less than 1ms and then again starts slipping until the end of the contact.

According to Fig. 2.13, the direction of the friction force is changed during the sticking period (at about $t = 132$ms) implying that the moving direction of the contact point on the sphere and therefore the sign of its velocity ($\boldsymbol{V}_{sph}$) is changed. The reason is that after hitting the ground with initial positive velocity, the sphere starts rolling on the ground (and thus spinning about its CoM) so the velocity of the contact point changes its direction. The change in the sign of the velocity is also illustrated in Fig. 2.14. This figure shows the x components of $\boldsymbol{c}$ and $\boldsymbol{V}_{sph}$ (i.e., the position of the CoM and the velocity of the contact point). Looking at this figure, it can be seen that after 0.5s, the velocity is zero but the position is still increasing, which means that the sphere is rolling on the ground.

2.4 Summary

A complete full 3D nonlinear contact model for both normal and friction forces is presented in this chapter. The new nonlinear normal force model is different from previous ones in the exponent of the ground deformation in the damping term. By comparing simulation results with empirical results of the contact between a sphere and a plate or between two spheres, it is shown that the new model can predict values

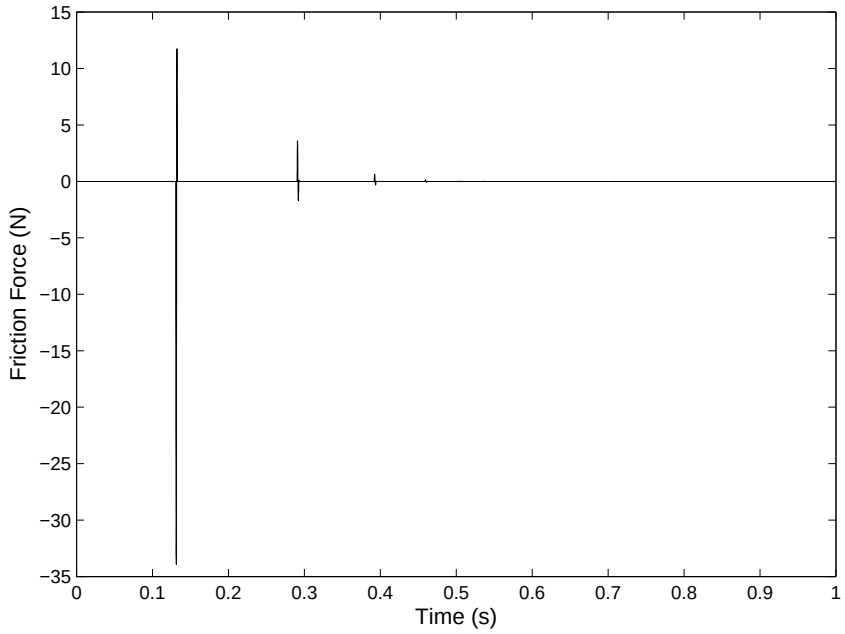


Figure 2.12: Friction force in x direction during the rolling motion of the sphere

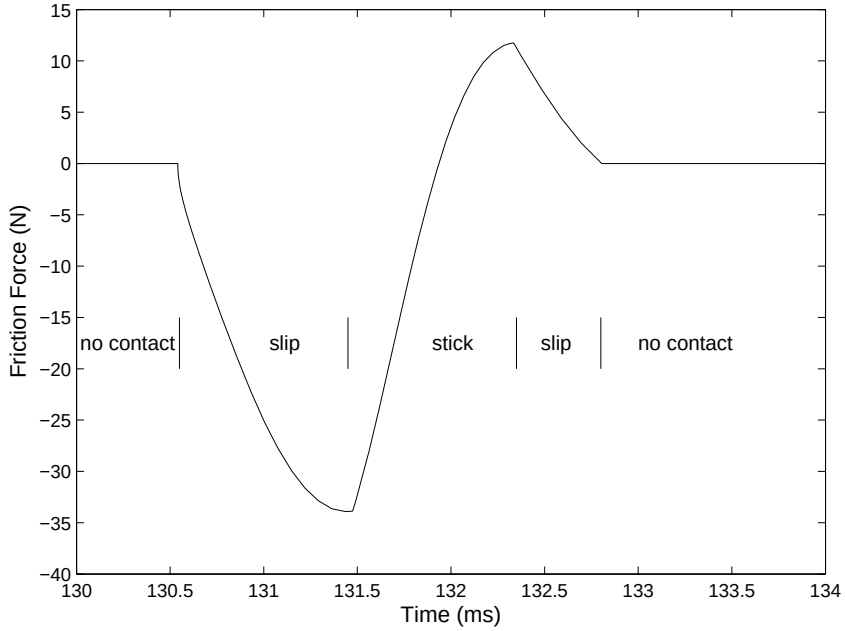


Figure 2.13: Enlargement of the first bounce in Fig. 2.12

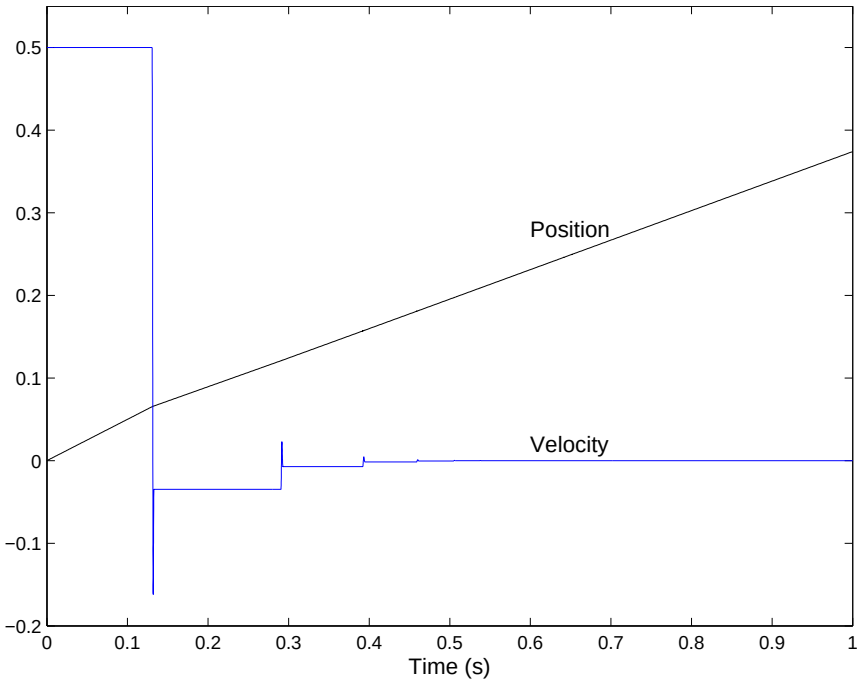


Figure 2.14: Position of the CoM and velocity of the contact point in x direction

of the coefficient of restitution more accurately than previous models. It is therefore likely to be a better choice when physically accurate simulations are required.

Finally, rolling motion of a sphere on the ground is simulated and the results are presented. As a non-spherical foot can be represented by a union of spheres, by using this model, it is possible to simulate the 3D motion of a robot's foot on the ground.

Chapter 3

Balancing Control Algorithm for an Under-actuated Planar Robot

In this chapter a new and simple balancing control algorithm is introduced for an under-actuated planar robot based on its angular momentum. The robot "2D balancer" consists of two bodies connected to each other by an actuated revolute joint (knee joint). The lower body is attached to the ground by a passive revolute joint. The 2D balancer resembles the acrobot (for acrobatic robot) first introduced by Hauser and Murray [1990].

It is proved that the new controller is able to stabilize the robot at any unstable balanced configuration in which the robot is physically controllable. The controller is able to follow setpoint commands, where only the target configuration is given, and also motion trajectory commands, where the motion of the actuated joint is a prescribed function of time. However, the latter necessarily involves tracking errors for the purpose of maintaining balance.

The proposed new controller is also able to cope with certain imperfections in the system such as model errors, state estimation errors and quantization errors. It is shown in simulations that the controller still shows reasonably good performance even in the presence of a significant amount of imperfections in the system.

The new controller is compared in simulation with three other balancing control algorithms in the literature. It is shown that, during the straightening motion, the new controller outperforms the other controllers when there are not any imperfections in the system. However, simulation results show that the controller is slightly more sensitive to errors in estimates of the vertical direction in comparison with Grizzle's control algorithm Grizzle et al. [2005].

Finally, the performance of the new control algorithm is studied in balancing motion of a curved-foot 2D balancer robot. The difference between the 2D balancer and a curved-foot 2D balancer is that, in the latter, the lower body of the robot contains a curve, called the foot, that is in rolling contact with a flat supporting surface (the ground). Hence, the original 2D balancer is a special case of the curved-foot 2D balancer where the curve is shrunk to a single point so the rolling contact is simplified to a passive revolute joint. The balancing motion of a one-leg robot with a rolling contact is considered because of its application in balancing legged-robots

when the shape of the foot is modelled with an arbitrary curve that is in rolling contact with the ground.

Most of the results presented in this chapter are published in [Azad and Featherstone, 2012].

3.1 Robot Model and Motion Equations

The 2D balancer is essentially an inverted double pendulum mechanism which is fixed passively to the ground via its first joint. This system is under-actuated with only one actuator that applies torque to the knee joint. Fig. 3.1 shows a schematic diagram of the 2D balancer and the parameters of the system.

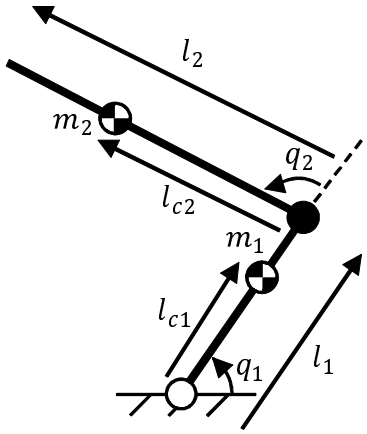


Figure 3.1: 2D balancer robot model and its parameters

The motion equations for this robot are:

$$\begin{aligned} 0 \;=\; & (c_1 + c_2 + 2c_3\cos(q_2))\ddot{q}_1 + (c_2 + c_3\cos(q_2))\ddot{q}_2 \\ & - (2c_3\sin(q_2)\dot{q}_1\dot{q}_2 + c_3\sin(q_2)\dot{q}_2^2) + c_4 g\cos(q_1) + c_5 g\cos(q_1+q_2) \end{aligned} \tag{3.1}$$

$$\tau = (c_2 + c_3\cos(q_2))\ddot{q}_1 + c_2\ddot{q}_2 + c_3\sin(q_2)\dot{q}_1^2 + c_5 g\cos(q_1+q_2) \tag{3.2}$$

where τ is the torque applied at the knee joint, I_1 and I_2 are the moments of inertia of the links about their centers of mass (CoM), g is the acceleration of gravity, and

$$\begin{aligned} & c = m_1 + m_2\,, \quad c_1 = m_1 l_{c1}^2 + m_2 l_1^2 + I_1\,, \quad c_2 = m_2 l_{c2}^2 + I_2\,, \\ & c_3 = m_2 l_1 l_{c2}\,, \qquad c_4 = m_1 l_{c1} + m_2 l_1\,, \qquad c_5 = m_2 l_{c2}\,. \end{aligned}$$

3.2 Balancing Controller

In this section a feedback control law for balancing motion of the robot is derived. First the balancing conditions of the robot are formulated and then an output func-

tion based on those conditions is defined in a way that zeroing the output function is equivalent to satisfying the conditions for balance. The feedback control law will drive the output function to zero exponentially and therefore it will stabilize the robot in its unstable balanced configuration.

Let X denote the horizontal displacement of the CoM relative to the first joint, so

$$X = \frac{1}{c}(c_4 \cos(q_1) + c_5 \cos(q_1 + q_2)) \,, \tag{3.3}$$

and let L denote the angular momentum of the robot about the first joint:

$$L = (c_1 + c_2 + 2c_3 \cos(q_2))\dot{q}_1 + (c_2 + c_3 \cos(q_2))\dot{q}_2 \,. \tag{3.4}$$

The conditions for balance are: $X = 0$ and $\dot{q}_1 = \dot{q}_2 = 0$. However, as $\dot{X}$ and L are both linear functions of $\dot{q}_1$ and $\dot{q}_2$, the two velocity constraints can be replaced with $\dot{X} = 0$ and $L = 0$. According to elementary classical mechanics, the rate of change of angular momentum of a multibody system about any fixed point equals the moment about that point of the external forces acting on the system. As L is chosen to be expressed at the first joint, which is passive, it follows that $\dot{L}$ must equal the moment of the gravitational force about the first joint. So

$$\dot{L} = -cgX \,, \tag{3.5}$$

and therefore also

$$\ddot{L} = -cg\dot{X} \,. \tag{3.6}$$

Equation (3.5) could also be derived by differentiating (3.4) and plugging it into (3.1). The conditions for balance can now be written as $L = \dot{L} = \ddot{L} = 0$; so it is possible to consider the angular momentum as an output function and define a feedback controller that drives it (and its derivatives) to zero exponentially. One possible control law is

$$\tau = k_{dd}\ddot{L} + k_d\dot{L} + k_pL \,, \tag{3.7}$$

where k_{dd}, k_d and k_p are controller gains. Another possibility is

$$\tau = -k_v\dot{X} - k_xX + k_pL \,, \tag{3.8}$$

where $k_v = cgk_{dd}$ and $k_x = cgk_d$. This equation is obtained by substituting (3.5) and (3.6) into (3.7).

3.2.1 Modifying the Controller

In the proposed control laws in (3.7) and (3.8), the control torque in the balanced configuration is zero, which happens only if the robot is in its vertical configuration. It means that the proposed controller is able to stabilize the robot only in the vertical unstable balanced configuration. In this subsection the controller is modified to be able to stabilize the robot in any other unstable balanced configuration as well as the vertical one.

Let τ_g and τ_g^d denote the gravity terms of the second link's motion equation at the current and desired configurations. So from (3.2) it is obvious that

$$\tau_g = c_5 g \cos(q_1 + q_2) \quad \text{and} \quad \tau_g^d = c_5 g \cos(q_1^d + q_2^d)\,, \tag{3.9}$$

where q_1^d and q_2^d are the desired values of q_1 and q_2 at the desired configuration, respectively. Any desired balanced configuration is characterised by the value of q_2^d. The value of q_1^d can then be calculated using the balance condition:

$$X = 0 \implies c_4 \cos(q_1^d) + c_5 \cos(q_1^d + q_2^d) = 0\,. \tag{3.10}$$

One possible way to modify the controller, to reach the above mentioned goal, is to add τ_g^d as an extra feed-forward term to the control law. So the discrepancy between τ_g in the motion equation and τ_g^d in the control law drives q_2 to its desired value q_2^d. The problems with this method are:

1. the range of the desired configurations in which the controller can balance the robot is limited to those with the links pointing upward (i.e., $\sin(q_1) > 0$ and $\sin(q_1 + q_2) > 0$).

2. in the presence of modelling error, which is unavoidable in practice, the value of τ_g^d is not accurate so the robot will not converge exactly to the desired configuration, but to a nearby one instead.

3. because of using a feed-forward term in the controller, any error in the actuator directly affects the accuracy of convergence to the desired configuration of the robot.

Another possible way to modify the control law (3.7) which does not lead to the above mentioned problems is adding a gravity compensator to the controller and a virtual spring to the robot model. The gravity compensator (τ_g) cancels out the gravity term in (3.2) and the virtual spring (i.e., linear rotational spring between the two links) adds an extra term in the left-hand side of (3.2). The effect of the virtual spring is to provide a feedback term in the control law to decrease the error between q_2 and q_2^d. The new modified control law is then

$$\tau = k_{dd}\ddot{L} + k_d\dot{L} + k_p L + k_s(q_2^d - q_2) + \tau_g \tag{3.11}$$

or alternatively

$$\tau = -k_v\dot{X} - k_x X + k_p L + k_s(q_2^d - q_2) + \tau_g \tag{3.12}$$

where k_s is the stiffness of the virtual spring in the robot model. The modified controller is relies on the virtual spring force, which is a feedback term, to drive q_2 to q_2^d. Also the modified controller offers one more parameter to tune (k_s) which can be used to improve the performance of the controller and extend the range of configurations in which the controller can successfully balance the robot.

3.3 Stability Analysis

Rewriting the nonlinear state-space equations for the robot as

$$\dot{\eta} = f(\eta) + g(\eta)\ \tau$$

where $\eta = (q_1 - q_1^d, q_2 - q_2^d, \dot{q}_1, \dot{q}_2)$, and employing a controller such that τ is a function of η, it follows that $\dot{\eta}$ is a function of η, so the closed-loop system can be described by a nonlinear equation of the form $\dot{\eta} = h(\eta)$. Linearizing about $\eta = 0$ yields

$$\dot{\eta} = A\,\eta$$

where $A = \frac{\partial h}{\partial \eta}|_{\eta=0}$ is a 4×4 matrix. To check the exponential (local) stability of the system and calculate the controller gains, the eigenvalues of A, which are the roots of its characteristic equation, need to be calculated. The characteristic equation for A has the general form:

$$a\lambda^4 + (gKk_{dd})\lambda^3 + (gKk_d + \beta + \gamma_1 k_s)\lambda^2 + (gKk_p)\lambda + (\gamma_2 k_s) = 0\,, \tag{3.13}$$

where

$$\begin{aligned}
a &= c_1c_2 - (c_3\cos(q_2^d))^2\,,\\
K &= c_4\sin(q_1^d)(c_2 + c_3\cos(q_2^d)) - c_5\sin(q_1^d + q_2^d)(c_1 + c_3\cos(q_2^d))\,,\\
\beta &= g(c_3c_5\cos(q_2^d)\sin(q_1^d + q_2^d) - c_2c_4\sin(q_1^d))\,,\\
\gamma_1 &= c_1 + c_2 + 2c_3\cos(q_2^d)\,,\\
\gamma_2 &= -g(c_4\sin(q_1^d) + c_5\sin(q_1^d + q_2^d))\,.
\end{aligned} \tag{3.14}$$

According to the Routh-Hurwitz stability criterion, and because a is always a positive constant independent of the choice of gains, the linearized system is locally exponentially stable if and only if

$$gKk_{dd} > 0\,, \quad \gamma_2 k_s > 0\,, \tag{3.15}$$

$$(gKk_d + \beta + \gamma_1 k_s) - \frac{k_p}{k_{dd}}a > 0\,, \tag{3.16}$$

$$gKk_p - \frac{gKk_{dd}\,\gamma_2 k_s}{(gKk_d + \beta + \gamma_1 k_s) - \frac{k_p}{k_{dd}}a} > 0\,. \tag{3.17}$$

It is obvious that the inequalities in (3.15) can be satisfied by choosing proper values for k_{dd} and k_s, if $\gamma_2 \neq 0$ and $K \neq 0$. Also it can be proved that under the same conditions, there always exists a set of values for k_d and k_p that satisfy (3.16) and (3.17), simultaneously.

Proof*[1]*: Let k_d be

$$k_d = \frac{\frac{k_p}{k_{dd}}a - \beta - \gamma_1 k_s + \delta}{gK},$$

where δ is a positive number. Inequality (3.16) then simplifies to $\delta > 0$ and inequality (3.17) can be written as

$$gKk_p - \frac{gKk_{dd}\,\gamma_2 k_s}{\delta} > 0\,.$$

Therefore, by choosing a proper value for k_p which satisfies

$$Kk_p > \frac{k_{dd}\gamma_2 k_s}{\delta}K$$

both inequalities in (3.16) and (3.17) will be satisfied, simultaneously. ■

Consequently, the Routh-Hurwitz stability criterion for the 2D balancer simplifies to

$$\gamma_2 \neq 0 \quad \text{and} \quad K \neq 0\,. \tag{3.18}$$

It will now be shown that $\gamma_2 = 0$ happens only when the CoM of the robot coincides with its first joint, which is called " neutral balance", and $K = 0$ only happens at desired configurations in which the robot is uncontrollable.

Neutral balance at $\gamma_2 = 0$: According to (3.10), it is clear that

$$c_4^2 \cos(q_1^d)^2 = c_5^2 \cos(q_1^d + q_2^d)^2$$

at every balanced configuration, which implies

$$c_4^2 - c_5^2 = c_4^2 \sin(q_1^d)^2 - c_5^2 \sin(q_1^d + q_2^d)^2\,. \tag{3.19}$$

So if $c_4 \neq c_5$ then $c_4^2 - c_5^2 \neq 0$ and consequently

$$c_4|\sin(q_1^d)| \neq c_5|\sin(q_1^d + q_2^d)|\,. \tag{3.20}$$

Therefore, the only condition that permits $\gamma_2 = 0$ is $c_4 = c_5$; and the only value of q_2^d that satisfies both (3.10) and $\gamma_2 = 0$ when $c_4 = c_5$ is $q_2^d = \pm\pi$. In this case, the robot is in a neutral balanced configuration with its CoM at the rotation center of the first joint.

Uncontrollability at $K = 0$: It can be proved that if $K = 0$ at a desired configuration then the velocity gain (G_V) is also zero at that configuration. The velocity gain, introduced in [Featherstone, 2012], is a dimensionless measure that expresses the degree to which the robot's CoM will move in response to motion of the actuated joint. When the velocity gain is zero, applying instantaneous torque to the actuator has no effect on the CoM of the robot. In other words if G_V at a desired configuration is zero then the robot is intrinsically uncontrollable at that configuration.

The following is the proof of the statement that at any desired unstable balanced

[1]This proof is due to Dr. Roy Featherstone.

configuration of the 2D balancer robot other than a neutral balanced configuration, $K = 0$ if and only if the velocity gain (G_V) is zero. Neutral balanced configurations are excluded because G_V as defined in [Featherstone, 2012] is not defined at such configurations.

Proof: According to Eq. 5 in [Featherstone, 2012], the velocity gain for a 2D balancer is

$$G_V(q_2) = \frac{c_y^2 + \frac{m_2\, c_x\, c_{2x}}{m_1+m_2}}{c_x^2 + c_y^2} - \frac{H_{12}}{H_{11}}\,, \tag{3.21}$$

where

$$\begin{aligned} H_{11} &= c_1 + c_2 + 2c_3\cos(q_2)\,, \\ H_{12} &= c_2 + c_3\cos(q_2)\,, \\ c_x &= (c_4 + c_5\cos(q_2))/c\,, \\ c_y &= c_5\sin(q_2)/c\,, \\ c_{2x} &= l_{c2}\cos(q_2) \quad \text{and} \quad c_{2y} = l_{c2}\sin(q_2)\,. \end{aligned}$$

So G_V at a desired configuration can be written as

$$G_V(q_2^d) = \frac{c_5^2 + c_4c_5\cos(q_2^d)}{c_4^2 + c_5^2 + 2c_4c_5\cos(q_2^d)} - \frac{c_2 + c_3\cos(q_2^d)}{c_1 + c_2 + 2c_3\cos(q_2^d)}\,. \tag{3.22}$$

The first denominator is zero if and only if $c_4 = c_5$ and $\cos(q_2^d) = -1$, which is the condition for neutral balance, so this possibility is excluded. The second denominator is nonzero because it is the first element of the robot's joint-space inertia matrix, which is a positive-definite matrix. By multiplying both sides of (3.22) by both denominators, it follows that $G_V(q_2^d) = 0$ if and only if

$$c_5(c_5 + c_4\cos(q_2^d))(c_1 + c_3\cos(q_2^d)) = c_4(c_4 + c_5\cos(q_2^d))(c_2 + c_3\cos(q_2^d))\,. \tag{3.23}$$

Also using (3.14) it is clear that $K = 0$ if and only if

$$c_4\sin(q_1^d)(c_2 + c_3\cos(q_2^d)) = c_5\sin(q_1^d + q_2^d)(c_1 + c_3\cos(q_2^d))\,. \tag{3.24}$$

The proof is now concluded for two different cases:

First, if $\sin(q_2^d) = 0$: In this case, $\cos(q_2^d) = \pm 1$ and the balance condition in (3.10) becomes

$$(c_4 \pm c_5)\cos(q_1^d) = 0\,.$$

Given that neutral balance configurations are excluded, then $\cos(q_1^d) = 0$ and consequently $q_1^d = \frac{\pi}{2}$ and $\sin(q_1^d + q_2^d) = \cos(q_2^d) = \pm 1$. Substituting these values into (3.23) and (3.24) both of these conditions become identical.

Second, if $\sin(q_2^d) \neq 0$: In this case the balance condition in (3.10) can be written as

$$c_4\cos(q_1^d) + c_5\cos(q_1^d)\cos(q_2^d) - c_5\sin(q_1^d)\sin(q_2^d) = 0\,. \tag{3.25}$$

Multiplying both sides of the above equation by $\sin(q_2^d) \neq 0$, yields

$$c_4 \cos(q_1^d) \sin(q_2^d) + c_5 \cos(q_1^d) \cos(q_2^d) \sin(q_2^d) - c_5 \sin(q_1^d) \sin(q_2^d)^2 = 0\,, \tag{3.26}$$

which implies

$$c_4 \cos(q_1^d) \sin(q_2^d) + c_5 \cos(q_2^d) \sin(q_1^d + q_2^d) = c_5 \sin(q_1^d)\,. \tag{3.27}$$

Adding the term $c_4 \sin(q_1^d) \cos(q_2^d)$ to both sides of the above equation, it simplifies to

$$\sin(q_1^d + q_2^d)(c_4 + c_5 \cos(q_2^d)) = \sin(q_1^d)(c_5 + c_4 \cos(q_2^d))\,. \tag{3.28}$$

Now, if $\sin(q_1^d + q_2^d) \neq 0$ and $\sin(q_1^d) \neq 0$, then multiplying each side of (3.28) by the corresponding side of (3.24), it is concluded that $K = 0$ if and only if

$$c_4(c_4 + c_5 \cos(q_2^d))(c_2 + c_3 \cos(q_2^d)) = c_5(c_5 + c_4 \cos(q_2^d))(c_1 + c_3 \cos(q_2^d))\,, \tag{3.29}$$

which is equivalent to $G_V = 0$ in (3.23). Otherwise, if $\sin(q_1^d + q_2^d) = 0$, then from (3.24) it is clear that $K = 0$ if and only if $c_2 + c_3 \cos(q_2^d) = 0$. Also, considering (3.28), it is concluded that $c_5 + c_4 \cos(q_2^d) = 0$ when $\sin(q_1^d + q_2^d) = 0$. On the other hand, if $\sin(q_1^d) = 0$ then $c_4 + c_5 \cos(q_2^d) = 0$ and also $K = 0$ if and only if $c_1 + c_3 \cos(q_2^d) = 0$. Thus, if $\sin(q_1^d + q_2^d) \neq 0$ or $\sin(q_1^d) \neq 0$, then both of sides of (3.23) become zero which proves that $K = 0$ if and only if $G_V = 0$. ■

In summary, the proposed controller is able to stabilize the robot in any unstable balanced configurations except the neutral ones ($\gamma_2 = 0$) or those in which the robot is uncontrollable ($K = 0$).

3.4 Gain Calculations

Given the stability analysis in the previous section, a fast controller is defined by using the pole placement method to maximize the speed of the slowest pole. In this case all poles of the closed loop system are placed at the same point which means that

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = -p\,,$$

where p is related directly to k_s as

$$\prod_{i=1}^{4} \lambda_i = p^4 = \frac{\gamma_2 k_s}{a}\,, \tag{3.30}$$

where γ_2 and a are defined in (3.14). The above relationship between k_s and p implies the direct effect of the value of k_s on the location of the poles and therefore on the overall rate at which the controller converges to the desired configuration (convergence rate of the controller). Increasing the convergence rate, by increasing the value of k_s, will reduce the region of stability around the desired configuration and cause more energetic movements of the robot. Thus, choosing a value for k_s requires

a trade-off between the convergence rate, the region of stability and the consumed energy. Once k_s has been chosen, the other gains must then be chosen so that (3.13) matches the polynomial

$$(\lambda + p)^4 = \lambda^4 + 4p\lambda^3 + 6p^2\lambda^2 + 4p^3\lambda + p^4 = 0\,. \tag{3.31}$$

With this choice of gains, all poles of the closed loop system are negative, so the system is asymptotically stable and the remaining controller gains become

$$k_p = \frac{4\,p^3\,a}{gK}\,, \quad k_d = \frac{6\,p^2 a - \beta - \gamma_1 k_s}{gK}\,, \quad k_{dd} = \frac{4\,p\,a}{gK}\,. \tag{3.32}$$

3.5 Following a Trajectory

As described in section 3.2, making the robot move from one balanced configuration to another can easily be accomplished by feeding the desired value of q_2^d to the control law in (3.11). However, making the robot follow a prescribed trajectory is harder. In this section the control law is modified to enable the robot to follow arbitrary motion trajectories. Here, an 'arbitrary' motion trajectory for the 2D balancer is defined to be an equation that specifies q_2^d as an explicit function of time and is not constrained to have any particular algebraic form. Also the desired trajectory is not required to be a member of a special class of trajectories that the robot can follow exactly, such as the trajectories described in [Berkemeier and Fearing, 1999]. In general, following such a trajectory with no tracking error is physically impossible without losing balance. The controller follows the trajectory as closely as possible and therefore it is expected that the controller generates an appropriate (small) tracking error while it is trying to follow an arbitrary trajectory and maintain its balance at the same time.

Clearly, there will be commanded trajectories that will cause the controller to lose balance. No attempt has been made to characterize these trajectories. The simulations in the next section consider only trajectories that the controller can follow successfully.

To implement trajectory following, the control law in (3.11) is used, but L is replaced with $(L - L_d)$. L_d is the theoretical value of L assuming that the robot is perfectly following the desired trajectory at the current instant. Since the desired trajectory of the robot, which is determined by q_2^d, is a function of time then L_d is also a function of time depending on q_2^d and the desired velocity ($\dot{q}_1^d$ and $\dot{q}_2^d$) of the robot. According to (3.4) we have

$$L_d = (c_1 + c_2 + 2c_3\cos(q_2^d))\dot{q}_1^d + (c_2 + c_3\cos(q_2^d))\dot{q}_2^d\,, \tag{3.33}$$

where $\dot{q}_2^d$ is computed from the reference trajectory and $\dot{q}_1^d$ is calculated from (3.3) assuming that $\dot{X} = 0$ and the robot is balanced.

During a trajectory-following motion of the robot, the gains of the controller are not constant but vary as a function of the desired configuration at each instant of the motion.

3.6 Simulations

The parameters for the balancer that are used in the simulations are the same as the "*good balancer*" in [Featherstone, 2012] , and are listed in the row 'simulator's model' in Table 3.1. The bottom row in this table lists the incorrect parameters that will be used later to test the controller's robustness to modelling errors.

Table 3.1: Model parameters used in the simulations

	m_1 m_2	l_1 l_2	l_{c1} l_{c2}	I_1 I_2
simulator's model	0.49 0.11	0.4 0.6	0.1714 0.4364	0.0036 0.0043
controller's model	0.44 0.1	0.4 0.6	0.1543 0.4078	0.0032 0.0039

To demonstrate the performance of the new controller, the results of three sets of simulations are presented in this section. All simulations have been done with Simulink. In the first set, presented in subsection 3.6.1, it is assumed that there are no imperfections in the system and the controller has access to the exact model of the plant (i.e., the controller uses the same model parameters as the simulator). In this set of simulations, a continuous variable time step 4^{th} order Runge-Kutta integrator (ode45) with maximum step size of 10ms is used.

In the second and third sets of simulations, presented in subsections 3.6.2 and 3.6.3, some imperfections are added to the system to model a variety of practical effects. The difference is that the imperfections that influence the controller's perception of the balanced configuration of the robot (q_1^d and q_2^d) are considered only in subsection 3.6.3. As already mentioned in subsection 3.2.1, the controller calculates the balanced configuration (q_1^d) given the value of q_2^d and using (3.10). If the imperfections do not change the ratio of c_4/c_5, then the controller is able to correctly predict the value of q_1^d in the desired balanced configuration. In this case, the controller is more robust than in the alternative scenario.

For the sets of simulations in subsections 3.6.2 and 3.6.3, the controller is modelled as a sampled-data system running at a servo rate of 1kHz, and so a continuous variable time step 4^{th} order Runge-Kutta integrator (ode45) with maximum step size of 1ms is used to simulate the dynamics of the robot. The controller uses a first-order integrator for its own internal calculations.

3.6.1 Perfect Modelling

To show the best performance of the controller in theory, simulations have been done for a theoretically perfect system where the controller uses:

1. the same model as the simulator (i.e. the simulator's model in Table 3.1),
2. the actual values of the robot's state variables,

3. an actuator which is an ideal torque source and operates without any torque limits.

The gains are calculated so as to place the poles of the closed-loop system at -7 at the desired configuration (i.e. $p = 7$).

Figure 3.2 shows the balancer moving from a crouched position ($q_2 = -\frac{\pi}{2}$) to the upright balanced position ($q_2^d = 0$) and Fig. 3.3 shows the robot starting from the upright position and moving to a balanced crouching position ($q_2 = -\frac{\pi}{2}$). In both examples it can be seen that the convergence rate is quite high and the settling time is about one second.

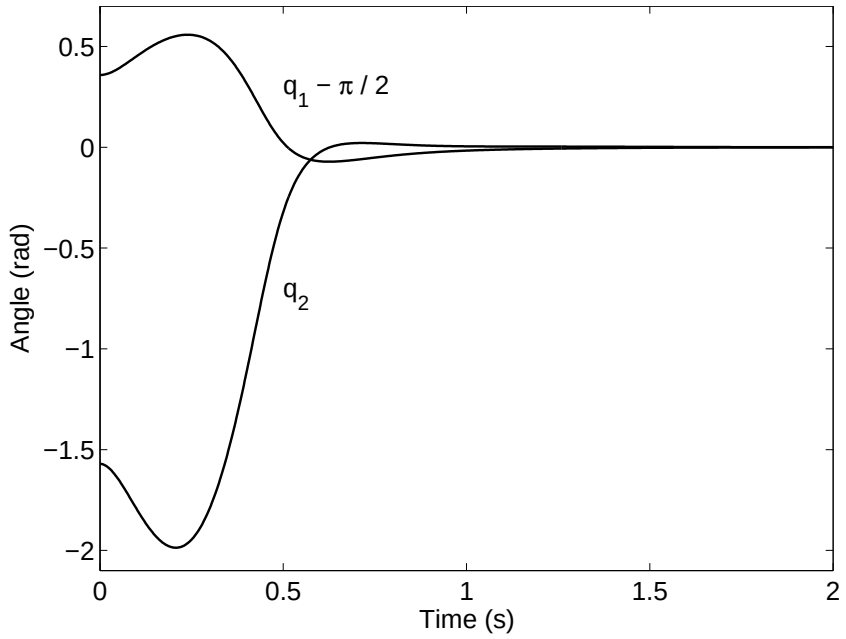


Figure 3.2: Robot's straightening motion—perfect modelling

Figure 3.4 shows the trajectory tracking performance of the robot when the command for q_2 consists of two steps, a linear ramp and a sine wave function. Although there are some tracking errors in the resulting motion, the robot still follows the desired trajectory very well. It is also noticeable in all Figs. 3.2, 3.3 and 3.4 that once the robot receives a command to follow, it first starts moving in the opposite direction of the command and then it quickly comes back to the direction of the desired trajectory. This behaviour is a property of all double-pendulum under-actuated robots with negative velocity gains[2], and tracking errors due to that behaviour are physically unavoidable.

Finally, Fig. 3.5 shows how q_1 converges to $q_1^d = \frac{\pi}{2}$ during a straightening motion starting from an initial crouched configuration ($q_2 = -\frac{\pi}{2}$) using different values of

[2]A double-pendulum robot with a negative velocity gain tips in the opposite direction of the one in which the robot is commanded. For example, to make a double-pendulum begin to tip forward (starting from its balanced upright position), it is necessary to bend the upper link backwards, at least momentarily.

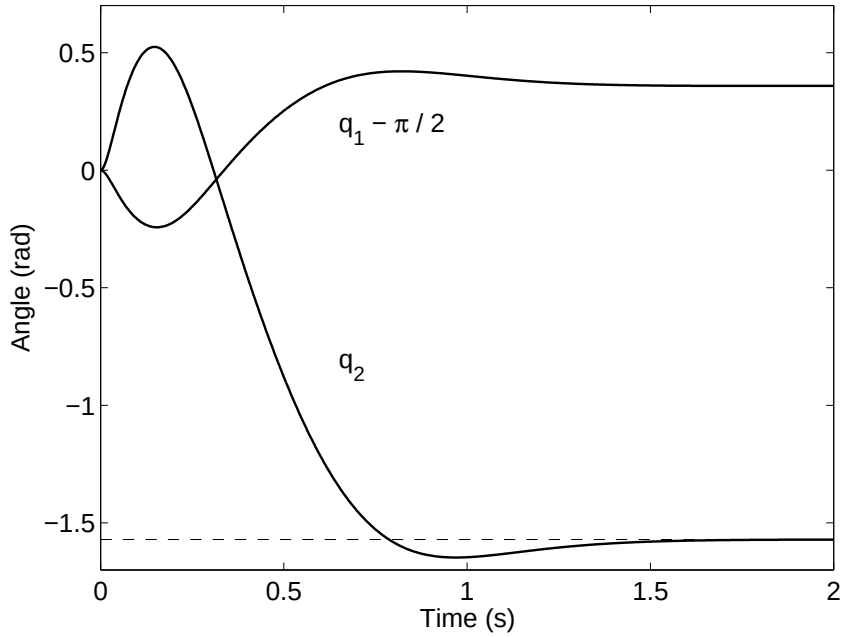


Figure 3.3: Robot's crouching motion—perfect modelling

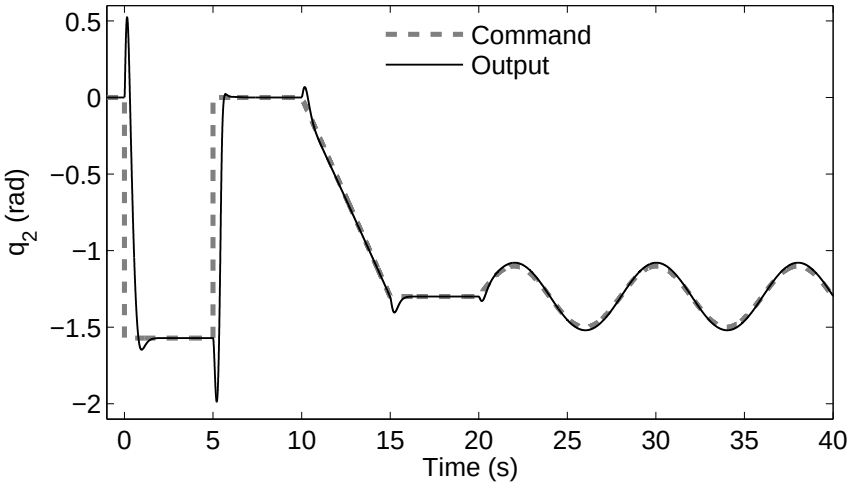


Figure 3.4: Trajectory-tracking performance of the robot—perfect modeling

p. Comparing the graphs in this figure, it follows that by increasing p the robot will reach the desired configuration faster. However, using higher values of p will cause the controller to produce more energetic movements and therefore make larger initial movements in the opposite direction at the beginning of the motion, as well as larger overshoots as q_1 approaches $\frac{\pi}{2}$. Thus, as has already been mentioned, choosing p (i.e. choosing k_s) requires a trade-off between speed and overshoot in which the dynamics of the robot and the available torque have to be considered.

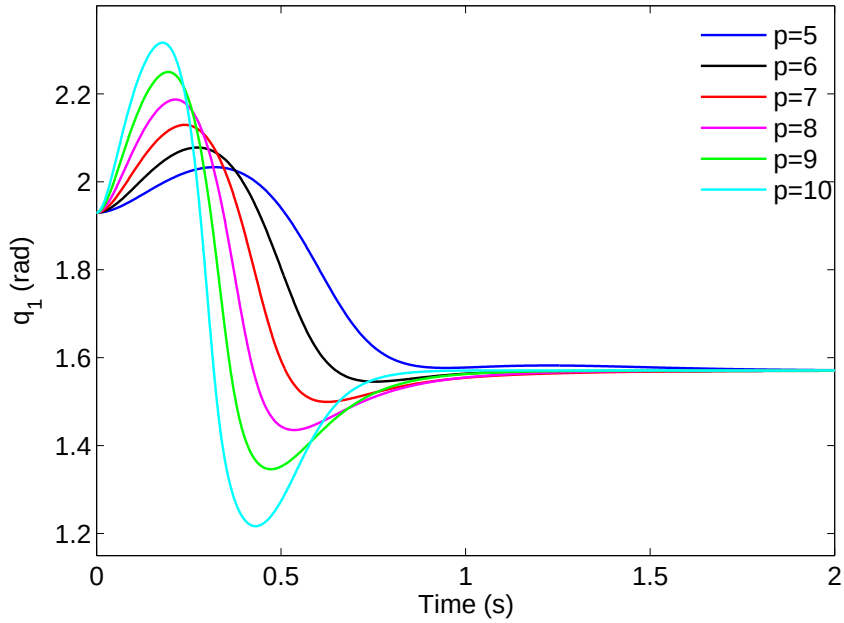


Figure 3.5: Effect of different pole locations on a straightening motion

3.6.2 Considering Model Imperfections

In the second set of simulations we demonstrate the performance of the controller in more realistic situations by including imperfections in the simulations. The imperfections considered in this subsection are those that do not affect the controller's calculation of the balanced configuration of the robot (i.e., the calculation of q_1^d from q_2^d via (3.10)). Specifically, the following imperfections are considered.

1. *Discrete execution* – As already mentioned, the controller is modelled as a sampled-data system. Thus, the controller evaluates its inputs every 1ms, performs calculations using these inputs and stored data from previous executions, and updates the outputs and stored data. Since the outputs are updated immediately after the inputs are sampled, the average time delay between inputs and outputs is 0.5ms.

2. *Modelling error* – Modelling error is the error that occurs in model-dependent calculations of the controller when it uses an 'estimated' model of the robot

instead of the 'exact' one. The parameters of the estimated and exact models are listed in Table 3.1 under "controller's model" and "simulator's model", respectively. The parameters of the estimated model are different from the exact one by almost 10%. The parameters have been calculated in a way that they do not affect the ratio c_4/c_5 and therefore the controller's perception of balanced configurations will be correct.

3. *Quantization* – Two encoders have been used at the joints to read the joint angles. Thus, outputs of the encoders are quantized values of the joint angles instead of their actual values. The encoder in the passive joint has a resolution of 8192 counts per revolution, and the encoder in the active joint which is embedded in the motor has a resolution of 512 counts per revolution. A gearbox with a reduction ratio of 66 is used to transmit the torque from the motor to the joint. Thus, the resolution at the active joint is 66×512 counts per revolution.

4. *Velocity estimation* – Instead of integrating the angular accelerations to obtain the angular velocities ($\dot{q}_1$ and $\dot{q}_2$), the controller uses the quantized values of the joint angles and estimates the velocities via a velocity estimation method. The velocity estimator which is used in the simulations is a linear observer as described in [Harnefors and Nee, 2000]. The linear observer works as follows. If $\theta(n)$ is the angle measurement at the n^{th} time step, coming from an encoder then the estimated velocity of that angle at the next time step, $\hat{\dot{\theta}}(n+1)$ is

$$\hat{\dot{\theta}}(n+1) = \hat{\dot{\theta}}(n) + T_s\rho^2(\theta(n) - \hat{\theta}(n))\,, \tag{3.34}$$

where $\hat{\theta}(n)$ is an internal state variable that is updated via

$$\hat{\theta}(n+1) = \hat{\theta}(n) + T_s(\hat{\dot{\theta}}(n) + 2\rho(\theta(n) - \hat{\theta}(n)))\,. \tag{3.35}$$

In the above equations, T_s is the time step, which is 1ms, and ρ is a gain parameter which is set to 200 in our simulations.

5. *DC motor model and velocity servo* – A brushless DC motor is used to actuate the knee joint via a reduction gearbox. A velocity servo, which is in fact a PI controller, is used to command the motor. The servo receives the desired velocity ($\dot{q}_2^d$) from the balancing controller as an input and works out the required voltage of the motor. The proportional and integral gains of the PI controller are 15 and 200, respectively. It is assumed that the sampling rate of the servo is sufficiently fast and therefore the servo is regarded as a continuous-time system which has access to the exact value of the velocity of the motor for its internal calculations.

   Let i and T denote the current through the motor and the output torque, respectively. Then

$$T = K_t\, i\,, \tag{3.36}$$

   where K_t is the torque constant of the motor. The current (i) is an internal state

Table 3.2: DC motor parameters

stall torque	127mN.m	nominal torque	23.1mN.m
no-load speed	12900rpm	nominal speed	10500rpm
resistance(R)	13.1 Ω	inductance($\mathcal{L}$)	0.729mH
torque constant(K_t)	34.8mN.m.A^{-1}	rotor inertia	4.45g.cm^2
speed constant (K_e)	274rpm.V^{-1}	power	25W

variable which is calculated using the following differential equation:

$$V - K_e\dot{\phi} = \mathcal{L}\frac{\mathrm{d}i}{\mathrm{d}t} + R\,i\,, \tag{3.37}$$

where V is the voltage across the motor, $\dot{\phi}$ is the rotor velocity (computed from $\dot{q}_2$), and K_e, $\mathcal{L}$ and R are constant parameters. The parameters of the DC motor are listed in Table 3.2. They have been obtained from the data sheet for part number 283860 from MaxonMotors. According to the data sheet, the voltage of the motor is restricted to $\pm$48V implying that there is a speed-dependent torque limit on the actuator. This motor was chosen for no reason other than as a source of realistic parameter values.

As already mentioned, a reduction gearbox with zero-backlash is used to transmit the torque from the motor to the joint. Based on part number 166940 from MaxonMotors, the efficiency of the gearbox is 70%. The inertia of the gearbox is assumed to be zero but the motor's rotor inertia is incorporated into the equation of motion of the robot after being multiplied by the square of the gear ratio.

Figure 3.6 shows a schematic diagram of the subsystems used in the simulations in this subsection. The plant, including the model of the mechanism, the DC motor model and the velocity servo, are treated as continuous-time subsystems, whereas the controller, including all other parts, are treated as sampled-data subsystems with a sampling time of 1ms. The encoders are not actually part of the controller but are treated as a sampled-data subsystem.

According to the signal flow in Fig. 3.6, the plant outputs the joint angles (q_1 and q_2) to the encoder block, which passes the quantized values of the joint angles ($\hat{q}_1$ and $\hat{q}_2$) to the velocity estimator block. The control law block calculates the desired value of the torque on the basis of the estimated joint angles, the estimated angular velocities ($\hat{\dot{q}}_1$ and $\hat{\dot{q}}_2$) and the command signal using the estimated model. The estimated model is also used by the forward dynamics block to compute the desired acceleration and therefore the desired velocity of the knee joint ($\dot{q}_2^d$) according to the desired torque. The velocity servo uses the desired velocity and calculates the required voltage as already described.

Figure 3.7 shows the balancing performance of the robot when it starts from a crouched position ($q_2 = -\frac{\pi}{2}$) and is aiming for the upright position. Figure 3.8 shows the crouching motion of the robot when it starts from the balanced upright position.

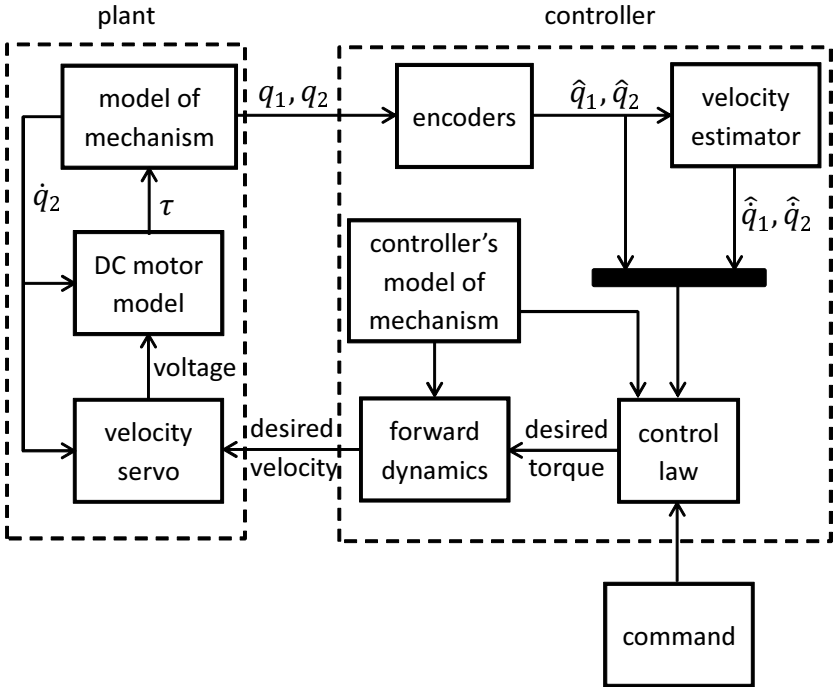


Figure 3.6: Block diagram of the system with model imperfections

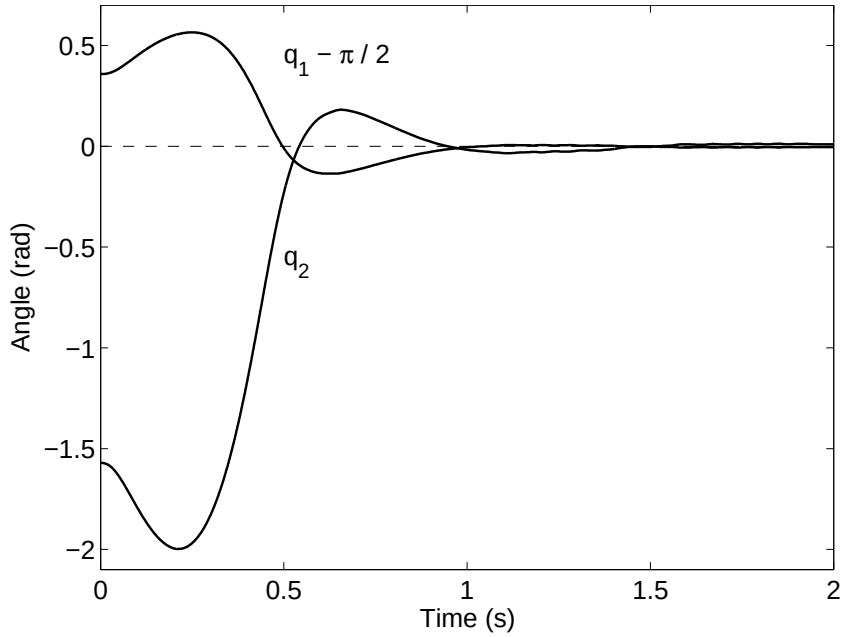


Figure 3.7: Robot's straightening motion with imperfections

By comparing the straightening motion of the perfect system (Fig. 3.2) and the system

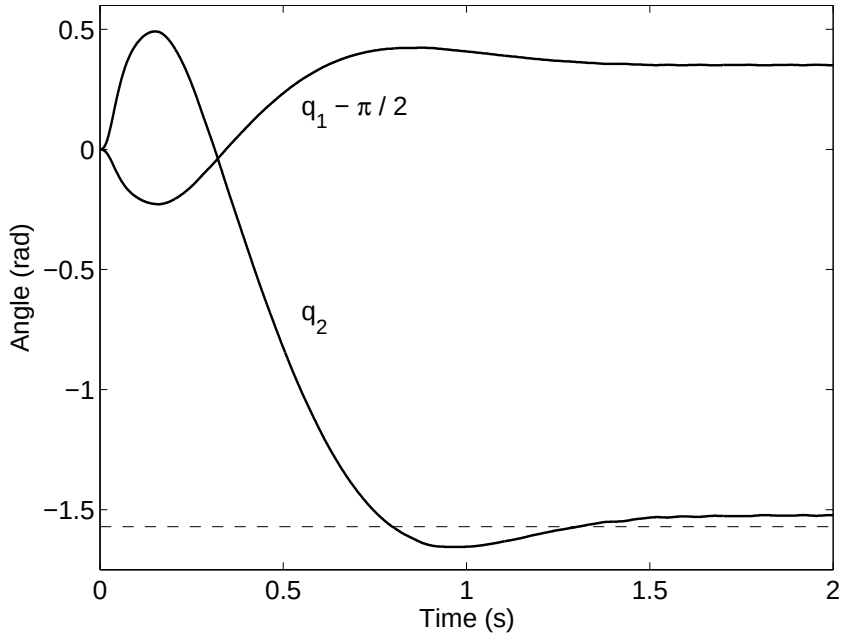


Figure 3.8: Robot's crouching motion with imperfections

with imperfections (Fig. 3.7), it can be seen that overshoot is larger and the settling time is longer in the latter. Regarding the crouching motion of these two systems (Figs. 3.3 and 3.8), clearly there are not any noticeable differences in overshoot and settling time but a small steady-state error in the system with imperfections (Fig. 3.8). The reason behind the steady-state error is that the gravity compensation term (τ_g) is a model-dependent variable (depends on c_5) which is incorrectly calculated by the controller in the system with imperfections. Notice that the steady-state error does not appear in the straightening motion because $\tau_g = 0$ in the upright balanced configuration. In spite of the already mentioned differences in the balancing motion of the perfect system and the one with imperfections, the overall performance of the controller is almost the same implying that it is robust to the types of imperfections considered in this subsection.

Figure 3.9 shows the trajectory tracking performance of the robot in the presence of model imperfections in the system. The robot does not follow the desired trajectory as well as it does in the perfect modelling case, but it still shows a reasonably good performance. The output shows a lag in following the ramp trajectory; and there is a steady-state or offset error in every configuration other than the upright one, which is due to the error in the calculation of τ_g.

3.6.3 Imperfections that Influence the Balanced Configuration

There are two types of error that influence the controller's calculation of the balanced configuration which are modelling errors (errors in the estimated model) and bias

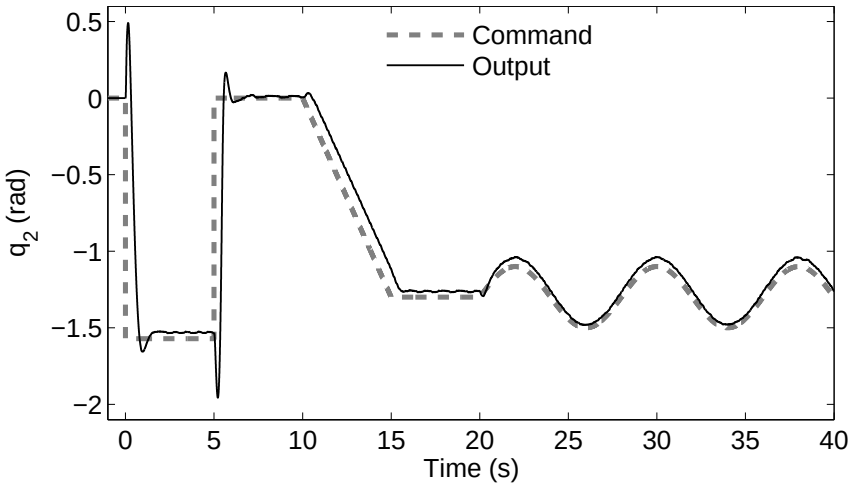


Figure 3.9: Trajectory-tracking performance of the robot with imperfections

in the encoders. Since both of these types of error have the same influence on the controller's perception of the balanced configuration, considering only one of them is enough. To investigate the effect of this type of error on the controller's performance, a small bias of 1° is added to the passive joint's encoder.

Figures 3.10 and 3.11 show the straightening and crouching motions of the robot in the presence of all imperfections, including the 1° bias. Both figures show significant steady-state errors. In the straightening motion there is almost a 6° error in the value of q_1 and 18° in q_2; while in the crouching motion the errors are 2° and 10.6°, respectively. Figure 3.11 also shows a wobble near the beginning, which is due to the actuator hitting its saturation limit.

In conclusion, the proposed controller in this chapter is robust to the kinds of imperfections which are likely to appear in a practical system except those that influence the controller's perception of the robot's balanced configuration. In other words, the controller is sensitive to errors in the estimate of the direction of gravity.

3.7 Comparison to other Control Algorithms

In this section the performance of the controller is compared with three other balancing control algorithms in the literature: Spong's LQR controller [Spong, 1995], Berkemeier and Fearing's linear controller [Berkemeier and Fearing, 1999] and Grizzle's nonlinear controller [Grizzle et al., 2005]. First the performance of all four controllers are compared during the straightening motion of the robot when there are no imperfections in the system. Then the balancing performance of the controllers are compared when there is a single imperfection in the system, namely a 1° bias in the passive joint's encoder.

To make the comparisons as fair as possible, the gains have been set as follows. For Spong's LQR controller, different values for Q and R have been used by different

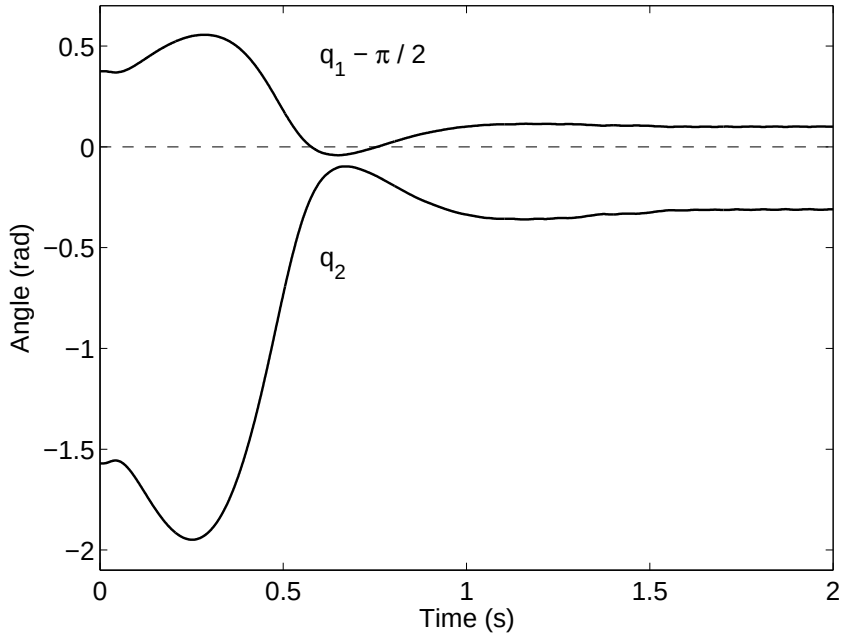


Figure 3.10: Robot's straightening motion with imperfections and encoder bias

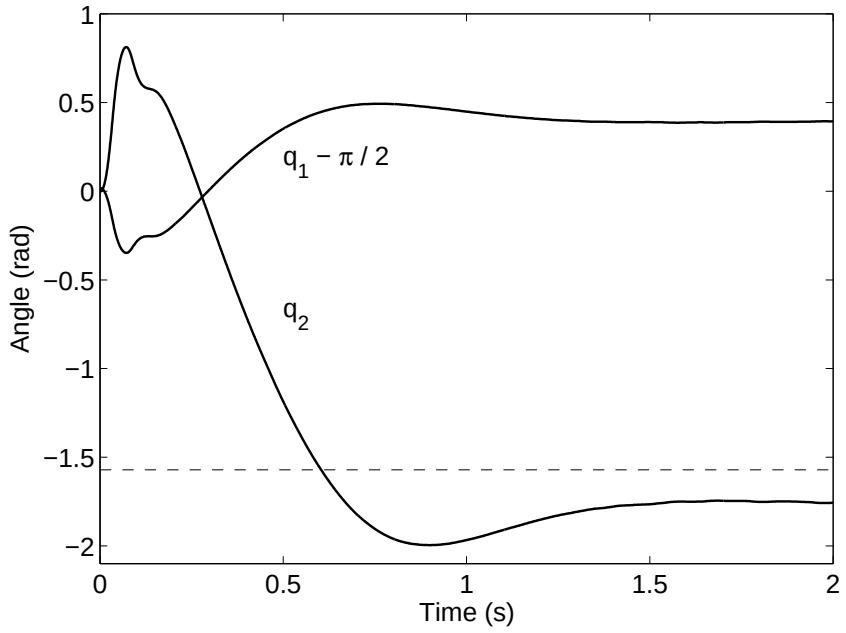


Figure 3.11: Robot's crouching motion with imperfections and encoder bias

researchers [Spong, 1994, 1995; Brown and Passino, 1997; Xin and Kaneda, 2001; Mahindrakar and Banavar, 2005; Lai et al., 2005; Inoue et al., 2007] but the most common approach is to set Q to an identity matrix and then adjust the value of scalar R to get good results in simulations. In this section R is set to 100. Although assigning these values to Q and R are not the best ones, but simulation results show that they are good choices and on the basis of the definition of a LQR controller, it is optimal for the given values of R and Q. For Berkemeier and Fearing's controller, all of the poles are placed at -5 which is the same value used in [Berkemeier and Fearing, 1999]. Simulation results show that using less negative values for the poles will cause slower convergence and using more negative ones will result in instability. The convergence rate of Grizzle's controller is determined by the eigenvalues of the error equation. Thus, these eigenvalues have the same effect as the poles of the closed loop system $(-p)$ in the proposed controller in this chapter. Therefore, to have a fair comparison, both the eigenvalues of the error equation for Grizzle's controller and $-p$ have been set to -7.

Figure 3.12 shows how the robot converges to its upright balanced configuration starting from an initial crouched position ($q_2 = -\frac{\pi}{2}$) using each of the four control algorithms. It can be seen that, apart from Berkemeier and Fearing's linear controller, which has a large overshoot and a long settling time, the other controllers all perform reasonably well. Although Spong's controller converges to $\frac{\pi}{2}$ faster than the others, but produces much larger overshoot with respect to Grizzle's controller and the controller proposed in this chapter. Figure 3.13 shows the same balancing motion

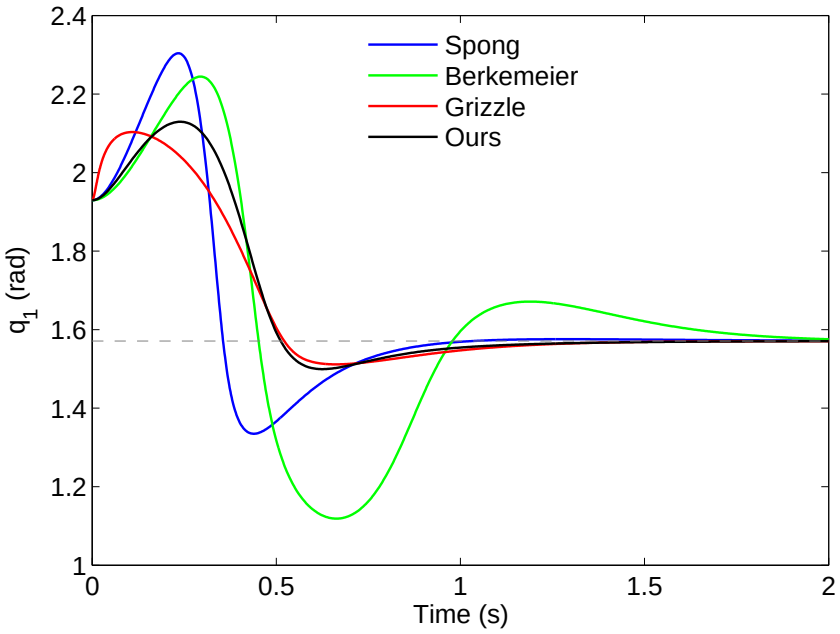


Figure 3.12: Robot's straightening motion—perfect modelling

assuming that there is a one-degree bias in the first joint's encoder. As can be seen, there is a steady-state error with all four controllers. The magnitude of this error is

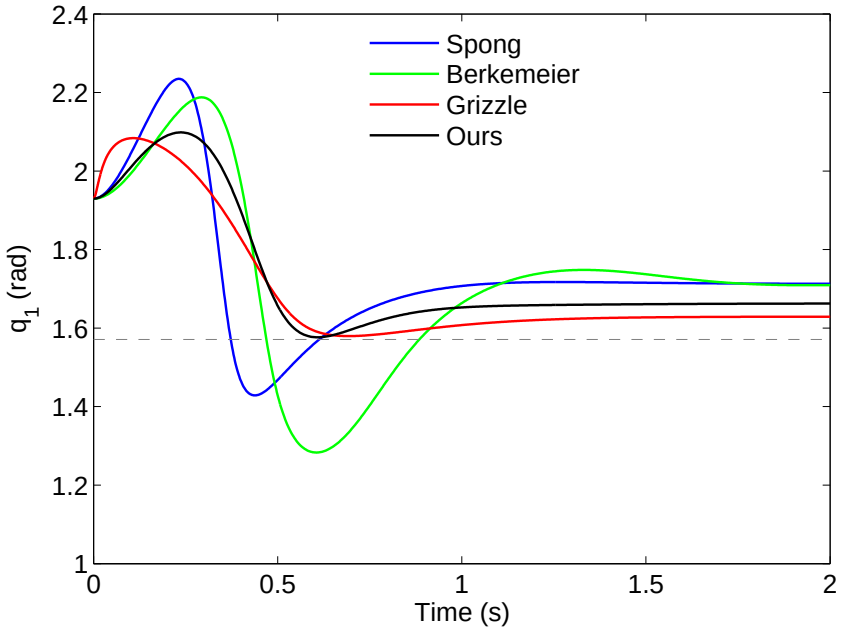


Figure 3.13: Robot's straightening motion with 1° bias in the passive joint's encoder

the same in Spong's and Berkemeier's controller, and it is roughly one third smaller for the proposed controller, and one third smaller still for Grizzle's.

Comparing the balancing performance of Grizzle's controller and the other ones in both Figs. 3.12 and 3.13 it is clear that Grizzle's controller uses a very large torque at the beginning of the motion to make the robot reach its maximum velocity within the first few milliseconds of the motion. Having access to such a large torque is not possible in a real robot. The performance of Grizzle's controller in a real robot would be different from that demonstrated in Figs. 3.12 and 3.13 in relation to settling time.

3.8 Balancing Motion of a Curved-foot 2D Balancer Robot

The balancing problem of an inverted pendulum on a rolling contact has been studied by many researchers [Ha and Yuta, 1994; Nakajima et al., 1997; Lauwers et al., 2006; Muskinja and Tovornik, 2006; Jung and Kim, 2008; Nagarajan et al., 2009]. The most famous such study is called 'ballbot'. Ballbot, a robot built at Carnegie Melon University [Hollis, 2008], consists of a main body (an inverted pendulum) and a spherical wheel which is free to roll on the ground. Ballbot is a mobile robot that is able to balance itself at a place as well as move from one place to another one by rolling the wheel in 3D. In this section, the balancing motion of an inverted double pendulum robot on a rolling contact is considered. Although both ballbot and a curved-foot balancer are rolling-contact balancing robots, they are different in terms of their dimensions, DoF, number of links, etc.

Similar to the 2D balancer, a curved-foot 2D balancer consists of two links con-

nected by an actuated revolute joint. The difference is that the lower body of a curved-foot balancer contains a surface called the foot, which makes contact with the ground at a single point. Fig. 3.14 shows a curved-foot robot model and its parameters. In this figure, q_1 is the angle between the lower body and the horizontal axis (ground), and x and y are the horizontal and vertical locations of point P which is the connection point between the lower body and the foot. As is shown in Fig. 3.14, the length of the lower body (l_1) and the location of its CoM (l_{c1}) are measured from point P. The location of the contact point with respect to the reference coordinate frame is denoted by s. The origin of the reference frame (see Fig. 3.14) is a point on the ground which coincides with P when the lower body is vertical.

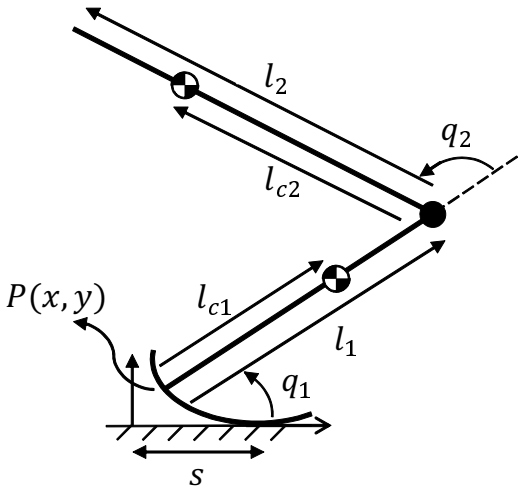


Figure 3.14: Models and parameters of a curved-foot 2D balancer robot

It is assumed that the ground is flat and horizontal, and that there is no slipping or loss of contact between the foot and the ground. So the equations of motion for the curved-foot 2D balancer robot in Fig. 3.14 are

$$\begin{aligned} f_x \;=\; & c\ddot{x} - (c_4\sin(q_1) + c_5\sin(q_1+q_2))\ddot{q}_1 - c_5\sin(q_1+q_2)\ddot{q}_2 \\ & - c_4\cos(q_1)\dot{q}_1^2 - c_5\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2)^2\,, \end{aligned} \tag{3.38}$$

$$\begin{aligned} f_y \;=\; & c\ddot{y} + (c_4\cos(q_1) + c_5\cos(q_1+q_2))\ddot{q}_1 + c_5\cos(q_1+q_2)\ddot{q}_2 \\ & - c_4\sin(q_1)\dot{q}_1^2 - c_5\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2)^2 + cg\,, \end{aligned} \tag{3.39}$$

$$\begin{aligned} f_x y + f_y(s-x) \;=\; & -(c_4\sin(q_1) + c_5\sin(q_1+q_2))\ddot{x} \\ & + (c_4\cos(q_1) + c_5\cos(q_1+q_2))\ddot{y} + (c_1 + c_2 + 2c_3\cos(q_2))\ddot{q}_1 \\ & + (c_2 + c_3\cos(q_2))\ddot{q}_2 - 2c_3\sin(q_2)\dot{q}_1\dot{q}_2 \\ & - c_3\sin(q_2)\dot{q}_2^2 + c_4 g\cos(q_1) + c_5 g\cos(q_1+q_2)\,, \end{aligned} \tag{3.40}$$

$$\begin{aligned}\tau \;=\; & -c_5\sin(q_1+q_2)\ddot{x}+c_5\cos(q_1+q_2)\ddot{y}+(c_2+c_3\cos(q_2))\ddot{q}_1+c_2\ddot{q}_2\\ & +c_3\sin(q_2)\dot{q}_1^2+c_5 g\cos(q_1+q_2)\,,\end{aligned} \tag{3.41}$$

where f_x and f_y are the ground reaction forces exerted to the robot via its contact point (i.e. the point $(s,0)$). The left hand side of (3.40) is in fact the moment of the contact force about P. Since the contact between the robot's foot and the ground is modelled as a rolling-contact joint with no slipping or loss of contact, then x, y and s are always dependent on the joint variable which is q_1. Therefore, based on the curve shape of the foot, there always exist relationships which express x, y and s as functions of q_1. Using these relationships, it is possible to express $\ddot{x}$ and $\ddot{y}$ in terms of q_1 and its derivatives, and consequently they can be eliminated from (3.38) and (3.39). Hence, f_x and f_y can be written in terms of q_1, q_2 and their derivatives. Having expressed f_x, f_y and s in terms of the joint angles and their derivatives, (3.40) and (3.41) reduce to two equations with two unknowns ($\ddot{q}_1$, $\ddot{q}_2$) where τ is the control torque.

Since the contact point is not a fixed point then (3.5) and (3.6) are not valid which means that the control law (3.11) and the alternative control law (3.12) are not equivalent for a curved-foot balancer. To control the balancing motion on a rolling contact, the alternative control law (3.12) is used, replacing L, X and $\dot{X}$ by L_c, X_c and $\dot{X}_c$, respectively. The resulting control law is

$$\tau = -k_v\dot{X}_c - k_x X_c + k_p L_c + k_s(q_2^d - q_2) + \tau_g\,, \tag{3.42}$$

where L_c is the angular momentum of the robot about its contact point, X_c is the horizontal location of the CoM relative to the contact point and $\dot{X}_c$ is the horizontal velocity of the CoM relative to the contact point. Note that the radius of curvature of the foot at the contact point is, in general, only a piecewise-continuous function of q_1 and if it changes discontinuously at some values of q_1. Then the velocity of the contact point also changes discontinuously, meaning that $\dot{X}_c$ is also capable of changing discontinuously.

The angular momentum of the robot about its contact point is

$$L_c = m_1(\vec{r}_1^{\,c}\times\dot{\vec{r}}_1^{\,c}) + m_2(\vec{r}_2^{\,c}\times\dot{\vec{r}}_2^{\,c}) + I_1\dot{q}_1 + I_2(\dot{q}_1+\dot{q}_2)\,, \tag{3.43}$$

where $\vec{r}_1^{\,c}$ and $\vec{r}_2^{\,c}$ are the vectors pointing to the CoM of the links from the contact point. If $\hat{i}$ and $\hat{j}$ denote unit vectors in the horizontal and vertical directions, then

$$\vec{r}_1^{\,c} = (x - s + l_{c1}\cos(q_1))\hat{i} + (y + l_{c1}\sin(q_1))\hat{j}\,, \tag{3.44}$$

$$\begin{aligned}\vec{r}_2^{\,c} \;=\; & (x - s + l_1\cos(q_1) + l_{c2}\cos(q_1+q_2))\hat{i}\\ & +(y + l_1\sin(q_1) + l_{c2}\sin(q_1+q_2))\hat{j}\,.\end{aligned} \tag{3.45}$$

Hence

$$\dot{\vec{r}}_1^{\,c} = (\dot{x} - \dot{s} - l_{c1}\sin(q_1)\dot{q}_1)\hat{i} + (\dot{y} + l_{c1}\cos(q_1)\dot{q}_1)\hat{j}\,, \tag{3.46}$$

$$\begin{aligned}\dot{\vec{r}}_2^c &= (\dot{x} - \dot{s} - l_1\sin(q_1)\dot{q}_1 - l_{c2}\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2))\hat{i} \\ &\quad + (\dot{y} + l_1\cos(q_1)\dot{q}_1 + l_{c2}\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2))\hat{j}\,. \end{aligned} \tag{3.47}$$

By replacing (3.44), (3.45), (3.46) and (3.47) into (3.43) we have

$$\begin{aligned} L_c &= m_1(x + l_{c1}\cos(q_1))(\dot{y} + l_{c1}\cos(q_1)\dot{q}_1) - m_1(y + l_{c1}\sin(q_1))(\dot{x} - l_{c1}\sin(q_1)\dot{q}_1) \\ &\quad + m_2(x + l_1\cos(q_1) + l_{c2}\cos(q_1+q_2))(\dot{y} + l_1\cos(q_1)\dot{q}_1 + l_{c2}\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2)) \\ &\quad - m_2(y + l_1\sin(q_1) + l_{c2}\sin(q_1+q_2))(\dot{x} - l_1\sin(q_1)\dot{q}_1 - l_{c2}\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2)) \\ &\quad + m_1\dot{s}(y + l_{c1}\sin(q_1)) - m_1 s(\dot{y} + l_{c1}\cos(q_1)\dot{q}_1) \\ &\quad + m_2\dot{s}(y + l_1\sin(q_1) + l_{c2}\sin(q_1+q_2)) \\ &\quad - m_2 s(\dot{y} + l_1\cos(q_1)\dot{q}_1 + l_{c2}\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2)) + I_1\dot{q}_1 + I_2(\dot{q}_1+\dot{q}_2)\,. \end{aligned} \tag{3.48}$$

Simplifying the above equation, the angular momentum of the robot about its contact point is

$$\begin{aligned} L_c &= x(c\dot{y} + c_4\cos(q_1)\dot{q}_1 + c_5\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2)) \\ &\quad + y(-c\dot{x} + c_4\sin(q_1)\dot{q}_1 + c_5\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2)) \\ &\quad - \dot{x}(c_4\sin(q_1) + c_5\sin(q_1+q_2)) + \dot{y}(c_4\cos(q_1) + c_5\cos(q_1+q_2)) \\ &\quad + \dot{q}_1(c1 + c_2 + 2c_3\cos(q_2)) + \dot{q}_2(c_2 + c_3\cos(q_2)) \\ &\quad - s(c\dot{y} + c_4\cos(q_1)\dot{q}_1 + c_5\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2)) \\ &\quad + \dot{s}(cy + c_4\sin(q_1) + c_5\sin(q_1+q_2))\,. \end{aligned} \tag{3.49}$$

According to [Featherstone, 2012], the shape of the foot (i.e. its radius of curvature at the contact point) can substantially influence the balancing performance of a curved-foot robot. To investigate the performance of the proposed control algorithm in balancing robots with different foot shapes, two foot shapes are considered in this section: 1) a circular arc which has a constant radius of curvature at the contact point, and 2) a general convex curve. The former is called wheelfoot balancer and the latter is called camfoot balancer in this section. Note that no effort has been made to optimize the inertia parameters of the robots, radius of circular arc of the wheelfoot or shape of the camfoot.

3.8.1 Wheelfoot 2D Balancer

The wheelfoot 2D balancer is a special case of curved-foot 2D balancers in which the foot is a circular arc with a constant radius of R. So for the wheelfoot balancer we have

$$s = R(\frac{\pi}{2} - q_1) \implies \dot{s} = -R\dot{q}_1\,, \tag{3.50}$$

$$x = s - R\cos q_1 \implies \dot{x} = -R\dot{q}_1 + R\sin(q_1)\dot{q}_1\,, \tag{3.51}$$

$$y = R - R\sin(q_1) \implies \dot{y} = -R\cos(q_1)\dot{q}_1\,. \tag{3.52}$$

Replacing (3.50), (3.51) and (3.52) into (3.49) will give us the angular momentum of the wheelfoot 2D balancer about the contact point as

$$\begin{aligned} L_c \;=\; & (c_1 + c_2 + 2c_3\cos(q_2))\dot{q}_1 + (c_2 + c3\cos(q_2))\dot{q}_2 \\ & - 2Rc_4\dot{q}_1 + R(c_4\sin(q_1)\dot{q}_1 + c_5\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2)) \\ & + cR^2(1-\sin(q_1))\dot{q}_1 - Rc_5\cos(q_2)(2\dot{q}_1+\dot{q}_2)\,. \end{aligned} \tag{3.53}$$

Also using (3.50), (3.51) and (3.52), the motion equations can be simplified to

$$\begin{aligned} 0 \;=\; & \ddot{q}_1(c_1 + c_2 + 2c_3\cos(q_2)) + \ddot{q}_2(c_2 + c_3\cos(q_2)) + Rc_5\sin(q_2)(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2) \\ & + \ddot{q}_1(cR(1-\sin(q_1)) + c_4\sin(q_1) + c_5\sin(q_1+q_2) - c_4 - c_5\cos(q_2))2R \\ & + \ddot{q}_2(\sin(q_1+q_2) - \cos(q_2))Rc_5 - c_3\sin(q_2)(2\dot{q}_1+\dot{q}_2)\dot{q}_2 \\ & - cR^2\cos(q_1)\dot{q}_1^2 + Rc_4\cos(q_1)\dot{q}_1^2 + Rc_5\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2)^2 \\ & + g(c_4\cos(q_1) + c_5\cos(q_1+q_2) - cR\cos(q_1))\,, \end{aligned} \tag{3.54}$$

$$\begin{aligned} \tau \;=\; & \ddot{q}_1(\sin(q_1+q_2) - \cos(q_2))Rc_5 + (c_2 + c_3\cos(q_2))\ddot{q}_1 + c_2\ddot{q}_2 \\ & - Rc_5\sin(q_2)\dot{q}_1^2 + c_3\sin(q_2)\dot{q}_1^2 + c_5 g\cos(q_1+q_2)\,. \end{aligned} \tag{3.55}$$

Simulations have been conducted for straightening and crouching motions of a wheelfoot balancer where the radius of its foot is 5cm. The parameters of the wheelfoot balancer are the same as the simulator's model of the 2D balancer listed in Table 3.1. To derive the characteristic equation, the motion equations (3.54) and (3.55) are linearized about the target position using the method described in 3.3. Thus the characteristic equation at the desired configuration for the straightening motion ($q_2^d = 0$) is

$$\begin{aligned} 0 \;=\; & \lambda^4 + (5.42433k_v - 0.73795k_p)\lambda^3 + (5.42433k_x - 28.72984 + 187.24507k_s)\lambda^2 \\ & + (39.74559k_p)\lambda + (-2701.98917k_s)\,, \end{aligned} \tag{3.56}$$

and for the crouching motion ($q_2^d = -\frac{\pi}{2}$) is

$$\begin{aligned} 0 \;=\; & \lambda^4 + (2.98359k_v - 0.20973k_p)\lambda^3 + (2.98359k_x - 24.73253 + 71.01911k_s)\lambda^2 \\ & + (19.21326k_p)\lambda + (-1226.96263k_s)\,. \end{aligned} \tag{3.57}$$

In both cases, the gains are computed using the method described in section 3.4, placing all the poles of the closed-loop system at -7.

Figure 3.15 shows the straightening motion of the wheelfoot balancer moving from its initial crouched position ($q_2 = -\frac{\pi}{2}$) and with zero initial velocity. The initial

value of q_1 is calculated for perfect balance given $q_2 = -\frac{\pi}{2}$. Although this motion is very similar to the straightening motion of the 2D balancer in Fig. 3.2, it is noticeably slower. Of course it is possible to produce quicker straightening motions by choosing more negative values for the location of the poles at the cost of a larger overshoot.

Figure 3.16 shows the corresponding crouching motion, in which the wheelfoot balancer starts in a balanced upright position ($q_2 = 0$) and moves to a balanced crouching position ($q_2 = -\frac{\pi}{2}$). Comparing this motion with the crouching motion of the 2D balancer (Fig. 3.3), it can be seen that the initial movement of the wheelfoot balancer in the opposite direction is much larger (almost 2.5 times). However, the wheelfoot balancer converges to the desired position ($q_2^d = -\frac{\pi}{2}$) a bit quicker than the 2D balancer. During the crouching motion of the wheelfoot balancer, q_1 reaches its desired final value almost 0.5s before q_2 does. It implies that the physics of balancing does not require q_1 and q_2 to reach their final positions, simultaneously.

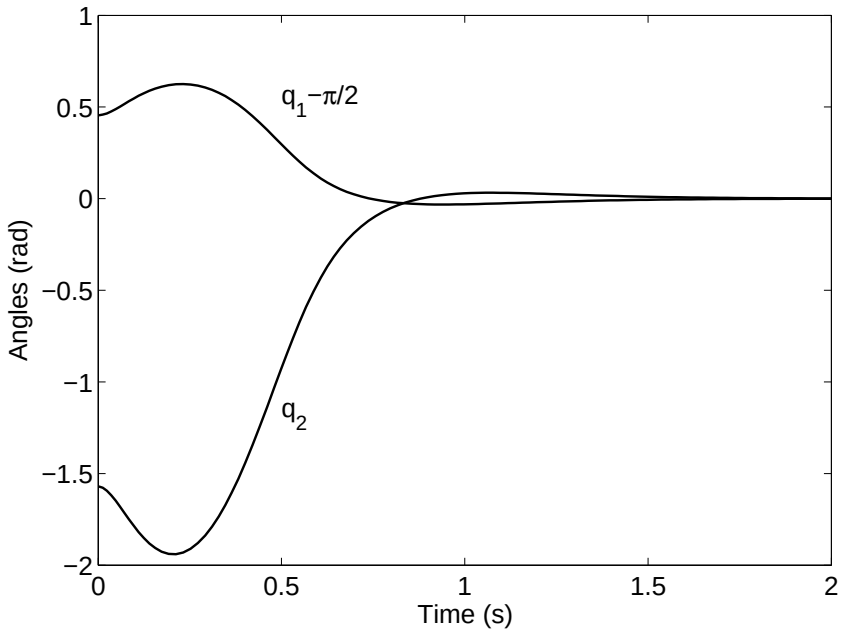


Figure 3.15: straightening motion of the wheelfoot balancer

3.8.2 Camfoot 2D Balancer

For the curve of the camfoot balancer the clothoid function (or Euler spiral) is used because of its interesting and useful property that the curvature (κ) of the clothoid varies linearly with its variable parameter (θ):

$$\kappa = \frac{\theta}{\rho},$$

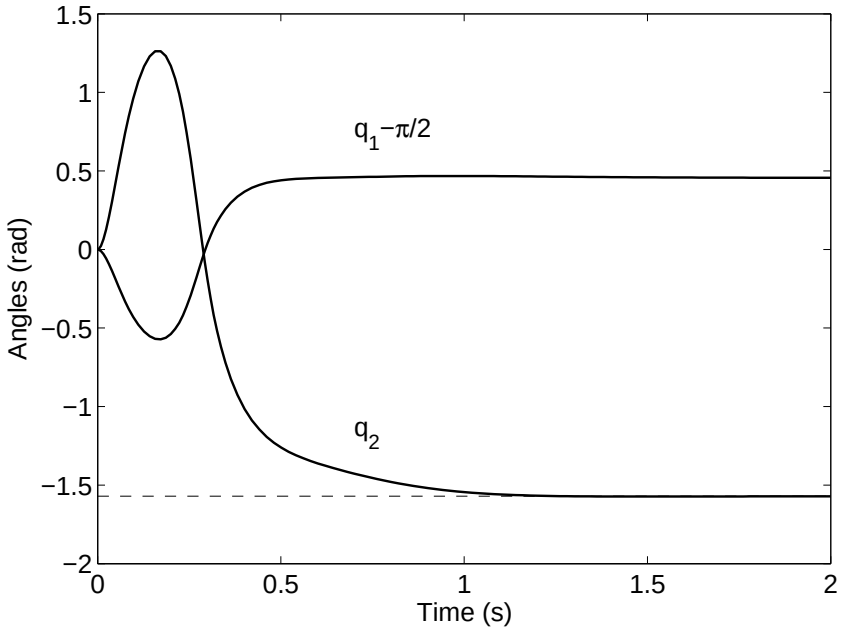


Figure 3.16: Crouching motion of the wheelfoot balancer

where ρ is a constant parameter. The equations of the clothoid are

$$u = \rho\sqrt{\pi}\ \text{FresnelS}(\frac{\theta}{\sqrt{\pi}})\,, \tag{3.58}$$

$$w = \rho\sqrt{\pi}\ \text{FresnelC}(\frac{\theta}{\sqrt{\pi}})\,, \tag{3.59}$$

where

$$\text{FresnelS}(\frac{\theta}{\sqrt{\pi}}) = \int_0^{\theta/\sqrt{\pi}} \sin(\frac{\pi\xi^2}{2})\mathrm{d}\xi\,, \tag{3.60}$$

$$\text{FresnelC}(\frac{\theta}{\sqrt{\pi}}) = \int_0^{\theta/\sqrt{\pi}} \cos(\frac{\pi\xi^2}{2})\mathrm{d}\xi\,. \tag{3.61}$$

Fig. 3.17 shows an example of a clothoid where θ varies between -10 and 10 and ρ is 0.1.

The foot of the camfoot balancer is composed of a clothoid curve segment and its mirror image, the two being joined to form a shield shape with a sharp point at the bottom (see Fig. 3.18). The curvature of the clothoid is 15 immediately adjacent to the sharp point, and increases gradually with increasing the distance from the point. The value of ρ is set to 0.1 implying that the starting value for θ is 1.5. The radius of curvature of the foot at the contact point is plotted against its corresponding value of q_1 in Fig. 3.19. According to this figure, there are step changes in the radius of curvature to both sides of $q_1 = \frac{\pi}{2}$ which will affect the controller's performance. Including these discontinuities, it will be possible to observe how the controller deals with a

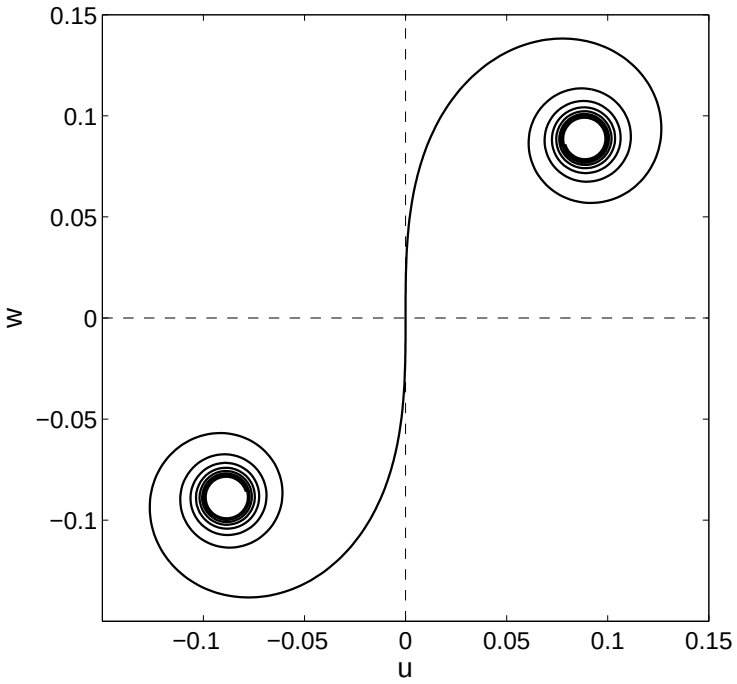


Figure 3.17: A clothoid plot with $\rho = 0.1$

step change in velocity due to the step change in radius of curvature. Therefore, the balancing motion of the camfoot is expected to be different from the wheelfoot due to 1) variable radius of curvature, and 2) the step change in the radius of curvature.

The kinematic and inertia parameters that are used in simulations for the camfoot balancer are the same as for the wheelfoot balancer and the 2D balancer. In an upright configuration, the camfoot is on its sharp point so the characteristic equation for its straightening motion and consequently the controller's gains are the same as they are for the 2D balancer. In a crouched configuration, the camfoot might either pivot on its sharp point or roll on the clothoid depending on q_2^d. Note that characteristic equations of the camfoot balancer and the 2D balancer are the same, as long as the contact point at the desired configuration is the sharp point of the foot. The desired value of q_2 for the crouching motion in this section is $-\frac{\pi}{2}$. Since the contact point at $q_2^d = -\frac{\pi}{2}$ is on the clothoid, the motion equations of the camfoot balancer are linearized about that point to obtain the characteristic equation as

$$\begin{aligned} 0 \;=\; & \lambda^4 + (3.56652k_v - 0.12747k_p)\lambda^3 + (2.63407k_x - 22.1207 + 68.0053k_s)\lambda^2 \\ & + (16.3538k_p - 6.21586k_v)\lambda + (-1065.42k_s)\,. \end{aligned} \tag{3.62}$$

Again the poles of the closed-loop system are placed at -7 and the gains are computed using the method described in section 3.4.

Figure 3.20 shows the straightening motion of the camfoot balancer. The robot starts from an initial balanced position ($q_2 = -\frac{\pi}{2}$), with its contact point on the

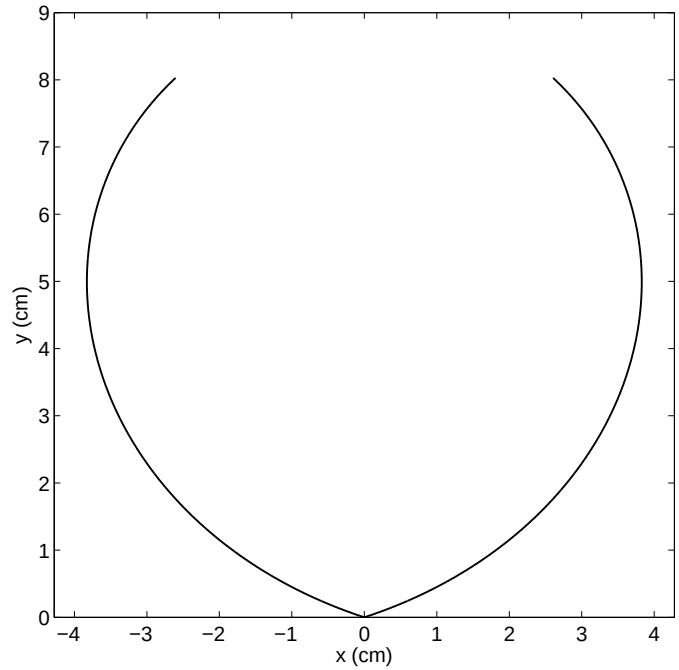


Figure 3.18: The shape of the foot of the camfoot balancer

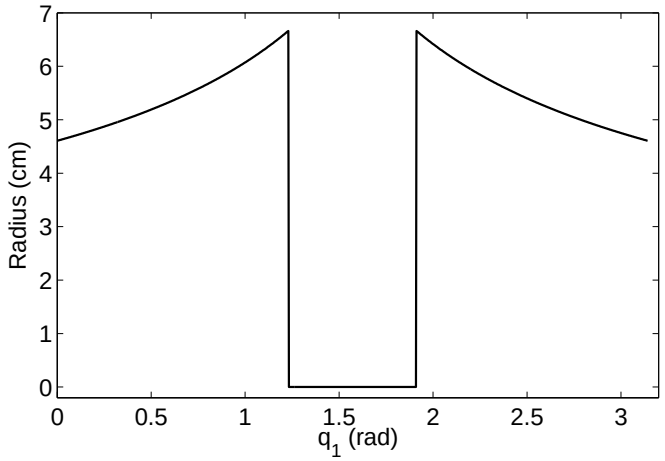


Figure 3.19: Radius of curvature of the camfoot balancer at the contact point

clothoid, rolls on its curved-foot and balances itself on the sharp point of the foot at the desired configuration ($q_2^d = 0$). Instants of transitions between rolling and pivotting motions are more clear in Fig. 3.21 which shows $\dot{s}$ during the motion. The robot is pivotting when $\dot{s} = 0$ and it is rolling at the other times. It can be seen in Fig. 3.21 that there are two rolling motions during the straightening motion. In fact, the robot reaches the desired configuration within the first 0.5s of the motion (see Fig. 3.20) but because of the high speed ($\dot{s} > 0.55\frac{\text{m}}{\text{s}}$), it misses the desired configuration and consequently rolls on the opposite side. Large overshoots of the joint angles in Fig. 3.20, between $t = 0.5$s and $t = 1$s, are due to the sudden change from rolling to pivotting. This sudden change, when the robot's momentum is quite high, makes overshoots unavoidable.

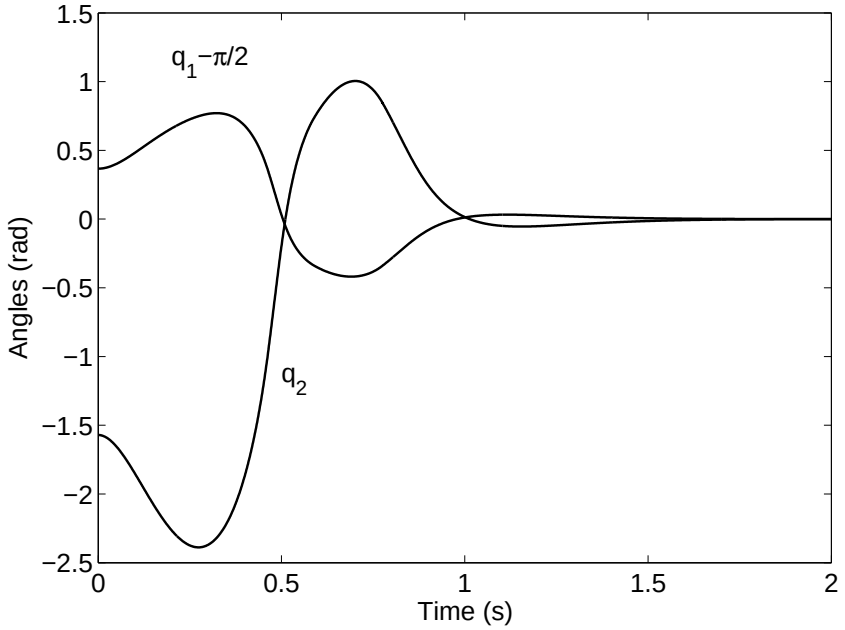


Figure 3.20: Straightening motion of the camfoot balancer

The joint angles during the crouching motion of the camfoot balancer are shown in Fig. 3.22. Sudden slope changes in q_1 and q_2 curves at about $t = 0.8$s, which is more obvious in $\dot{s}$ graph in Fig. 3.23, shows a transition from pivotting to rolling motion. There is only one rolling motion during the crouching motion. The robot slowly rolls on the clothoid and then balance itself at the desired crouched configuration ($q_2^d = -\frac{\pi}{2}$).

3.9 Summary

In this chapter, a new control algorithm based on angular momentum is presented for balancing an under-actuated planar robot. The controller is able to stabilize the robot in any unstable balanced configuration in which the robot is controllable, and

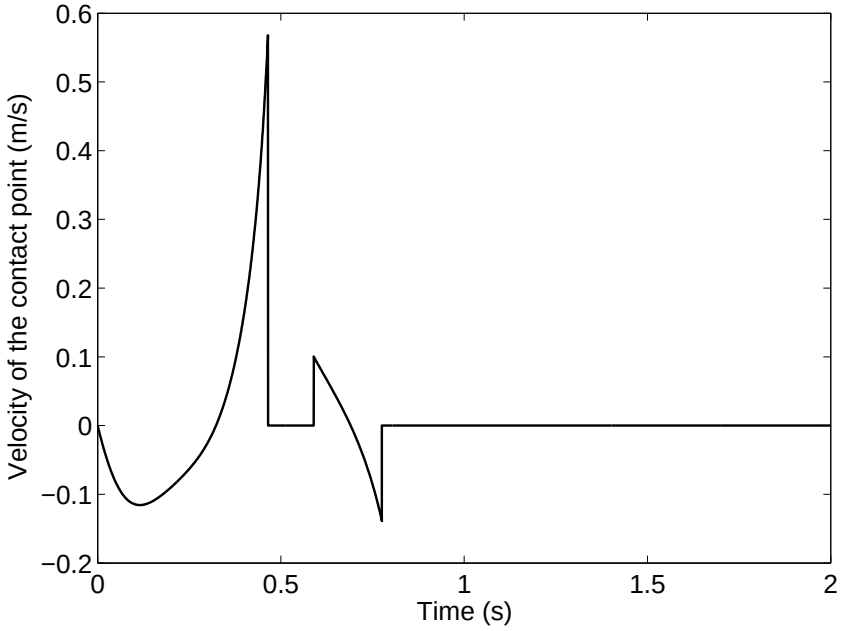


Figure 3.21: Velocity of the contact point ($\dot{s}$) during the straightening motion of the camfoot balancer

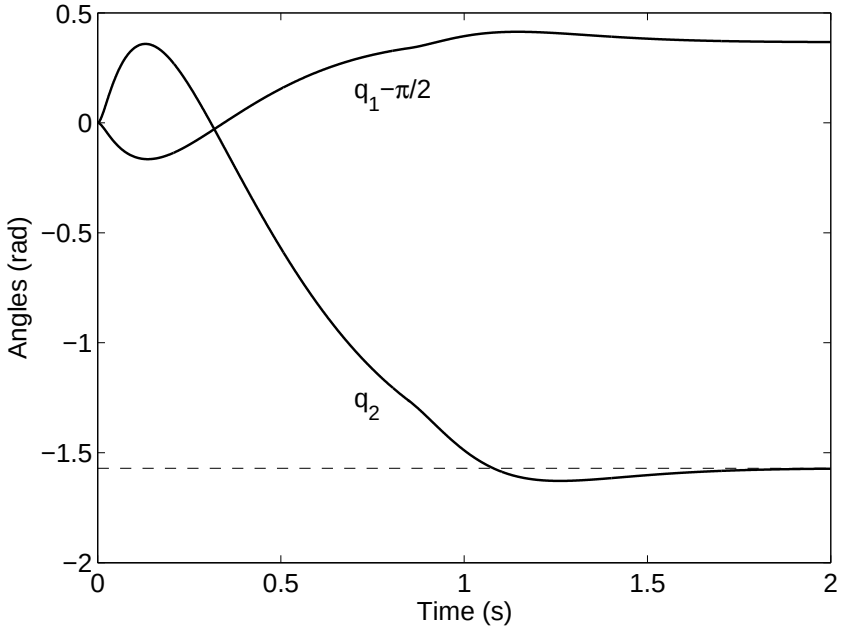


Figure 3.22: Crouching motion of the camfoot balancer

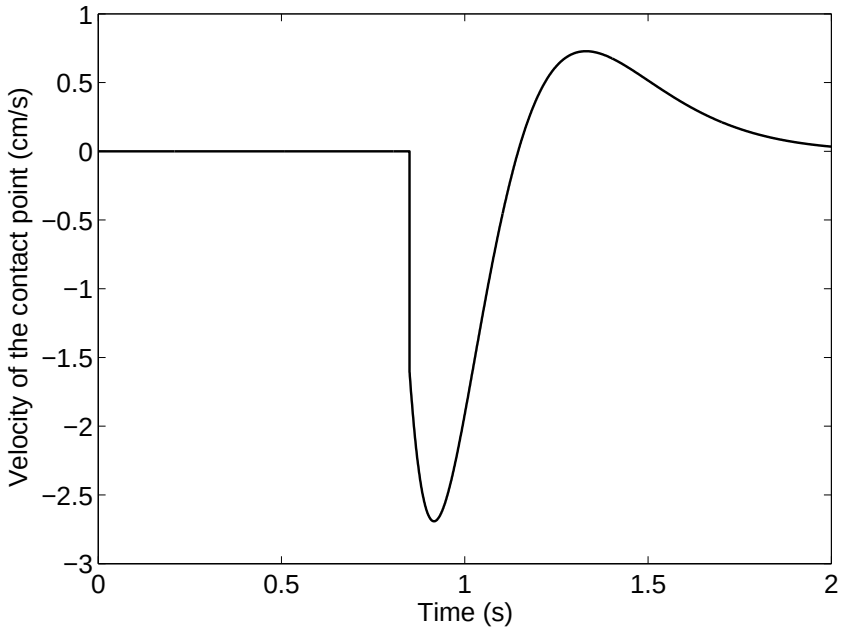


Figure 3.23: Velocity of the contact point ($\dot{s}$) during the crouching motion of the camfoot balancer

is also able to follow a class of arbitrary trajectories without losing balance. Simulation results show the good performance of the controller in balancing and trajectory tracking motions of the robot. The simulations also show that the proposed controller is robust to significant imperfections in the system, such as errors in the controller's dynamic model of the robot and imperfections in the sensors and actuators. The new controller is compared with three existing balancing controllers and is shown to outperform them. Also it is shown in simulations that the new controller is able to balance a planar under-actuated robot with an arbitrary shape of the foot on a rolling contact. Simulation results of the balancing motion of the camfoot balancer show the capability of the controller in dealing with sudden changes in the curve of the foot (radius of curvature of the contact point).

Chapter 4

Hopping Motion of an Under-actuated Planar Robot

This chapter considers the problem of hopping motion control of a knee leg robot with one actuator. This robot is called "2D hopper" in this chapter. The 2D hopper is similar to the acrobot which is used in [Berkemeier and Fearing, 1998]. Berkemeier and Fearing were first to study hopping in a knee leg hopper with only one actuator. Their proposed controller is briefly explained in subsection 1.1.1.2. The main difference between their controller and the one that is introduced in this chapter is in the flight phase control algorithm. In Berkemeier and Fearing's controller the robot rotates its leg during the flight phase and lands in the same configuration as it takes off the ground. However, the objective of the flight phase in the new controller is to control the hop length and land the robot in any configuration from which it can recover its balance.

Unlike previous studies on hopping robots, this chapter does not consider continuous hopping motion. Instead a single-hop motion is considered which consists of a crouch-and-launch phase, a flight phase, a landing, and a balance recovery phase. During a single-hop motion the robot starts and ends both in an upright balanced configuration. The balancing controller which was already introduced in chapter 3 is used to balance the robot during the balance recovery phase. This controller is also extended to conduct trajectory-tracking performance which is used to implement the crouch-and-launch and flight phases. During the crouch-and-launch phase, the robot is commanded to follow a specified trajectory to intentionally lose its balance and be prepared for the hop. During the flight phase, the horizontal location of the foot is controlled by using the trajectory-tracking controller.

The performance of the proposed control algorithm during a complete hop is demonstrated by simulation. Also, it will be shown in simulation that the controller is able to cope with slipping between the foot and the ground, which can happen during crouching, at lift-off, and after landing. Most of the results presented in this chapter are published in [Azad and Featherstone, 2013].

4.1 Robot Model and Motion Equations

The mechanism of the 2D hopper consists of two links connected to each other by an actuated revolute joint. Fig. 4.1 shows a schematic diagram of the 2D hopper. It is

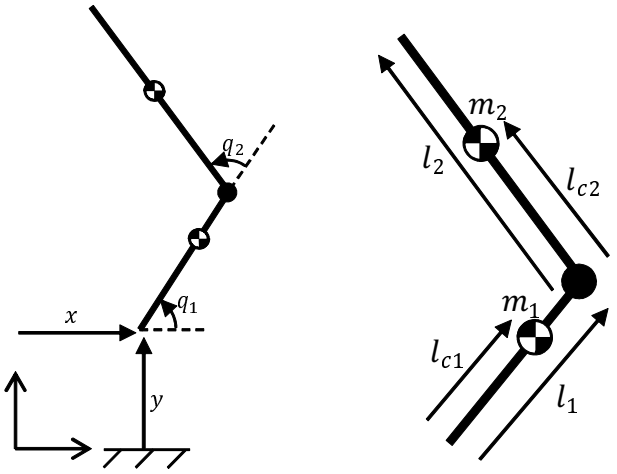


Figure 4.1: Planar hopper model, (a) generalized coordinates, (b) parameters

assumed that the tip of the lower leg (which is called the foot) is the only point that can make contact with the ground. The motion equations for this robot are [1]:

$$\begin{aligned} f_x \;=\; & c\ddot{x} - (c_4\sin(q_1) + c_5\sin(q_1+q_2))\ddot{q}_1 - c_5\sin(q_1+q_2)\ddot{q}_2 \\ & - c_4\cos(q_1)\dot{q}_1^2 - c_5\cos(q_1+q_2)(\dot{q}_1+\dot{q}_2)^2 \end{aligned} \tag{4.1}$$

$$\begin{aligned} f_y \;=\; & c\ddot{y} + (c_4\cos(q_1) + c_5\cos(q_1+q_2))\ddot{q}_1 + c_5\cos(q_1+q_2)\ddot{q}_2 \\ & - c_4\sin(q_1)\dot{q}_1^2 - c_5\sin(q_1+q_2)(\dot{q}_1+\dot{q}_2)^2 + cg \end{aligned} \tag{4.2}$$

$$\begin{aligned} 0 \;=\; & -(c_4\sin(q_1) + c_5\sin(q_1+q_2))\ddot{x} + (c_4\cos(q_1) + c_5\cos(q_1+q_2))\ddot{y} \\ & + (c_1 + c_2 + 2c_3\cos(q_2))\ddot{q}_1 + (c_2 + c_3\cos(q_2))\ddot{q}_2 - 2c_3\sin(q_2)\dot{q}_1\dot{q}_2 \\ & - c_3\sin(q_2)\dot{q}_2^2 + c_4 g\cos(q_1) + c_5 g\cos(q_1+q_2) \end{aligned} \tag{4.3}$$

$$\begin{aligned} \tau \;=\; & -c_5\sin(q_1+q_2)\ddot{x} + c_5\cos(q_1+q_2)\ddot{y} + (c_2 + c_3\cos(q_2))\ddot{q}_1 + c_2\ddot{q}_2 \\ & + c_3\sin(q_2)\dot{q}_1^2 + c_5 g\cos(q_1+q_2) \end{aligned} \tag{4.4}$$

where I_1 and I_2 are the moments of inertia of the links about their CoM, f_x and f_y are the contact forces exerted from the ground to the foot, g is the acceleration of gravity,

[1] Equations (4.1)-(4.4) are the same as the motion equations of a curved-foot balancer (3.38)-(3.41) when the moment of the contact force about P is zero.

and

$$\begin{aligned} c &= m_1 + m_2\,, & c_1 &= m_1 l_{c1}^2 + m_2 l_1^2 + I_1\,, \\ c_2 &= m_2 l_{c2}^2 + I_2\,, & c_3 &= m_2 l_1 l_{c2}\,, \\ c_4 &= m_1 l_{c1} + m_2 l_1\,, & c_5 &= m_2 l_{c2}\,. \end{aligned}$$

When the foot is in contact with the ground (the robot is in its stance phase) and it is not sliding, it is possible to consider the robot as a two degrees of freedom balancer robot. By neglecting the values of x and y in (4.1)-(4.4), the motion equations for the hopper during the stance phase are:

$$\begin{aligned} 0 \;=\; & (c_1 + c_2 + 2c_3\cos(q_2))\ddot{q}_1 + (c_2 + c_3\cos(q_2))\ddot{q}_2 - 2c_3\sin(q_2)\dot{q}_1\dot{q}_2 \\ & - c_3\sin(q_2)\dot{q}_2^2 + c_4 g\cos(q_1) + c_5 g\cos(q_1 + q_2)\,, \end{aligned} \tag{4.5}$$

$$\tau = (c_2 + c_3\cos(q_2))\ddot{q}_1 + c_2\ddot{q}_2 + c_3\sin(q_2)\dot{q}_1^2 + c_5 g\cos(q_1 + q_2)\,, \tag{4.6}$$

which are the same as the motion equations for the 2D balancer in chapter 3.

The parameters for the 2D hopper that have been used in the simulations in this chapter are (cf. the acrobot's parameters in [Berkemeier and Fearing, 1998]):

$$m_1 = 2\text{kg}\,, \quad m_2 = 14\text{kg}\,, \quad l_1 = 0.5\text{m}\,, \quad l_2 = 0.75\text{m}$$

$$I_1 = l_{c1} = 0\,, \quad l_{c2} = 0.375\,, \quad I_2 = m_2 l_{c2}^2\,.$$

Contact between the robot's foot and the ground is modelled as a nonlinear compliant contact with Coulomb friction, using a planar version of the 3D contact model described in chapter 2. According to (2.22) and (2.30), the forces acting on the foot in the normal and tangent directions are

$$f_y = \begin{cases} 0 & \text{if } y \geq 0 \\ \max(0, -\sqrt{-y}\,(K_n y + D_n \dot{y})) & \text{if } y < 0 \end{cases} \tag{4.7}$$

and

$$f_x = \begin{cases} \mu f_y \cdot \frac{f_t}{|f_t|} & \text{if } |f_t| > \mu f_y \\ f_t & \text{otherwise} \end{cases}\,, \quad f_t = \begin{cases} 0 & \text{if } y \geq 0 \\ -\sqrt{-y}(K_t u + D_t \dot{x}) & \text{if } y < 0 \end{cases}\,, \tag{4.8}$$

where K_n and D_n are the normal and K_t and D_t are the tangential stiffness and damping coefficients, respectively, u is the ground shear deformation and μ is the coefficient of friction. The parameter values used in the simulations are

$$\begin{aligned} K_n &= 8.5 \times 10^6\,, \quad K_t = 12.75 \times 10^6\,, \\ D_n &= D_t = 3.1 \times 10^5\,, \quad \mu = 0.4\,. \end{aligned}$$

4.2 Control Strategies

As already mentioned, this chapter considers a single-hop instead of continuous hopping motion and therefore studies a hopping gait in which the robot starts from an upright balanced configuration and ends in the same balanced configuration. The motion phases of such a hopping gait, after starting from a balanced configuration, are: launching (from the beginning to the instant of take-off), flight, and landing and balancing again at the end. The landing phase starts from the moment the foot touches the ground, and includes a short period when the foot is sliding, which will physically happen in a real robot. Since the controller for balancing the robot that was proposed in chapter 3 is able to balance the robot from the beginning of the landing phase without any further setup, the hopping motion is divided into three phases: launching, flight and balancing.

To control the robot's motion during the balancing phase, the balancing controller described in chapter 3 is used. For the other two phases, a modified version of the balancing controller is used which is able to track a prescribed trajectory. Before explaining the control strategies of the launching and flight phases, the trajectory-tracking controller that is used during these two phases is introduced in the following subsection.

4.2.1 Trajectory-tracking Controller

For the purpose of trajectory tracking, the control law (3.8) is modified as

$$\tau = -k_v(\dot{X} - \dot{X}^d) - k_x(X - X^d) + k_p(L - L^d)\,, \tag{4.9}$$

where X is the horizontal location of the CoM with respect to the foot, L is the angular momentum of the robot about its foot, L^d is the desired value of L and X^d is the desired value of X within the desired trajectory. When the robot is in its stance phase (without slipping), it is possible to calculate L^d from X^d using (3.5) as follows:

$$\dot{L}^d = -cgX^d \implies L^d = -cg\int X^d \mathrm{d}t\,.$$

For example, if the desired trajectory for X is a sine-wave function of time, such as $X^d = A\sin(\omega t)$, where A and ω are constants (with some limits on their values), then the controller is able to track this trajectory without losing balance. Fig. 4.2 shows an example of tracking a sine-wave trajectory for X. In this example L^d is chosen to be

$$L^d = \rho\cos(\omega t)\,,$$

where $\rho = \pi/2$ and $\omega = \pi$. Therefore,

$$\dot{L}^d = -\rho\omega\sin(\omega t) = -cgX^d \implies X^d = A\sin(\omega t)\,,$$

where $A = \frac{\rho\omega}{cg}$. Since the robot (which is under-actuated) is physically unable to fol-

low this trajectory exactly while keeping its balance, it follows the nearest physically possible trajectory instead. This is the reason for the relatively large tracking errors shown in Fig. 4.2.

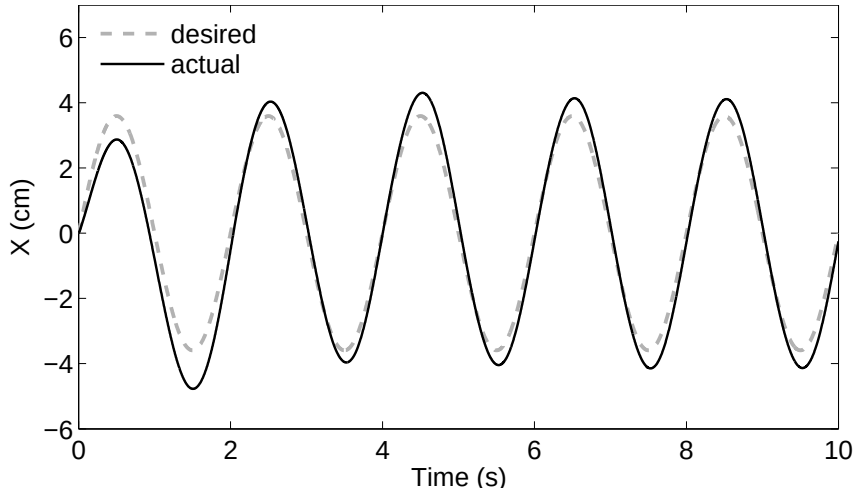


Figure 4.2: Performance of the trajectory-tracking controller in following a sine-wave function

The trajectory-tracking controller is designed to be used in the launching and flight phases of the hopping motion of the 2D hopper. This controller is different from the one that is described in section 3.5. The main difference is that the controller described in this section is designed to follow a prescribed trajectory for X whereas the controller described in section 3.5 follows a desired trajectory for q_2. Therefore the control laws are different. However, in both cases the commanded trajectories may be physically impossible to be exactly followed while the robot is maintaining its balance.

4.2.2 Launching Phase

In the launching phase, the objective is to move the robot from its initial balanced position to a desired dynamic state which is called the launching state in this chapter. The launching state is an unstable configuration at the end of the launching phase in which the robot has positive velocity in both the horizontal and vertical directions and is ready to hop. To implement the launching phase, the robot must be able to deliberately lose its balance and pass through a predetermined unstable configuration for an instant which is the take-off instant. In order to move the robot from its initial state at the balanced position to the desired state at the launching configuration, the trajectory-tracking controller, which is introduced in subsection 4.2.1, is used.

The trajectory-tracking controller in (4.9) uses the angular momentum about the foot (L) and the horizontal displacement and velocity of the CoM with respect to the foot (X and $\dot{X}$). To define a trajectory between the initial and final states of the launching phase, it is hence necessary to map the state variables to L, X and $\dot{X}$ and then find a suitable trajectory between their initial and final values.

A time-reversal technique is employed to find a suitable reference trajectory for the launching phase. This technique works as follows. First, the 2D hopper's initial conditions are set to the desired lift-off state (the launching state) but with the reversed launch velocities. Then the balancing controller (3.11) is applied to the robot to bring it to its upright balanced configuration. The resulting trajectory, when time-reversed, is a suitable reference trajectory for the launching phase. Note that this trajectory is not the only one that can be used in the launching phase but is straight-forward to calculate. In general, any trajectory between the upright balanced configuration and the launching state can serve as the reference trajectory if the relationships between L^d, X^d and $\dot{X}^d$ are valid within the trajectory.

Fig. 4.3 shows an example of the trajectory-tracking performance of the robot during the launching phase. In this example, the robot starts from a stationary balanced upright position and aims to reach the launching state $\eta^d = (2.04, -1.45, -4.4, 7.5)$, where η^d denotes the desired values of $\eta = (q_1, q_2, \dot{q}_1, \dot{q}_2)$. To obtain the value of η^d, a trial-and-error procedure is employed. The objective of this process was to find a suitable launching state that results in a successful hop with a desired hop length. Different values of launching states have been tried using the dynamics simulator to achieve a successful hop in which the robot can recover its balance after landing. The above mentioned value of η^d is a suitable launching state for a 50cm hop length. Note that this is not the only launching state that can provide such a successful hop with 50cm length. In fact, according to the simulation results (of the trial-and-error process), there are many suitable values between $(2, -1.4, -4.25, 7.2)$ and $(2.07, -1.5, -4.55, 7.8)$ that can be used as the launching state.

The solid and dashed curves in Fig. 4.3 show X and X^d versus time, respectively. In this figure the beginning part of the reference trajectory has been cut off and therefore X^d is not started from zero at $t = 0$. The part of the reference trajectory that is not used is the end part of the balancing motion in the time-reversal technique in which the robot is quite close to its upright balanced configuration. Cutting off this part of the motion helps the robot to make faster crouching and launching motions.

According to Fig. 4.3, there is almost no tracking error before $t = 0.8$s except in the beginning part of the motion which is due to the different start points. After $t = 0.8$s, the error starts to grow up until about $t = 1.3$s from which time onwards it stays almost constant until the end. The reason for this error can be seen in Fig. 4.4, which shows the normal component of the ground-reaction force acting on the foot (f_y) and also the ratio of the tangent and normal components (f_x/f_y) along with the same graph of X in Fig. 4.3. The ratio f_x/f_y is limited to ± 0.4 by the friction cone. It can be seen from these graphs that this ratio decreases after $t = 0.8$s (and more quickly after $t = 0.9$s) until it hits the limit at about $t = 1$s. This decrease means that there is an increase in the absolute value of f_x during this period ($t = 0.8$s and $t = 0.9$s) and therefore more pre-sliding movement of the foot on the ground.

As can be seen in the graphs in Fig. 4.4, the normal force is dropped to zero, and the contact between the foot and the ground is lost at about $t = 1.1$s. The reason for the drop in normal force and loss of contact is that the robot is crouching rapidly at this time, and so its CoM is accelerating rapidly downwards. Also it can be seen

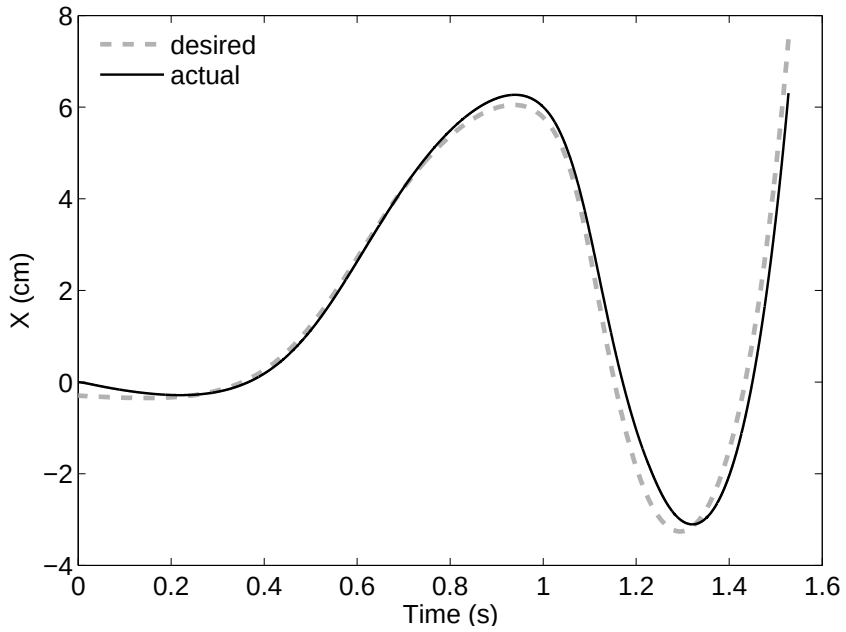


Figure 4.3: Horizontal location of the CoM of the robot with respect to its foot during the launching phase

that the foot is slipping for about 0.07s before losing and 0.04s after regaining its contact with the ground. The spike in the normal force graph, at a very short time before $t = 1.2$s, shows the instant of the impact between the foot and the ground. The slipping ends at approximately $t = 1.22$s, which is when the robot starts to decelerate as it reaches the end of the crouch and begins to launch itself upwards. The tracking controller is back on track about 0.1s after the slipping stops and follows the desired trajectory but with a little bit of time delay. There is also a short period just before lift-off when the foot is slipping backwards. All three graphs end when lift-off is supposed to happen, which is a few milliseconds before it actually happens. This is why the normal force has not dropped all the way to zero at the end of the graph. The spike at the beginning of the normal force is because of the impact between the foot and the ground at $t = 0$ since the initial value of y is set to zero.

As already mentioned, the reference trajectory that is used in the launching phase is not necessarily the optimal trajectory. However, the overall performance of the trajectory-tracking controller during the launching motion is reasonably good. The most important aspect of this motion is the accuracy at the end of the trajectory. This accuracy can be observed in the difference between the final actual state of the launching motion, which is denoted by η^a, and the launching state (η^d). The final actual state is calculated as follows. First, a measure which is called total error percentage is defined for each instant of the launching motion as

$$TEP = \sum_{i=1}^{4} \left| \frac{\eta(i) - \eta^d(i)}{\eta^d(i)} \right|,$$

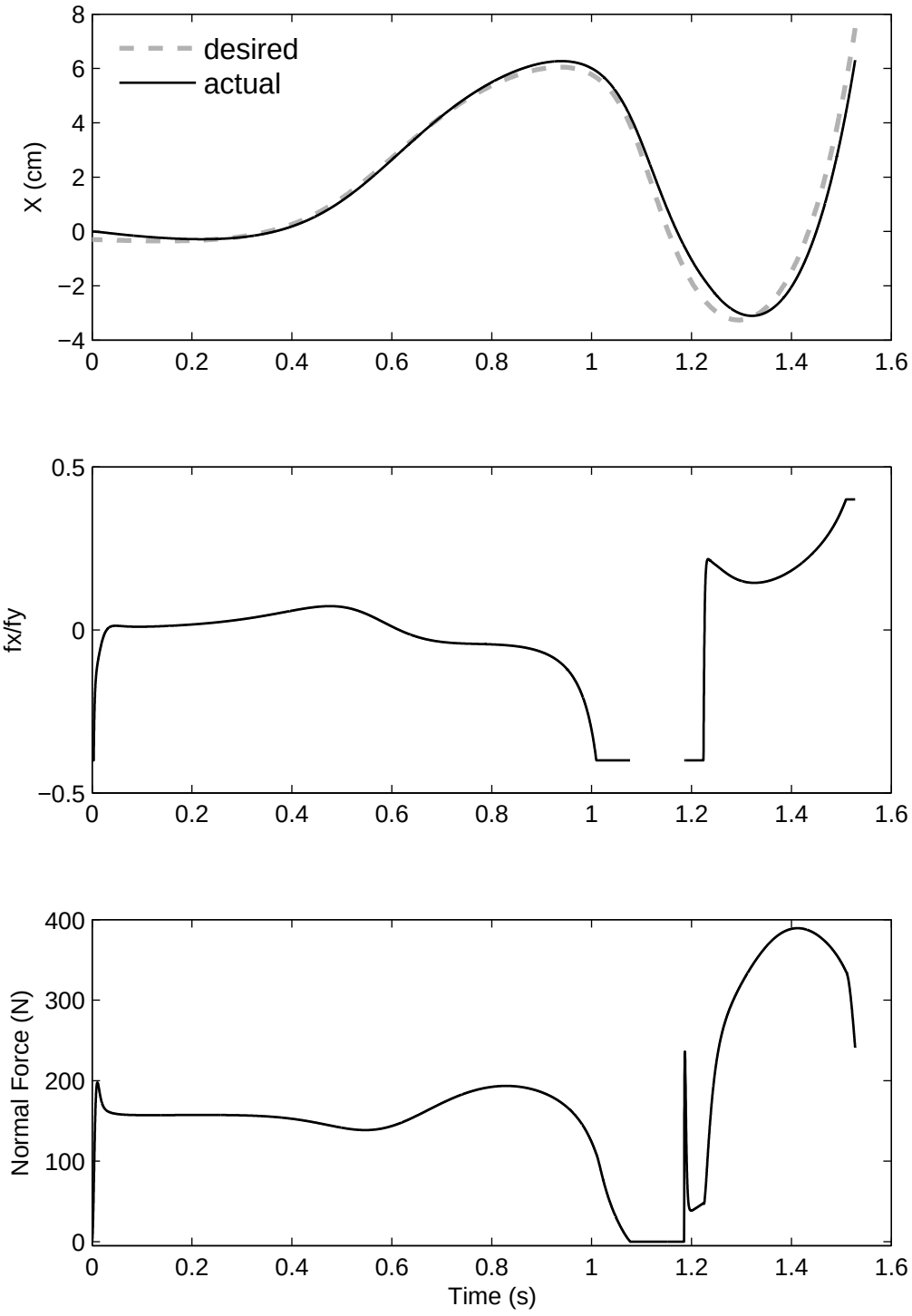


Figure 4.4: Normal force and the ratio between the friction force and normal force during the launching phase

where

$$\eta(1) = q_1\,, \quad \eta(2) = q_2\,, \quad \eta(3) = \dot{q}_1\,, \quad \eta(4) = \dot{q}_2\,.$$

Thus, TEP is an array with the same length as the time array of the simulation. Since TEP is always positive, then η^a is defined to be state variables of the robot at the instant in which TEP has its minimum value. For the launching motion in this section, the final actual state variables are $\eta^a = (2.0631, -1.443, -4.4038, 7.4806)$ which are quite close to the desired values.

4.2.3 Flight Phase

When the robot takes off from the ground, the ability to control its motion significantly decreases. During the flight phase, the angular momentum of the robot is constant, the CoM travels along a given trajectory which cannot be changed by the controller, and the robot has 3 degrees of under-actuation. Also, in the flight phase, because of the mass difference between the robot's upper and lower bodies (the lower body is much lighter), applying a torque to the actuator causes much more movement to the lower body than to the upper one. This means that controlling the upper body's motion is much more difficult than for the lower body's motion. For these reasons, the objective of the flight-phase control is limited to controlling foot placement at landing.

To control the location of the foot at landing, the trajectory-tracking controller is used to bring the robot from the launching state to the desired landing state (including the desired location of the foot at landing) during the flight phase. The launching state is the initial state for the flight phase. To use the trajectory-tracking controller, the first step is to define a reference trajectory for X between the launching state and the desired landing state. If X_G and x are the absolute horizontal displacement of the CoM and foot of the robot, respectively, then

$$X = X_G - x\,. \tag{4.10}$$

X_G is a given function of time right after the take-off instant, so by defining a desired trajectory for x, the desired value of X can be calculated easily. A sigmoid function of the form $f(z) = 1/(1 + \mathrm{e}^{-z})$ is used as a reference function for x so that the forward velocity of the foot at the landing instant will be zero. This helps the robot not to slide (or to slide only a little). The actual trajectory for x is chosen to be

$$x = \frac{L_{\mathrm{H}}}{1 + e^{-(a_1 t + a_0)}}\,, \tag{4.11}$$

where L_{H} is the desired hop length, $a_0 = -6$, $a_1 = 12/T_{\mathrm{H}}$ and T_{H} is the flight-phase duration time.

Therefore, the desired trajectory for the foot during the flight phase depends on the desired value of the hop length and the flight duration. Using (4.10) and (4.11),

we have

$$\dot{X} = \dot{X}_G - \dot{x} = \dot{X}_G - \frac{a_1 L_H e^{-(a_1 t + a_0)}}{(1 + e^{-(a_1 t + a_0)})^2}\,. \tag{4.12}$$

Since $\dot{X}_G$ is a given function of time during the flight phase, so the desired value of $\dot{X}$ within the desired trajectory can be easily calculated by using the above equation. Because the angular momentum of the robot is not controllable during this phase, k_p is set to zero and therefore the control law for the purpose of the trajectory-tracking is

$$\tau = -k_v(\dot{X} - \dot{X}^d) - k_x(X - X^d)\,. \tag{4.13}$$

Fig. 4.5 shows the controller's trajectory tracking performance during the flight phase for a hop with a desired length of 50cm. The flight phase duration is 0.286s. The small displacement from zero at the beginning of the graph shows the amount of slipping of the foot during the crouching phase. The hop length is the difference between x at the take-off and landing instants which is 49.26cm in this case. The final location of the foot after the hop is 53.39cm and the difference between these two values ($53.39 - 49.26 = 4.07$cm) is the total amount of slipping of the foot during the hopping motion.

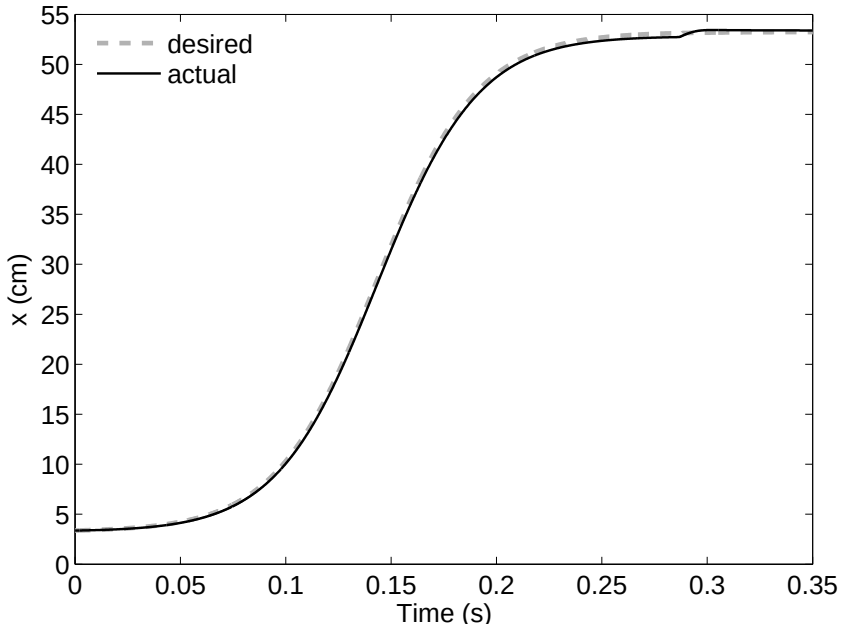


Figure 4.5: Horizontal location of the foot during the flight phase

4.2.4 Landing and Rebalancing

Once the foot touches the ground, the controller switches from trajectory-tracking to balancing in order to bring the robot to its upright balanced configuration. Fig. 4.6 shows the position and velocity of the foot during the flight phase represented in

Fig. 4.5 and the first 0.2 seconds of rebalancing. The landing moment ($t = 0.286$s) coincides with the obvious sharp discontinuities in both the $\dot{x}$ and $\dot{y}$ graphs. According to these graphs, the forward velocity of the foot ($\dot{x}$) at this instant is close to zero, but the downward velocity ($|\dot{y}|$) is about 2.5m/s.

When the foot hits the ground, most of the lower body's momentum is absorbed in the impact and some of it is converted to forward motion of the foot. Therefore $\dot{x}$ increases suddenly right after the impact (i.e. the landing instant) and the foot slips forward a little. However, no further slipping occurs during the rest of the rebalancing motion.

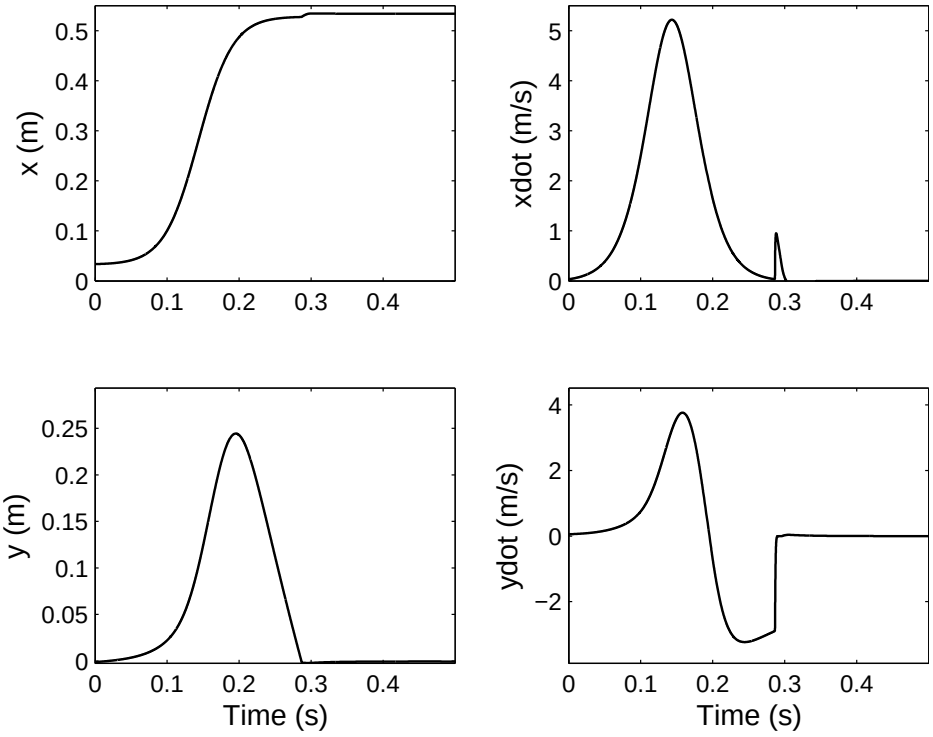


Figure 4.6: Translational state variables and their derivatives during flight phase

4.3 Discussion on Simulation Results

This section investigates the simulation results of a single-hop motion with 50cm length which is discussed in the previous section. Figure 4.7 shows the absolute vertical displacement of the CoM (Y_G) against its horizontal displacement (X_G). The start and end point of the graph show the location of the CoM when the robot is in its upright balanced configuration at either end. At the beginning of the motion, the robot tilts slightly forward and consequently brings the x component of the CoM from zero to a positive value. Then the crouching phase starts and Y_G decreases quite rapidly to about 45cm while X_G does not change much (by about 5cm). When

Y_G reaches its minimum value, the launching motion starts. The objective of this motion is to accelerate the robot in positive directions of X_G and Y_G and give it enough momentum to be able to hop. As a result, both X_G and Y_G increase quickly during this motion. As already mentioned, the crouching and launching motions are implemented in a single phase and by using the trajectory-tracking controller. The final value of X_G is 53.39cm which is identical to the final value of x.

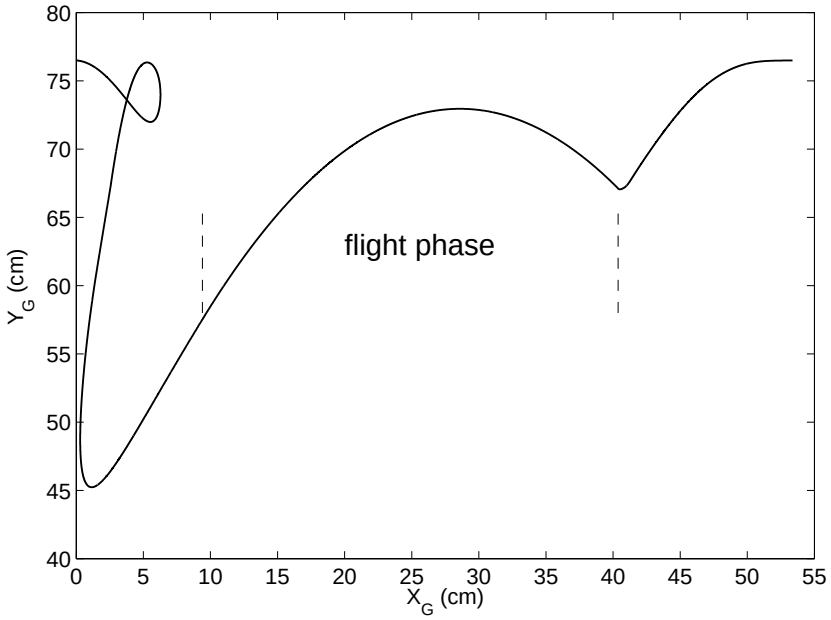


Figure 4.7: Footprint of the CoM of the 2D hopper during a complete hop

Figure 4.8 shows the absolute velocities of the CoM in the horizontal ($\dot{X}_G$ or V_x in the plot) and the vertical ($\dot{Y}_G$ or V_y in the plot) directions and also the controller torque(τ) during the hopping motion. As can be seen in the first graph in this figure, $\dot{X}_G$ is constant during the flight phase starting at about $t = 1.5$s. The value of $\dot{X}_G$ is also fixed for about 0.1s at $t = 1.07$s. This is because of the loss of contact during the crouching motion which is described in the previous section. Very tiny discontinuities in both $\dot{X}_G$ and $\dot{Y}_G$ graphs at about $t = 1.2$s show the moment in which the foot regains its contact with the ground during the crouching and launching phase.

According to the velocity graphs in Fig. 4.8, both $\dot{X}_G$ and $\dot{Y}_G$ increase quickly during the launching motion. Such a fast movement necessitates that the actuator applies a considerable amount of torque to the knee joint. As can be seen in the controller torque graph, τ rises very rapidly to about 160N.m during the launching motion. Once the flight phase starts, the controller shifts to the flight phase controller and aims to follow the reference sigmoid function and place the foot on the desired location. According to Fig. 4.8, the controller hits the saturation limit, which has been set to 160N.m, in both positive and negative directions during the flight phase. As is expected, the value of $\dot{Y}_G$ decreases linearly during the flight phase. Sharp breaks in all three graphs in Fig. 4.8 are related to the instant of the landing. At this moment

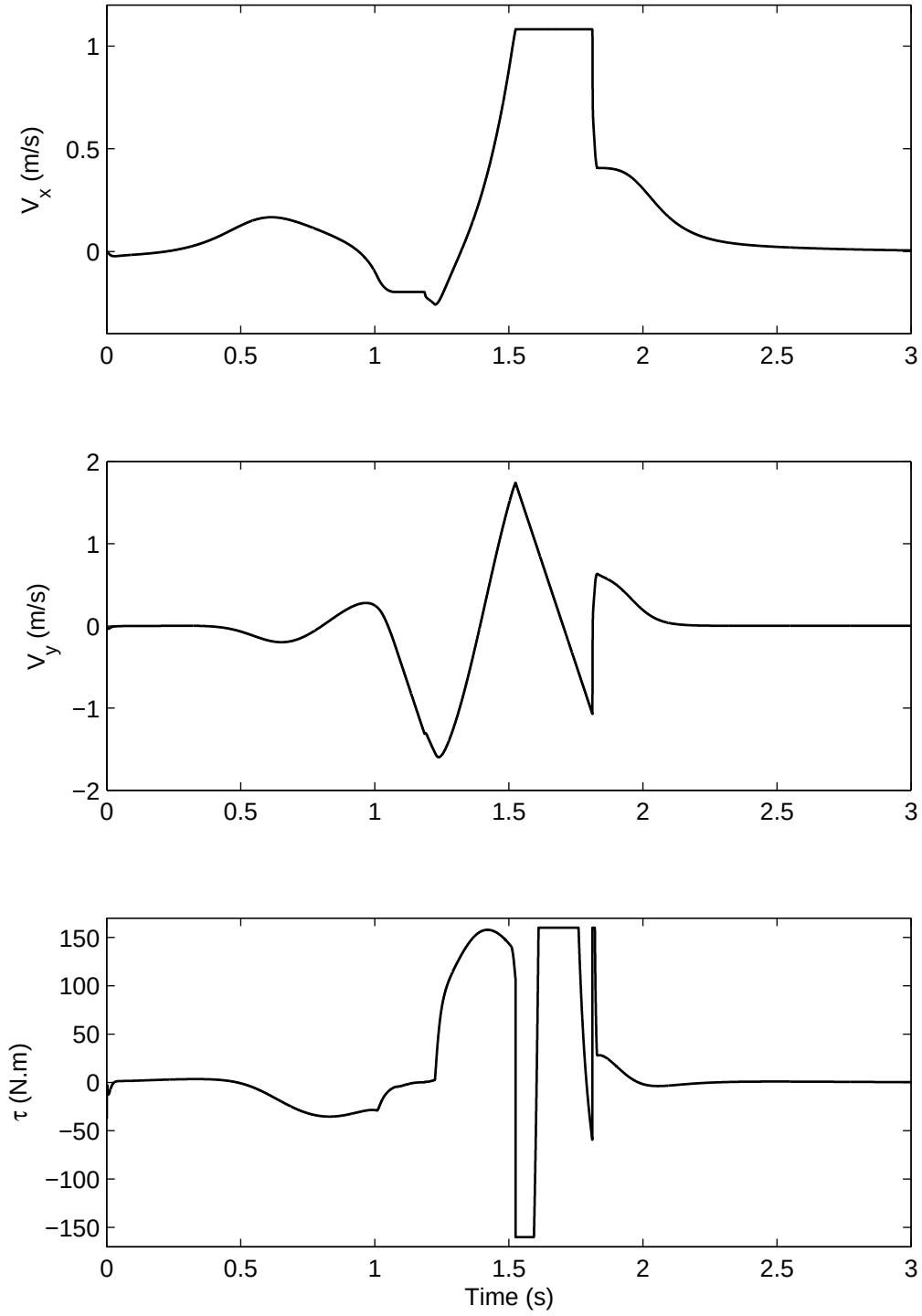


Figure 4.8: Velocity of the CoM and control torque during a complete hop

the controller hits the torque limit for a few milliseconds and then quickly drops to about 30N.m.

4.4 Summary

In this chapter, a single-hop motion is studied for a knee-leg hopping robot with only one actuated revolute joint. The balancing control algorithm from chapter 3 is extended to perform trajectory-tracking maneuvers, which enables the robot to perform the crouching, lift-off and flight phases of a single hop, as well as re-balancing after landing. Simulation results demonstrate that the control system works well, and that it is not significantly affected by small amounts of slipping between the foot and the ground. The proposed angular momentum based controller is able to control the robot's motion during a single-hop in addition to all of its capabilities that were discussed in the previous chapter. Although this chapter has not considered continuous hopping motion (which would require the robot to move from landing directly to the launching state), it has demonstrated the possibility of performing a complete hop using only one actuator.

Chapter 5

Balancing Control Algorithm for a 3D Under-actuated Robot

This chapter studies the balancing control problem for a 3D under-actuated robot. The robot, which is called "3D balancer" in this chapter, is essentially a spatial version of the 2D balancer in chapter 3 where the passive and active revolute joints are replaced with a passive spherical joint and an active two DoF joint, respectively. This particular 3D under-actuated robot has not previously been studied in the literature. However, Miyashita et al. [2006] and Xinjilefu et al. [2009] introduced control algorithms for similar spatial under-actuated robots.

Miyashita et al. [2006] proposed a 3D motion control algorithm for an under-actuated manipulator which is called AcroBox. The AcroBox is in fact a two-link 3D robot with five DoF and two actuators which is similar to the 3D balancer in the degrees of under-actuation. The difference between the two robots lies in the mechanism of the active joint which is explained in section 5.1. The proposed controller for the AcroBox, which is based on the idea of an output zeroing controller for a similar system in 2D, is able to balance the robot in its upright position. The controller is essentially an extended version of the planar balancing controller in [Yamakita et al., 2002; Yonemura and Yamakita, 2004].

Xinjilefu et al. [2009] studied the postural control problem for a four DoF 3D under-actuated robot. The robot consists of two links connected to each other by an actuated universal joint and the lower link is attached to the ground by a passive universal joint. Xinjilefu et al. [2009] used stochastic programming to find oscillatory inputs that bring the system close to the unstable upright balanced configuration.

In this chapter, it is shown that the 3D balancer's motion can be decomposed into bending and swivelling motions. An angular momentum based controller is introduced to control the angular momentum of the robot in bend and swivel directions. Simulation results show that the proposed controller is able to stabilize the 3D balancer robot in its upright balanced configuration starting from initial unstable configurations, stabilize the robot at any unstable balanced configuration in any arbitrary vertical plane starting from an upright balanced position, and rotate the plane of the robot about the vertical axis in order to change the orientation of the robot. The two latter capabilities of the controller, which are due to the decoupled motion

of the robot, are demonstrated for the first time in this chapter.

5.1 Robot Model

A schematic diagram of the 3D balancer robot studied in this chapter is shown in Fig. 5.1. The robot consists of two links connected to each other by a two DoF actuated joint (knee). The lower body is then connected to the ground via a spherical passive joint. Therefore, the whole system has five DoF with only two actuators.

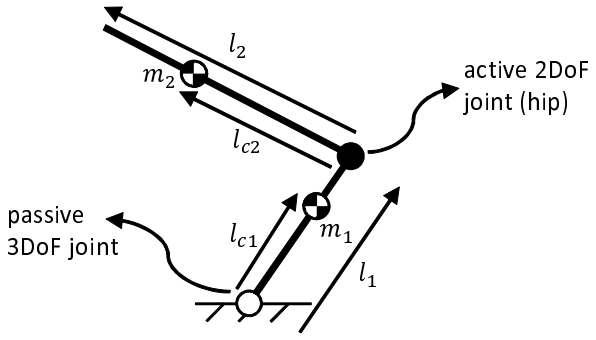


Figure 5.1: Schematic diagram of the 3D balancer robot

A double Cardan joint, which is a constant velocity joint, is used as the two DoF active joint of the 3D balancer. The reason is that by using a constant velocity joint at the knee, it is possible to "instantaneously" decouple the robot's motion into two parts which are called bending and swivelling motions in this chapter and will be explained later in section 5.3. This claim is proved in the Appendix. The robot's motion is instantaneously decoupled, meaning that it is decoupled only on the force-acceleration level (i.e. it may still be coupled by velocity terms). This property was discovered by Dr. Roy Featherstone. In fact, the 3D balancer is deliberately designed to provide this property by using a constant velocity joint in the knee.

5.1.1 Kinematics of the Spherical Passive Joint

Let q_1, q_2 and q_3 denote the generalized coordinates of the spherical passive joint. Fig. 5.2 shows the fixed coordinate frame xyz (fixed to the ground) and three other relative coordinate frames that are used to express the orientation of the lower body with respect to the ground. As shown in this figure, the angles q_1, q_2 and q_3 are the rotation angles of coordinate frames $x_1y_1z_1$, $x_2y_2z_2$ and $x_3y_3z_3$ with respect to their predecessor ones about z, y_1 and x_2, respectively (see Table 5.1).

Rotational coordinate transform matrices from $x_1y_1z_1$ to xyz, $x_2y_2z_2$ to $x_1y_1z_1$ and $x_3y_3z_3$ to $x_2y_2z_2$ are denoted by R_1, ${}^1R_2$ and ${}^2R_3$, respectively. These matrices are as follows:

$$R_1 = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 \\ \sin(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.1}$$

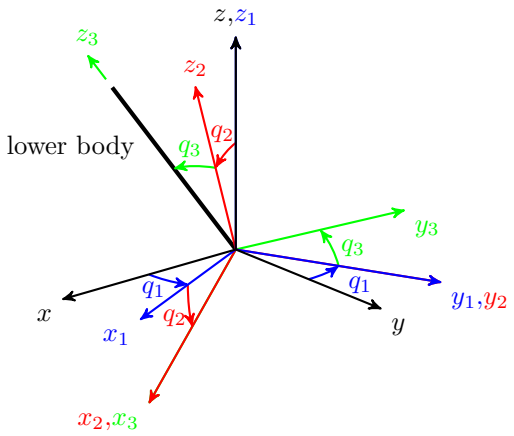


Figure 5.2: Coordinate frames of the passive joint

Table 5.1: Relationships between coordinate frames at the passive joint

frames	predecessor frame	axis of rotation	angle of rotation
$x_1y_1z_1$	xyz	z	q_1
$x_2y_2z_2$	$x_1y_1z_1$	y_1	q_2
$x_3y_3z_3$	$x_2y_2z_2$	x_2	q_3

$$ {}^1R_2 = \begin{bmatrix} \cos(q_2) & 0 & \sin(q_2) \\ 0 & 1 & 0 \\ -\sin(q_2) & 0 & \cos(q_2) \end{bmatrix}, \tag{5.2} $$

$$ {}^2R_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(q_3) & -\sin(q_3) \\ 0 & \sin(q_3) & \cos(q_3) \end{bmatrix}. \tag{5.3} $$

Thus, the coordinate transformation matrix from $x_3y_3z_3$ to xyz is $R_3 = R_1\,{}^1R_2\,{}^2R_3$. The coordinates of the CoM of the lower body, expressed in the third coordinate frame, are $(0, 0, l_{c1})$ implying that the first link is attached to the third frame along the positive direction of its z axis (z_3). The coordinates of the lower body's CoM, expressed in the fixed frame, are then

$$ C_1 = R_3 \begin{bmatrix} 0 \\ 0 \\ l_{c1} \end{bmatrix} = l_{c1} \begin{bmatrix} \sin(q_1)\sin(q_3) + \cos(q_1)\sin(q_2)\cos(q_3) \\ -\cos(q_1)\sin(q_3) + \sin(q_1)\sin(q_2)\cos(q_3) \\ \cos(q_2)\cos(q_3) \end{bmatrix}. \tag{5.4} $$

Therefore, independent of the value of q_1, the lower body is upright if and only if $q_2 = q_3 = 0$.

5.1.2 Kinematics of the Knee Joint – A Double Cardan Joint

A double Cardan joint is a constant velocity joint in which the angular velocity of the input shaft is always equal to the angular velocity of the output shaft. These angular velocities are denoted by ω_{in} and ω_{out} in Fig. 5.3. This figure shows a schematic diagram of a double Cardan joint which consists of two universal joints that are head-to-head coupled together via a connecting link.

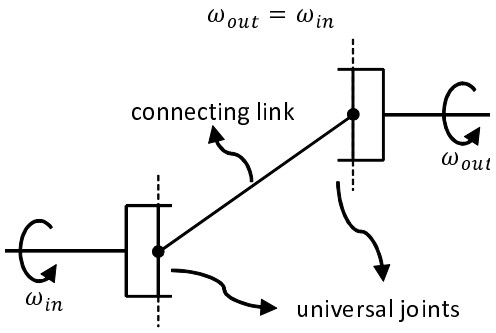


Figure 5.3: Schematic diagram of a double Cardan joint

To use a double Cardan joint in the knee joint of the 3D balancer, the connecting link between the two universal joints is shrunk to zero and then discarded. Therefore the two revolute joints, which connect the link to the universal joints, become one. A schematic diagram of the knee joint is shown in Fig. 5.4. The kinematics of this joint can be fully expressed by three generalized coordinates which are denoted by q_4, q_5 and q_6 in this figure. However, because of the constant velocity constraint, the joint has only two actual DoF and consequently only two actuators are required to actuate this joint.

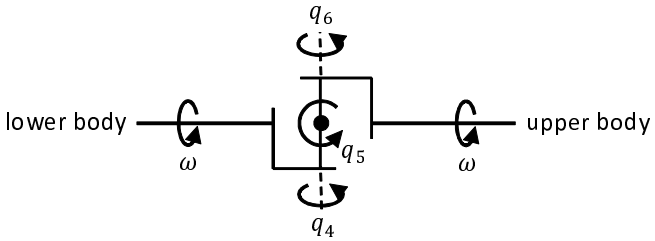


Figure 5.4: Schematic diagram of the constant velocity joint at the knee

As shown in Fig. 5.4, the rotation axis of q_5 is always perpendicular to those of q_4 and q_6. Here, q_5 is the angle between the rotation axes of q_4 and q_6, and consequently these axes are collinear if $q_5 = 0$. It is clear that, when $q_4 = q_5 = q_6 = 0$, both the upper and lower bodies are oriented in the same direction. The constant velocity constraint in the knee joint can be expressed as

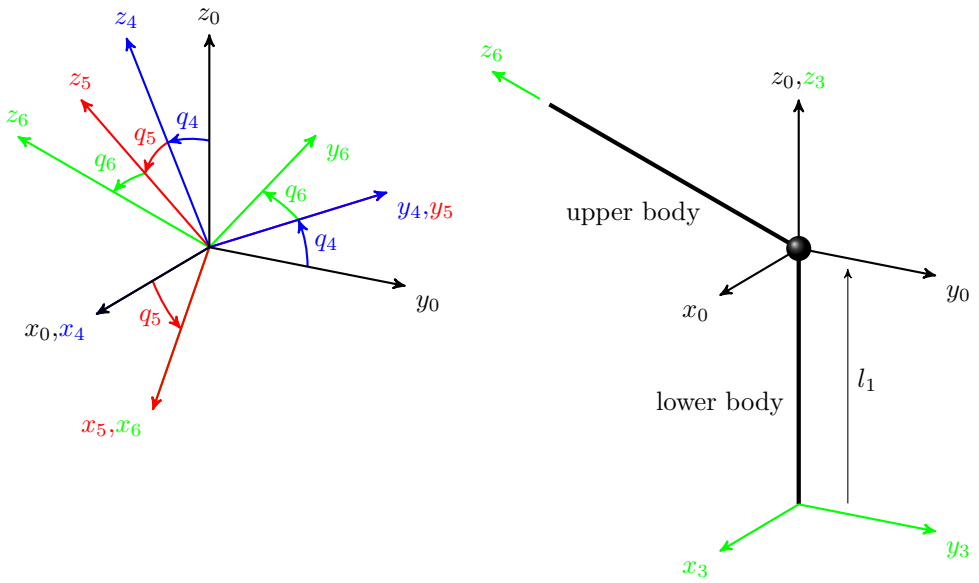
$$\dot{q}_4 = \dot{q}_6 \quad \Longrightarrow \quad q_4 = q_6 \,. \tag{5.5}$$

This constraint is in fact an explicit velocity constraint which is implemented in simulations using the method described in §3.2 of [Featherstone, 2008]. More details about employing this method in simulations are discussed in section 5.4.

Fig. 5.5 shows the coordinate frames that are used to express the relative rotations of the upper body with respect to the lower body (q_4, q_5 and q_6). The coordinate frame $x_0y_0z_0$ is fixed to the lower body with its origin at the knee joint and its z_0 axis along the extension of the lower body. This frame is identical to the $x_3y_3z_3$ frame (in Fig. 5.2) which is translated from the passive joint to the active one along the z_3 axis. Table 5.2 states the relationships between the coordinate frames $x_4y_4z_4$, $x_5y_5z_5$ and $x_6y_6z_6$ and their predecessor ones.

Table 5.2: Relationships between coordinate frames at the knee joint

frames	predecessor frame	axis of rotation	angle of rotation
$x_4y_4z_4$	$x_0y_0z_0$	x_0	q_4
$x_5y_5z_5$	$x_4y_4z_4$	y_4	q_5
$x_6y_6z_6$	$x_5y_5z_5$	x_5	q_6 $(= q_4)$



(a) (b)

Figure 5.5: Coordinate frames of the knee joint

The rotational coordinate transformation matrices from $x_4y_4z_4$ to $x_0y_0z_0$, $x_5y_5z_5$ to $x_4y_4z_4$ and $x_6y_6z_6$ to $x_5y_5z_5$ are denoted by ${}^0R_4$, ${}^4R_5$ and ${}^5R_6$, respectively, and are as follows:

$$ {}^0R_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(q_4) & -\sin(q_4) \\ 0 & \sin(q_4) & \cos(q_4) \end{bmatrix}, \tag{5.6} $$

$$ {}^{4}R_5 = \begin{bmatrix} \cos(q_5) & 0 & \sin(q_5) \\ 0 & 1 & 0 \\ -\sin(q_5) & 0 & \cos(q_5) \end{bmatrix} , \tag{5.7} $$

$$ {}^{5}R_6 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(q_6) & -\sin(q_6) \\ 0 & \sin(q_6) & \cos(q_6) \end{bmatrix} . \tag{5.8} $$

Therefore, knowing that $q_4 = q_6$, the coordinate transformation matrix from $x_6y_6z_6$ to $x_0y_0z_0$ is

$$ {}^{0}R_6 = {}^{0}R_4\,{}^{4}R_5\,{}^{5}R_6 = \begin{bmatrix} \cos(q_5) & \sin(q_4)\sin(q_5) & \cos(q_4)\sin(q_5) \\ \sin(q_4)\sin(q_5) & \cos^2(q_4) - \sin^2(q_4)\cos(q_5) & -\sin(q_4)\cos(q_4)(1+\cos(q_5)) \\ -\cos(q_4)\sin(q_5) & \sin(q_4)\cos(q_4)(1+\cos(q_5)) & -\sin^2(q_4) + \cos^2(q_4)\cos(q_5) \end{bmatrix} . \tag{5.9} $$

Let $\hat{l}$ and $\hat{u}$ represent unit vectors in the directions of the lower and upper bodies, respectively, both expressed in the coordinate frame $x_0y_0z_0$. According to Fig. 5.5,

$$ \hat{l} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} , \tag{5.10} $$

which points in the positive direction of the vertical axis. Also it is shown in this figure that the upper body is oriented along the positive direction of z_6 and therefore

$$ \hat{u} = {}^{0}R_6 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(q_4)\sin(q_5) \\ -\sin(q_4)\cos(q_4)(1+\cos(q_5)) \\ -\sin^2(q_4) + \cos^2(q_4)\cos(q_5) \end{bmatrix} . \tag{5.11} $$

Since the corresponding axes of $x_3y_3z_3$ and $x_0y_0z_0$ have the same orientation, the rotational transformation matrix between these two frames is an identity matrix: ${}^{0}R_3 = {}^{3}R_0 = I_{3\times3}$. This means that $\hat{u}$ is also the unit vector of the upper body's orientation expressed in the coordinate frame $x_3y_3z_3$.

As shown in Fig. 5.1, it is assumed that the CoM of the upper body is along the body and at a distance of l_{c2} from the knee joint. Thus, the coordinates of the CoM of the upper body expressed in the fixed frame are

$$ C_2 = R_3 \begin{bmatrix} 0 \\ 0 \\ l_1 \end{bmatrix} + R_3\,{}^{0}R_6 \begin{bmatrix} 0 \\ 0 \\ l_{c2} \end{bmatrix} = R_3(l_1\hat{l} + l_{c2}\hat{u}) . \tag{5.12} $$

5.2 Definitions

This section provides some definitions regarding the kinematics of the robot. These definitions are as follows:

- *robot plane (bend plane):* a plane which includes both links of the robot. The robot's CoM and the knee joint are always in this plane. This plane is not defined if both links are in a line (i.e. the bend angle is either zero or π).
- *bend axis:* an axis perpendicular to the robot plane passing through the knee joint (see Fig. 5.6).
- *bend angle:* the angle between the upper body and the extension of the lower body of the robot in the robot plane. The bend angle is denoted by ψ in Fig. 5.6(a). This angle is between 0 and π (i.e. $0 \leq \psi \leq \pi$).
- *bisector plane:* a plane with a normal vector along the bisector of the bend angle. This plane passes through the knee joint and is perpendicular to the robot plane.
- *swivel axis:* an axis perpendicular to the bisector of the bend angle and lying in the robot plane. This axis is the intersection line between the robot plane and the bisector plane (see Fig. 5.6).
- *swivel angle:* the angle between the projection of the upper body axis onto the x_0y_0 plane and the x_0 axis. The swivel angle is denoted by σ in Fig. 5.6. As mentioned in the previous section, the frame $x_0y_0z_0$ is fixed to the lower body.
- *vertical robot plane:* a vertical plane whose intersection with the horizontal plane (the ground) is the intersection line between the robot plane and the ground. This plane is illustrated in Fig. 5.7(a).
- *tilt angle:* the angle between the robot plane and the vertical robot plane. The tilt angle is represented by γ in Fig. 5.7(a) and is restricted between $-\pi/2$ and $\pi/2$. This angle is in fact the angle between the unit vector of the bend axis and its projection onto the horizontal plane (i.e. the xy plane of the fixed frame). The tilt angle is positive if the bend axis points upward.
- *robot plane angle:* the angle between the vertical robot plane and the x axis of the fixed frame. This angle is denoted by θ in Fig. 5.7(a).
- *angle of the lower body:* the angle between the lower body and the ground measured in the robot plane. This angle is represented by ϕ in Fig. 5.7(b).
- *balance error:* the horizontal distance between the CoM of the robot and its fixed joint in the robot plane. The balance error is denoted by X_{err} in Fig. 5.7(b).

5.3 Bending and Swivelling Motions

As already mentioned, it is possible to instantaneously decouple the robot's motion into bending and swivelling motions. These motions are defined as follows:

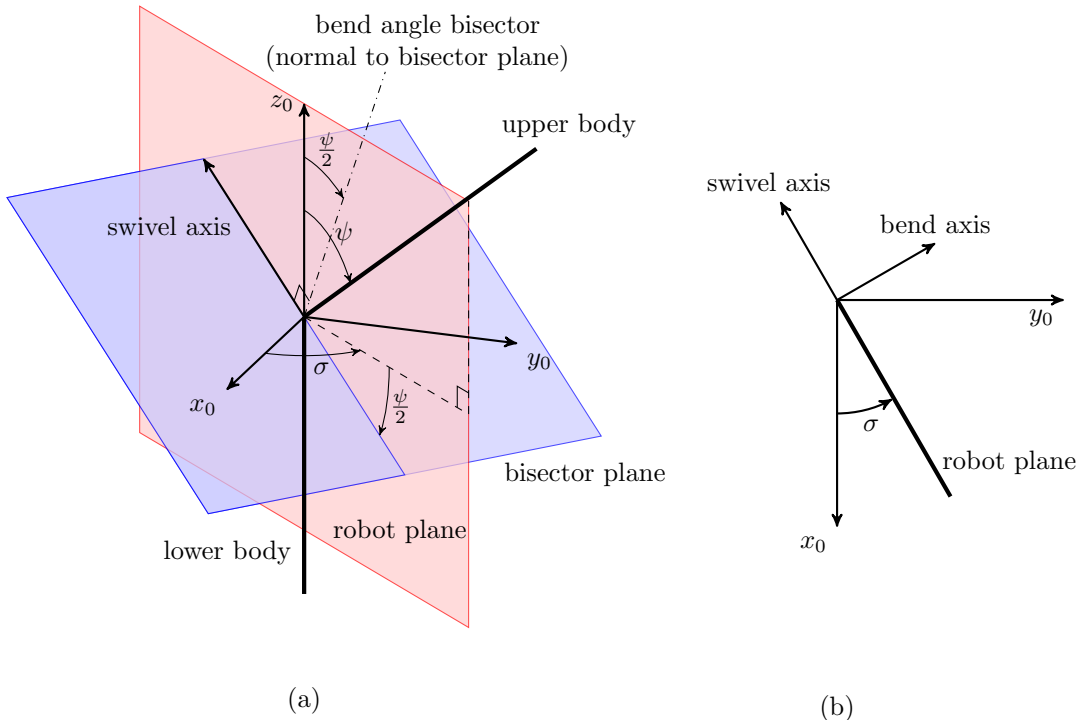


Figure 5.6: (a) robot and bisector planes, bend and swivel angles and swivel axis, (b) bend and swivel axes and swivel angle

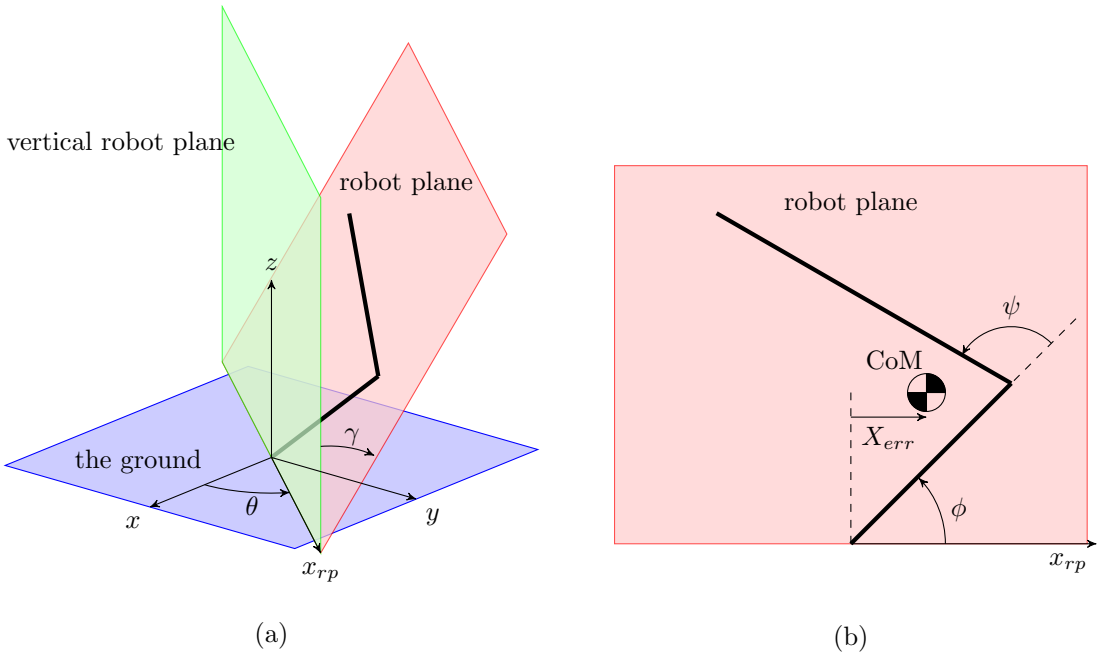


Figure 5.7: (a) vertical robot plane, robot plane, tilt angle and robot plane angle, (b) angle of the lower body, bend angle and balance error

Bending Motion — refers to a movement of the robot which occurs in the robot plane and affects the bend angle directly. Due to this motion, the upper link rotates relative to the lower link about the bend axis. This motion is similar to the planar motion of the 2D balancer robot. If there is no gravity (to change the orientation of the robot plane by making the robot fall over), applying torque to the knee joint and around the bend axis moves both links only in the robot plane and changes the bend angle without changing the orientation of the robot plane. Since the whole movement takes place in a plane, the direction of the bend axis remains fixed.

Swivelling Motion — is a motion where the upper body rotates relative to the lower body about the swivel axis. As this is an out-of-plane motion, it causes the direction of the swivel axis to spin relative to the lower body (and also the upper body). However, an observer in a fixed position relative to the robot plane sees the two bodies both rotating about their axes (at the same speed, of course) as shown in Fig. 5.8. In the absence of gravity, and with the robot at rest, a torque at the knee joint about the swivel axis causes a nonzero acceleration in the swivel direction ($\ddot{\sigma} \neq 0$) but zero acceleration in the bend direction ($\ddot{\psi} = 0$). The effect of swivelling is to rotate the robot plane, which alters the tilt angle (γ) between the robot plane and the vertical robot plane.

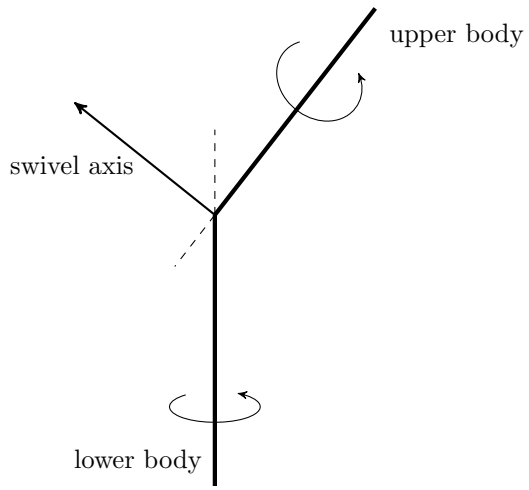


Figure 5.8: Swivelling motion of the robot

5.3.1 Kinematics of Bend and Swivel

In this subsection, the relationships between bend and swivel angles (ψ and σ) and joint coordinates in the dynamic model (q_4, q_5 and q_6) will be explored. According to the definitions, the bend angle is the angle between the upper body ($\hat{u}$) and the extension of the lower body ($\hat{l}$) which can be calculated as

$$\cos(\psi) = \hat{l} \cdot \hat{u} = -\sin^2(q_4) + \cos^2(q_4)\cos(q_5)\,. \tag{5.13}$$

Also the bend axis is an axis which is perpendicular to both links of the robot. The unit vector along the bend axis ($\hat{B}$) in the $x_0y_0z_0$ frame can be worked out as

$$\hat{B} = \frac{\hat{l} \times \hat{u}}{|\hat{l} \times \hat{u}|} = \frac{\hat{l} \times \hat{u}}{\sin(\psi)} = \frac{1}{\sin(\psi)} \begin{bmatrix} \sin(q_4)\cos(q_4)(1+\cos(q_5)) \\ \cos(q_4)\sin(q_5) \\ 0 \end{bmatrix} . \tag{5.14}$$

Clearly, this vector is in the x_0y_0 plane of the $x_0y_0z_0$ coordinate frame (see Fig. 5.6(b)). Note that, according to the definitions, the bend axis and therefore $\hat{B}$ are not defined when $\sin(\psi) = 0$. Since $|\hat{B}| = 1$ then

$$\sin(\psi) = \sqrt{\sin^2(q_4)\cos^2(q_4)(1+\cos(q_5))^2 + \cos^2(q_4)\sin^2(q_5)} . \tag{5.15}$$

As stated in the definitions, the swivel axis is an axis in the robot plane and is perpendicular to the bisector of the bend angle (see Fig. 5.6(a)). The unit vector along the swivel axis ($\hat{S}$) in the $x_0y_0z_0$ frame is

$$\hat{S} = \frac{\hat{l} - \hat{u}}{|\hat{l} - \hat{u}|} = \frac{1}{|\hat{l} - \hat{u}|} \begin{bmatrix} -\cos(q_4)\sin(q_5) \\ \sin(q_4)\cos(q_4)(1+\cos(q_5)) \\ 1+\sin^2(q_4) - \cos^2(q_4)\cos(q_5) \end{bmatrix} . \tag{5.16}$$

As shown in Fig. 5.6(b), the swivel angle (σ) is the angle between the projection of the upper body axis onto the x_0y_0 plane and the x_0 axis. Therefore, using $\hat{u}$ in (5.11), it follows that

$$\cos(\sigma) = \frac{\cos(q_4)\sin(q_5)}{\sqrt{\cos^2(q_4)\sin^2(q_5) + \sin^2(q_4)\cos^2(q_4)(1+\cos(q_5))^2}} , \tag{5.17}$$

and

$$\sin(\sigma) = \frac{-\sin(q_4)\cos(q_4)(1+\cos(q_5))}{\sqrt{\cos^2(q_4)\sin^2(q_5) + \sin^2(q_4)\cos^2(q_4)(1+\cos(q_5))^2}} . \tag{5.18}$$

Substituting (5.15) into (5.17) and (5.18), yields

$$\cos(\sigma)\sin(\psi) = \cos(q_4)\sin(q_5) , \tag{5.19}$$

and

$$\sin(\sigma)\sin(\psi) = -\sin(q_4)\cos(q_4)(1+\cos(q_5)) . \tag{5.20}$$

5.4 Motion Equations

This section describes the method that is used to calculate forward dynamics of the robot and enforce the constant velocity constraint. Let q denote the matrix of the coordinates of the robot and let F denote the matrix of the torques applied to the

robot by the controller (control torques) as

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix}, \quad \text{and} \quad F = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix},$$

where τ_4, τ_5 and τ_6 are the control torques which are exerted by the controller about the rotation axes of q_4, q_5 and q_6, respectively (see Fig. 5.4).

According to classical dynamics, the motion equation of the 3D balancer can be written as

$$M(q)\ddot{q} + N(q, \dot{q}) = F\,, \tag{5.21}$$

where $M(q)$ is the robot's inertia matrix and $N(q, \dot{q})$ is called the bias force matrix which accounts for the Coriolis and centrifugal forces, gravity and any other forces acting on the system other than the control torques.

As already mentioned, there is a constant velocity constraint on the knee joint of the robot (5.5) which can be explicitly expressed as

$$\dot{q} = G\dot{y}\,, \tag{5.22}$$

where y is a matrix containing the generalized coordinates of the robot

$$y = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix},$$

and G is

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

By differentiating the explicit constraint equation (5.22), $\ddot{q}$ will be

$$\ddot{q} = G\ddot{y} + \dot{G}\dot{y} \quad \xrightarrow{\dot{G}=0} \quad \ddot{q} = G\ddot{y}\,, \tag{5.23}$$

According to the equations of motion for a system with explicit motion constraints, which is described in §3.2 of [Featherstone, 2008], the motion equations for the 3D

balancer are

$$(G^T M G)\ddot{y} = G^T(F - N)\,, \tag{5.24}$$

where G^T is the transpose matrix of G. Therefore, the matrix of generalized forces is

$$Q = G^T F = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \tau_4 + \tau_6 \\ \tau_5 \end{bmatrix}. \tag{5.25}$$

Thus, as an effect of the constant velocity constraint in (5.5), there are only two independent control torques that influence the system which are $(\tau_4 + \tau_6)$ and τ_5.

5.5 Control Algorithm

The 3D balancer is designed to be instantaneously decoupled in the bend and swivel directions. Decoupling the robot's motion, it is now possible to decompose the controller into a bending motion controller and a swivelling motion controller. Each of these controllers is responsible for controlling its corresponding motion during a balancing motion. The decomposed controller is called "bend-swivel" controller in this chapter. The objective of the swivel controller, during a balancing motion, is to bring the robot plane to a vertical configuration and the objective of the bend controller is to stabilize the robot in the robot plane. The bend-swivel controller is able to stabilize the robot in its upright unstable balanced configuration starting from an initial unstable configuration.

The bending motion of the 3D balancer is planar and its dynamics resemble the dynamics of the 2D balancer in chapter 3. Therefore, it is possible to use the balancing controller in chapter 3 as the bend controller to control the robot's motion in the robot plane.

Let L represent the angular momentum of the 3D balancer about the passive joint, expressed in the fixed frame, as

$$L = \begin{bmatrix} L_x \\ L_y \\ L_z \end{bmatrix}. \tag{5.26}$$

Then

$$\dot{L} = \begin{bmatrix} \dot{L}_x \\ \dot{L}_y \\ \dot{L}_z \end{bmatrix}, \quad \text{and} \quad \ddot{L} = \begin{bmatrix} \ddot{L}_x \\ \ddot{L}_y \\ \ddot{L}_z \end{bmatrix}. \tag{5.27}$$

It is known that the rate of change of the angular momentum of the 3D balancer about its fixed point ($\dot{L}$) is equal to the moment of the external forces about that

point. So $\dot{L}$ equals the moment of the gravity force about the fixed point,

$$\dot{L} = cg \begin{bmatrix} -Y \\ X \\ 0 \end{bmatrix} \quad \Longrightarrow \quad \ddot{L} = cg \begin{bmatrix} -\dot{Y} \\ \dot{X} \\ 0 \end{bmatrix} . \tag{5.28}$$

Here, c is the total mass of the robot, g is the gravitational constant and X and Y (and Z) are the coordinates of the CoM of the robot in the fixed frame.

Since $\dot{L}$ and $\ddot{L}$ are always zero in the z direction, it is impossible for any controller to change the values of $\dot{L}_z$ and $\ddot{L}_z$ and therefore the robot is uncontrollable about the z axis. This means that, during a balancing motion, if the robot starts with a non-zero initial value of L_z, it will keep rotating about the z axis even after the robot reaches its balanced configuration and only the air resistance or friction in the joints are able to stop its rotating motion.

The angular momentum of the robot and its derivatives in the bend direction are

$$L_b = L^T \tilde{B} , \quad \dot{L}_b = \dot{L}^T \tilde{B} , \quad \text{and} \quad \ddot{L}_b = \ddot{L}^T \tilde{B} , \tag{5.29}$$

and in the swivel direction are

$$L_s = L^T \tilde{S} , \quad \dot{L}_s = \dot{L}^T \tilde{S} , \quad \text{and} \quad \ddot{L}_s = \ddot{L}^T \tilde{S} , \tag{5.30}$$

where $\tilde{B}$ and $\tilde{S}$ are the unit vectors along the bend and swivel axes in the fixed frame, respectively.

$$\tilde{B} = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = R_3 \hat{B} , \tag{5.31}$$

and

$$\tilde{S} = \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix} = R_3 \hat{S} . \tag{5.32}$$

Thus, according to (3.11), the desired control torque in the bend direction, which is denoted by τ_b, is

$$\tau_b = k_{dd} \ddot{L}_b + k_d \dot{L}_b + k_p L_b + k_s (\psi_d - \psi) + c_5 g \cos(\phi + \psi) , \tag{5.33}$$

where ψ_d is the desired value of the bend angle. Comparing (5.33) and (3.11), it is clear that ϕ and ψ are equivalent to q_1 and q_2 of the 2D balancer in chapter 3. The gains of the bend controller (k_{dd}, k_d, k_p and k_s) can be calculated using the method described in section 3.4.

The angular momentum of the robot is also used to control the robot's motion in the swivel direction. The swivel controller in this chapter is the control law in (3.7),

$$\tau_s = k_1 \ddot{L}_s + k_2 \dot{L}_s + k_3 L_s , \tag{5.34}$$

where k_1, k_2 and k_3 are the swivel controller's gains. Since the swivel axis is perpendicular to the bend axis, using this control law will lead to controlling the angular momentum of the robot (and its derivatives) about the fixed joint around two perpendicular axes (in two perpendicular planes).

As already mentioned, the swivelling controller brings the robot plane to a vertical during the balancing motion of the robot. However, if the robot's plane is already vertical, then the swivelling motion can rotate the robot plane about the vertical axis and change its direction. This motion is called rotating motion in this chapter. To perform such a movement, a small modification is made in the control law in (5.34) by adding a proportional correcting term (i.e. $k_4(\theta_d - \theta)$). The swivel control law is then

$$\tau_s = k_1 \ddot{L}_s + k_2 \dot{L}_s + k_3 L_s + k_4(\theta_d - \theta)\,, \tag{5.35}$$

where k_4 is a gain of the controller and θ_d is the desired value of the robot plane angle. The gains of the swivel controller (k_1, k_2, k_3 and k_4) are tuned manually to get good results in simulations. The best way to obtain these gains is to calculate them mathematically by using the dynamics equations of the 3D balancer in the swivel direction.

Since τ_b and τ_S are control torques in the bend and swivel directions, respectively, the total control torque is

$$\tau = \tau_b \tilde{B} + \tau_s \tilde{S}\,. \tag{5.36}$$

Therefore, the control torque in the direction of each joint is

$$\begin{aligned} \tau_4 &= \tau^T J_4 \\ \tau_5 &= \tau^T J_5 \\ \tau_6 &= \tau^T J_6 \end{aligned}\,, \tag{5.37}$$

where J_4, J_5 and J_6 are the unit vectors along the axes of rotation of q_4, q_5 and q_6, respectively. These vectors, in the fixed coordinate frame, are

$$J_4 = R_3 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\,, \quad J_5 = R_3\,{}^0R_4 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\,, \quad J_6 = R_3\,{}^0R_4\,{}^4R_5 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\,. \tag{5.38}$$

As mentioned in section 5.4, the torques that the controller applies to the knee joint are $(\tau_4 + \tau_6)$ and τ_5 which are calculated as

$$\tau_{46} = \tau_4 + \tau_6 = \tau^T (J_4 + J_6) \quad \text{and} \quad \tau_5 = \tau^T J_5\,. \tag{5.39}$$

5.6 Simulation Results

In this section, simulation results of straightening, crouching and rotating motions of a 3D balancer are presented. All simulations have been performed in Simulink. The

parameters of the 3D balancer which are used in the simulations are

$$\begin{array}{ll} m_1 = 0.49\text{kg}\,, & m_2 = 0.11\text{kg}\,, \\ l_1 = 0.4\text{m}\,, & l_2 = 0.6\text{m}\,, \\ l_{c1} = 0.1714\text{m}\,, & l_{c2} = 0.4364\text{m}\,, \\ I_{1x} = I_{1y} = 0.0036\text{kg.m}^2\,, & I_{2x} = I_{2y} = 0.0043\text{kg.m}^2\,, \\ I_{1z} = 0\,, & I_{2z} = 0.002\text{kg.m}^2\,. \end{array} \tag{5.40}$$

These parameters are in fact the same as the parameters of the simulator's model for the 2D balancer in Table 3.14. The only difference is that the upper body of the 3D balancer has a non-zero moment of inertia about its z axis (z_6). This inertia helps the robot to balance with less swivelling motion.

Figures 5.9 to 5.20 show the simulation results of the 3D balancer moving from three different initial unbalanced configurations to the upright balanced one (i.e. $\phi = \frac{\pi}{2}$ and $\psi = 0$ that is equivalent to $q_2 = q_3 = q_4 = q_5 = 0$). The initial conditions of these three motions are mentioned in Table 5.3. Note that during the straightening motion, since the bend and swivel planes are not defined in the robot's upright position, the robot ends up in a configuration which is not exactly the upright one but is very close to the upright position.

Table 5.3: Initial conditions of the robot for the straightening motions ($q_1 = 0$ and $q_6 = q_4$)

examples	ψ	γ	X_{err}	q_2 (rad)	q_3 (rad)	q_4 (rad)	q_5 (rad)
first	90°	15°	0	-0.4034	-0.1829	0.7158	0.7131
second	90°	5°	5 cm	-0.3485	-0.4810	0.7183	0.7015
third	120°	45°	0	-0.8490	0.1124	0.9998	0.7789

Three parameters are used to characterize the initial conditions of the robot which are the bend angle (ψ), the tilt angle (γ) and the balance error (X_{err}). According to the definitions, the tilt angle can be calculated using the z-component of the bend axis as

$$\sin(\gamma) = B_z \implies \gamma = \sin^{-1}(B_z)\,. \tag{5.41}$$

As shown in Fig. 5.7(b), the kinematics of the 3D balancer in the robot plane are completely the same as the ones of the 2D balancer. Hence, the balance error can be calculated by using (3.3) as

$$X_{err} = \frac{1}{c}(c_4\cos(\phi) + c_5\cos(\phi + \psi))\,, \tag{5.42}$$

where $c_4 = m_1 l_{c1} + m_2 l_1$ and $c_5 = m_2 l_{c2}$. Obviously the robot plane is vertical if $\gamma = 0$ and the robot is in its balanced configuration if $\gamma = X_{err} = 0$.

Figures 5.9, 5.13 and 5.17 show the joint angles of the robot during the straightening motions. To magnify the changes in the joint angles, they are plotted on a time scale of zero to one second whereas the other graphs (ψ,γ and X_{err}) cover the first two seconds of the motion. As can be seen in these graphs, the value of q_1, which is the

rotation angle of the lower body about z axis (vertical) of the fixed frame, converges to a value other than zero. This means that, during a balancing motion, the robot rotates about the vertical axis by an unknown amount which in fact depends on the initial position of the robot. q_1 is not plotted in Fig. 5.17 because it converges to a very large value (i.e. -16.52rad) at the end of the motion. Rapid changes in the joint angles (q_2, q_3, q_4 and q_5) within about 0.3s of the beginning of the motions are due to the swivelling motion of the robot. It can be seen that in all three cases the joint angles (except q_1) converge to zero in about the first second of the motion.

In the first example the robot starts from a configuration which is similar to the initial configuration of the 2D balancer in Fig. 3.2 but with a 15° tilt angle of the robot plane. In the second example, the tilt angle is decreased but a 5cm balance error is added to the initial conditions. Comparing the results of these two motions it can be seen that decreasing the tilt angle results in less and slower changes in the joint angles and therefore less swivelling motion at the beginning. Also there is a relatively large movement in the opposite direction of the desired bend angle in the first motion, but the settling times of these two motions are almost the same.

In the third example the robot starts from an initial configuration in which $\gamma = 45^\circ$ and $\psi = 120^\circ$. It can be seen that the controller is able to balance the robot from such an unbalanced position with relatively large tilt and bend angles. The high energetic movements at the beginning of the motion are due to the fast swivelling motion which is required to correct the tilt angle. Comparing Figs. 5.17 and 5.18 it is clear that, during the first 0.2 second of the motion, q_4 and q_5 oscillate rapidly whereas ψ hardly changes which shows the fast swivelling motion of the robot. These three examples demonstrate that the controller is able to stabilize the robot in its upright balanced configuration from arbitrary stationary initial positions.

During a crouching motion, the objective of the controller is to stabilize the robot in an unstable balanced configuration at an arbitrary vertical robot plane, when the robot starts from an initial upright balanced configuration. Therefore, the command signal includes the desired bend angle (ψ_d) and the desired orientation of the robot plane (θ_d). Figs. 5.21 to 5.24 show the joint angles, the bend angle (ψ) and the robot plane angle (θ) during two crouching motions. In both of these motions, the robot starts from a configuration that is quite close to an upright balanced one (i.e. $q_2 = q_3 = q_5 = 0$ and $q_4 = 1e^{-4}$) and aims to balance at configurations which are characterized by $\psi_d = \frac{\pi}{2}$ and $\theta_d = \frac{\pi}{6}$ in the first example and $\psi_d = \frac{\pi}{2}$ and $\theta_d = \frac{\pi}{3}$ in the second one.

The observed discontinuity in the bend angle is due to it being defined as always positive and less than π (i.e. $0 \leq \psi \leq \pi$). Comparing Figs. 5.22 and 5.24, it is clear that the curves of the bend angle in both examples are the same. Also it can be seen that the behaviour of the bend angle is similar to the behaviour of q_2 of the 2D balancer during its crouching motion (Fig. 3.3).

Due to the initial conditions of the robot ($q_4 = q_6 = 1e^{-4}$), the robot plane angle starts from $\theta = \pi/2$ in both examples and then converges to its desired value quite rapidly. There are also sudden changes in θ when ψ is very close to zero (see Figs. 5.22 and 5.24) which is because the robot plane is not defined at $\psi = 0$. According to

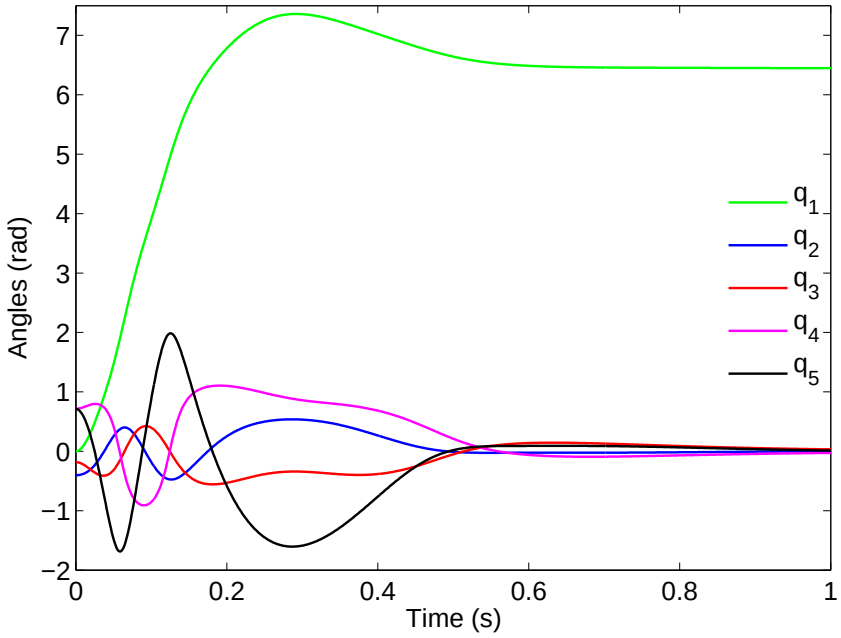


Figure 5.9: Joint angles in a straightening motion of the 3D balancer–1st example

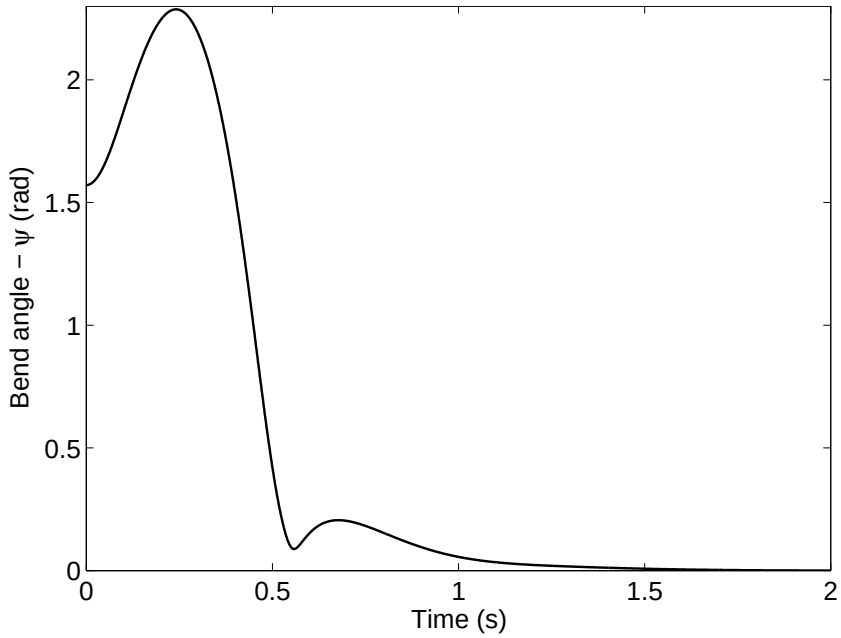


Figure 5.10: Bend angle in a straightening motion of the 3D balancer–1st example

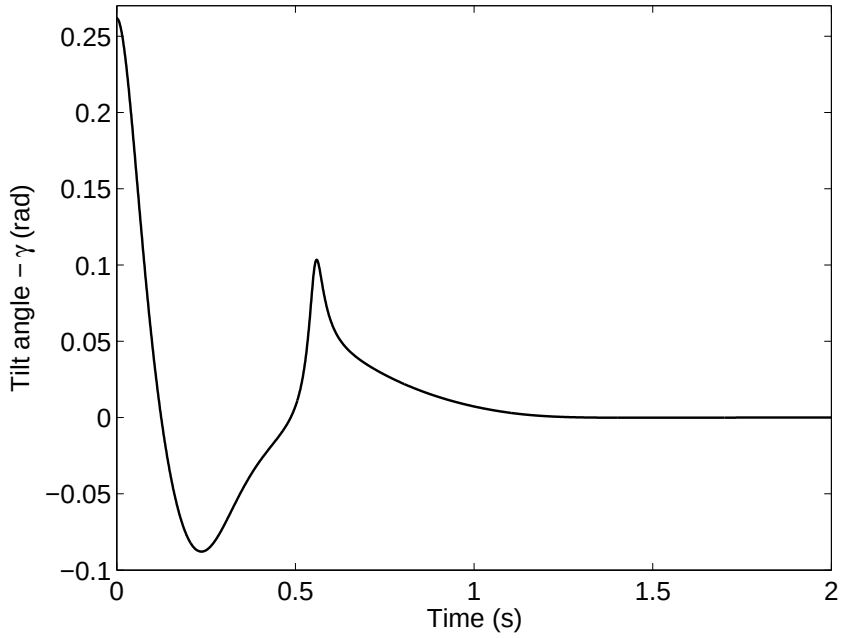


Figure 5.11: Tilt angle in a straightening motion of the 3D balancer–1st example

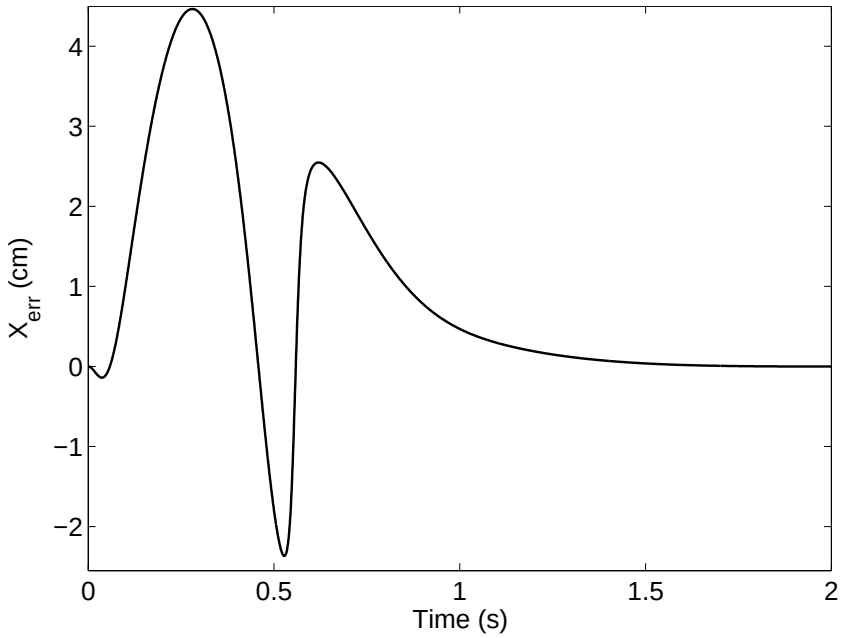


Figure 5.12: Balance error in a straightening motion of the 3D balancer–1st example

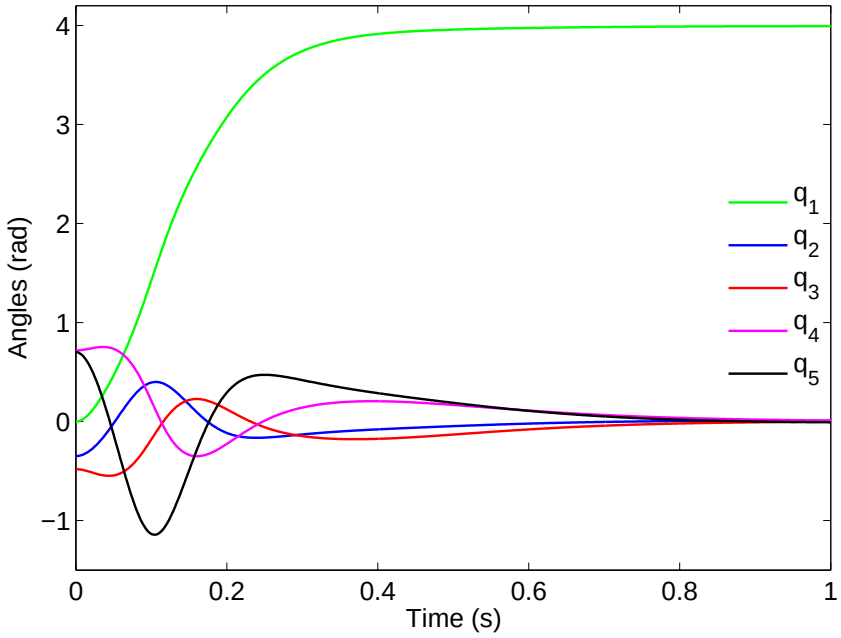


Figure 5.13: Joint angles in a straightening motion of the 3D balancer–2nd example

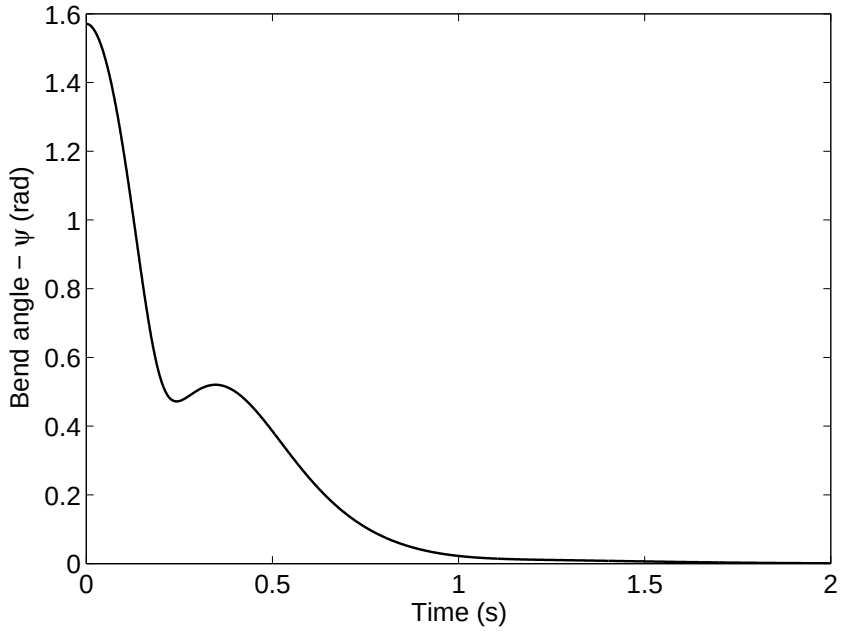


Figure 5.14: Bend angle in a straightening motion of the 3D balancer–2nd example

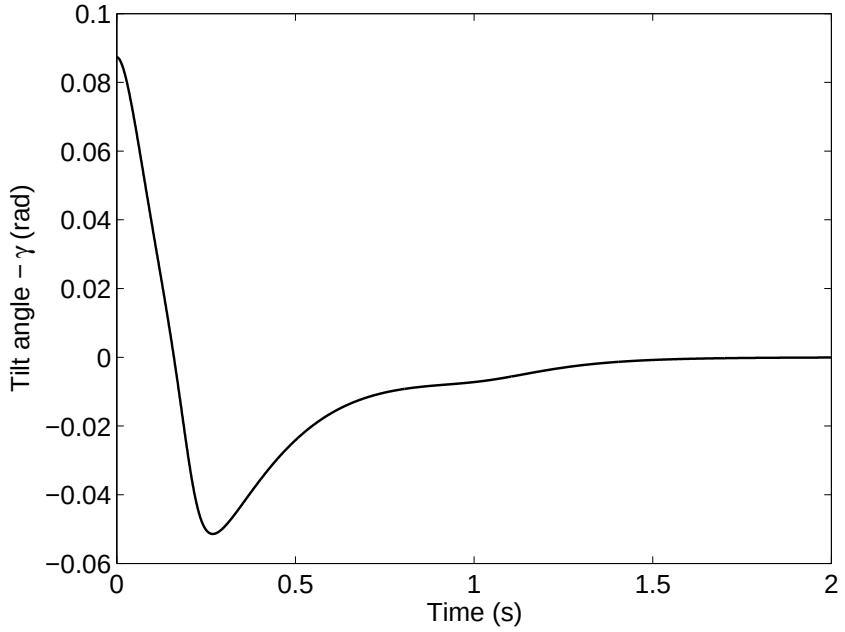


Figure 5.15: Tilt angle in a straightening motion of the 3D balancer–2nd example

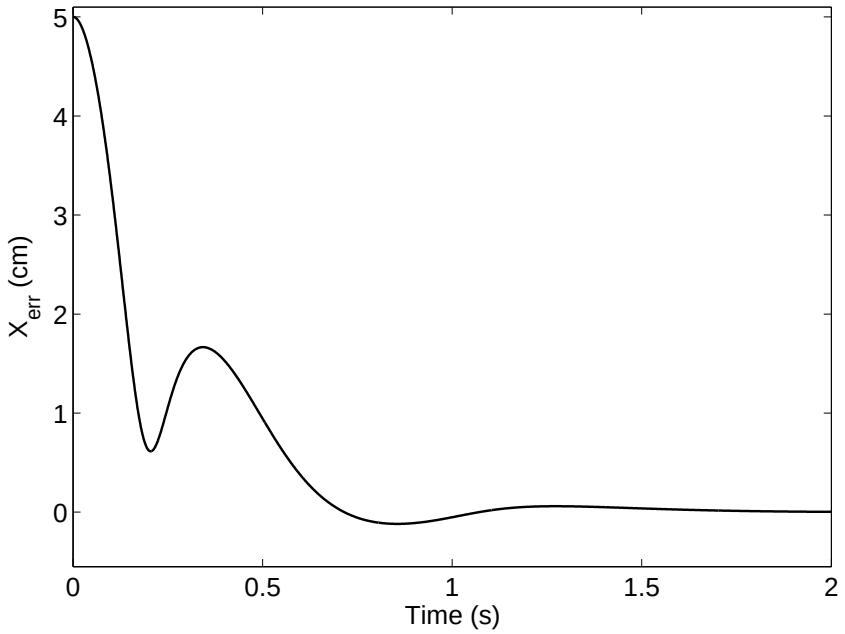


Figure 5.16: Balance error in a straightening motion of the 3D balancer–2nd example

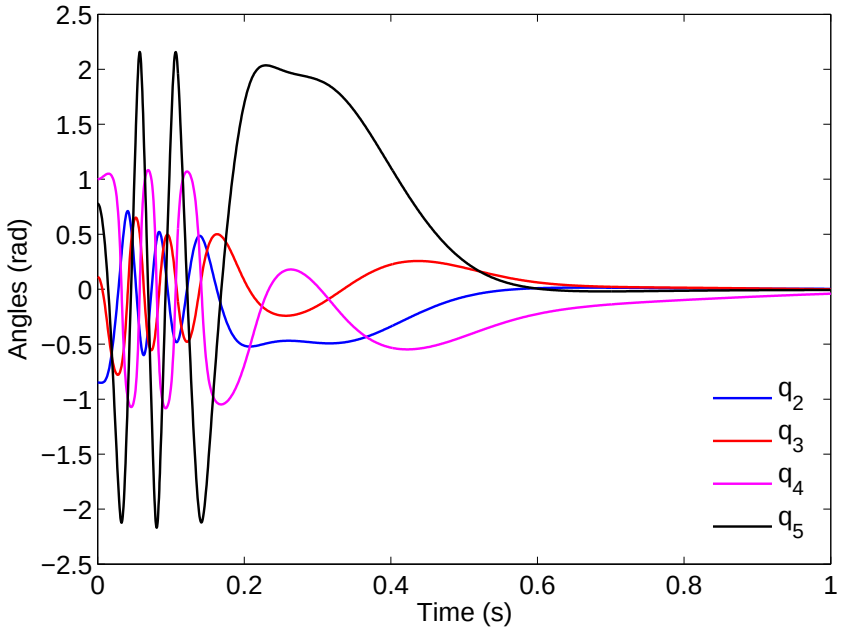


Figure 5.17: Joint angles in a straightening motion of the 3D balancer–3rd example

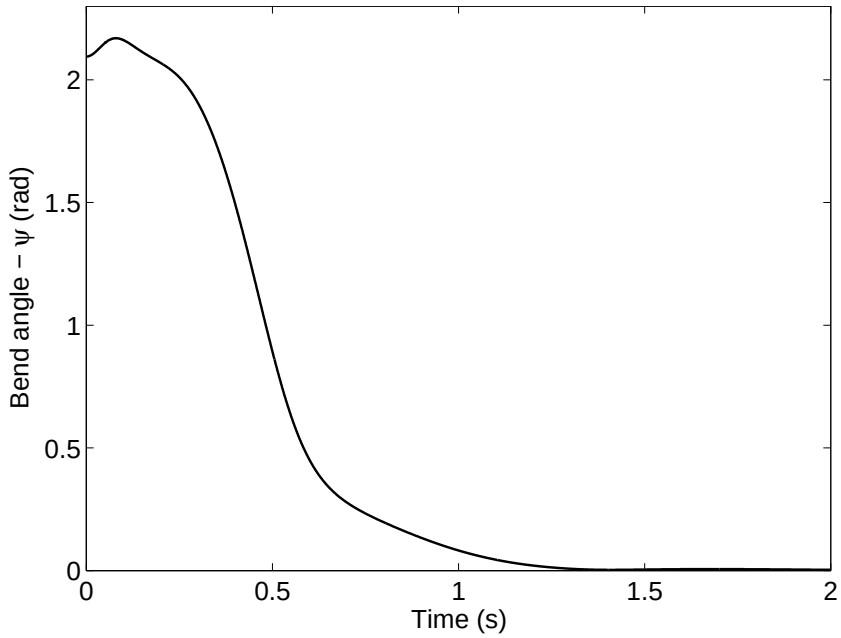


Figure 5.18: Bend angle in a straightening motion of the 3D balancer–3rd example

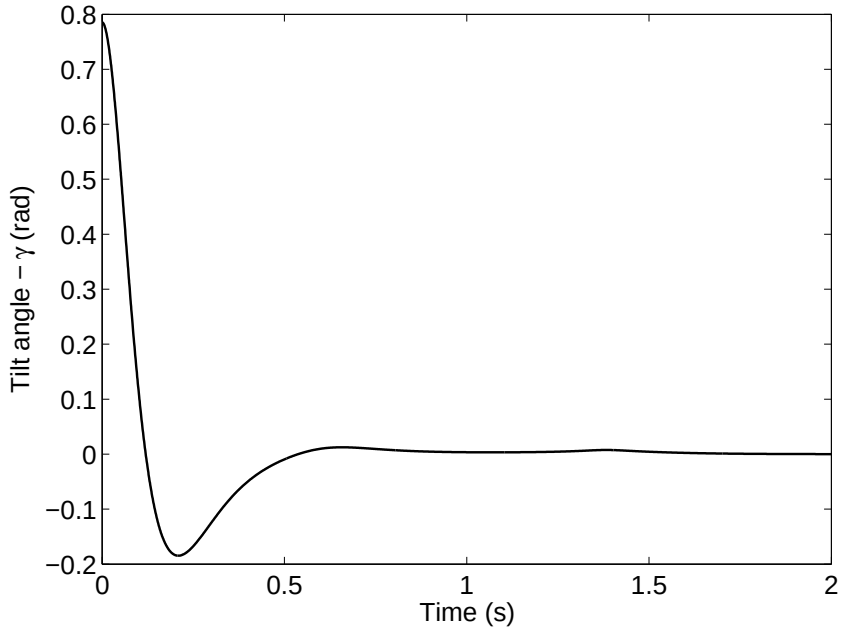


Figure 5.19: Tilt angle in a straightening motion of the 3D balancer–3rd example

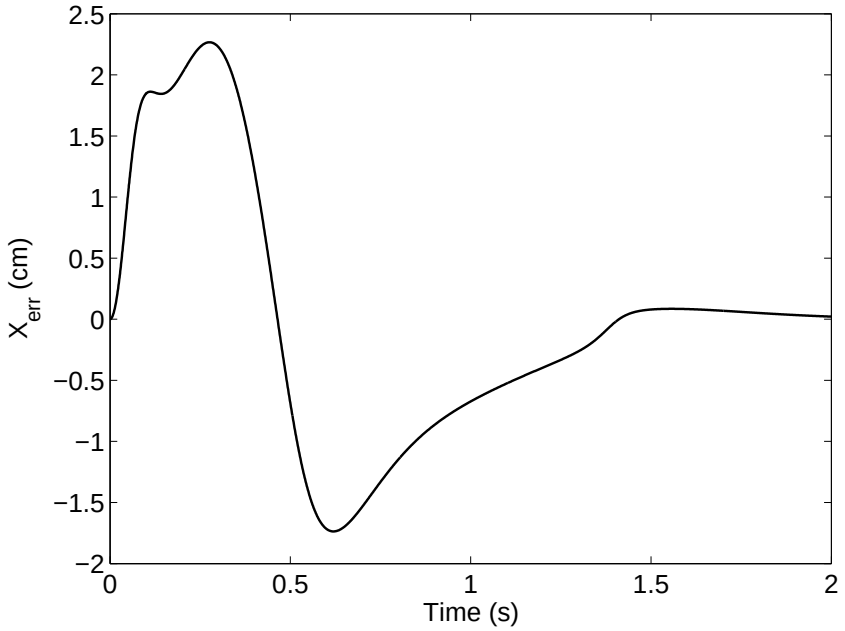


Figure 5.20: Balance error in a straightening motion of the 3D balancer–3rd example

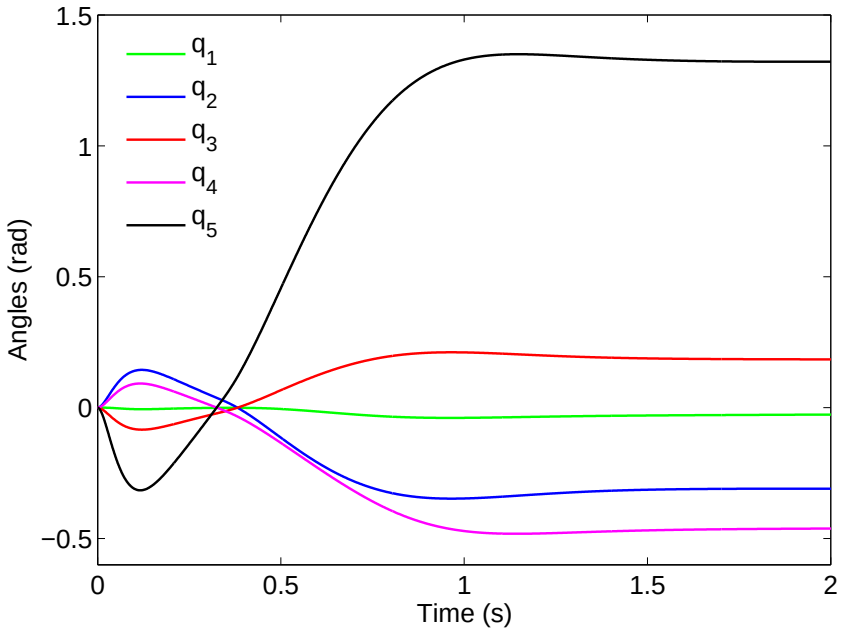


Figure 5.21: Joint angles in a crouching motion of the 3D balancer–1st example

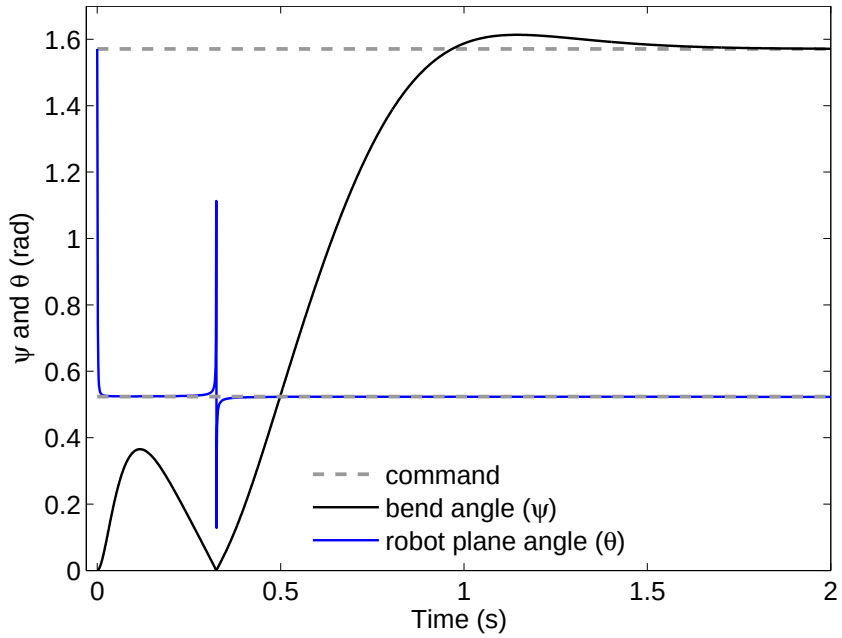


Figure 5.22: Bend and robot plane angles in a crouching motion of the 3D balancer–1st example

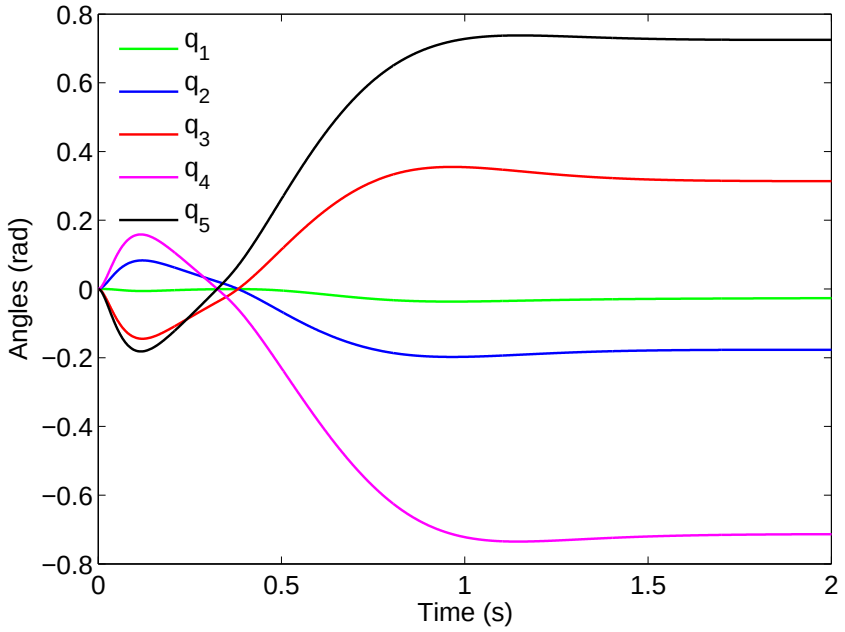


Figure 5.23: Joint angles in a crouching motion of the 3D balancer–2nd example

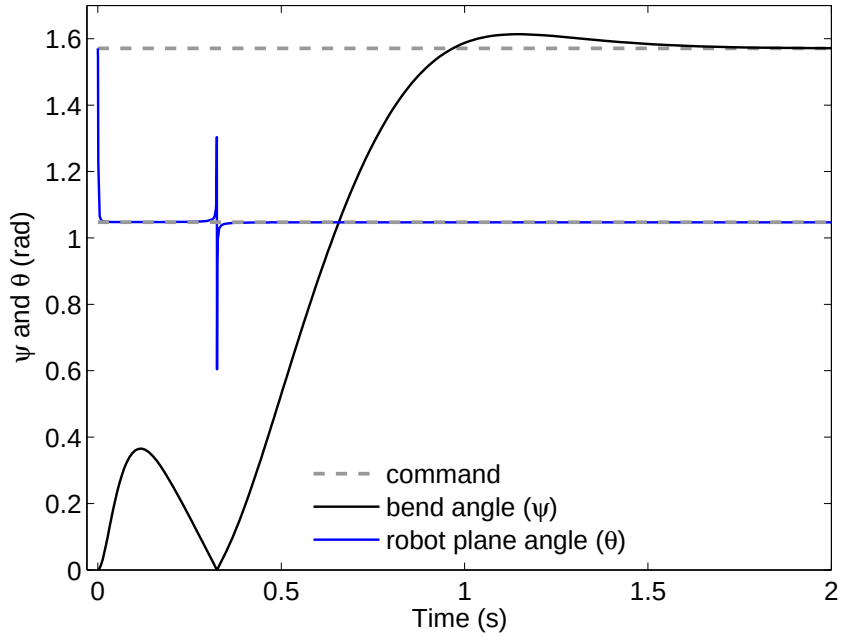


Figure 5.24: Bend and robot plane angles in a crouching motion of the 3D balancer–2nd example

Figs. 5.22 and 5.24, θ follows the command signal (θ_d) exactly and does not change during the crouching motion (except when ψ is very close to zero) which illustrates that the bending motion is completely decoupled from the swivelling motion. The controller can change the bend angle without affecting the orientation of the robot plane. Also Figs. 5.21 and 5.23 show that different values of the joint angles are required to crouch the robot in different planes with the same bend angles.

During a rotating motion, the controller aims to change the direction of the robot plane from an arbitrary vertical plane to any other vertical one, when the robot starts from an initial crouched balanced configuration. Figs. 5.25 and 5.26 show the bend and the robot plane angles while the 3D balancer is following a command signal including crouching, rotating and straightening motions. The command signal consists of two steps for the bend angle (including one at $t = 0$ and one at $t = 22$) and one step (at $t = 5$) and three ramps (with different slopes: 1rad/s, 0.5rad/s and 0.2rad/s) for the robot plane angle. The rate of change of θ is called rotating velocity of the robot in this chapter and is denoted by $\dot{\theta}$.

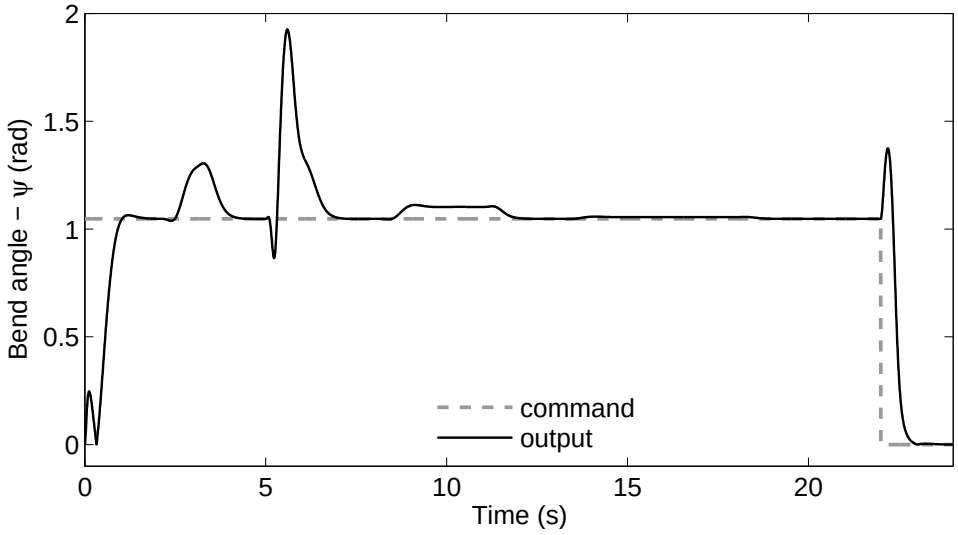


Figure 5.25: Bend angle in crouching, rotating and straightening motions of the 3D balancer

As can be seen in Figs. 5.25 and 5.26, although the trajectory following performance is reasonably good, there is always a time delay in following the command trajectories of θ. There are also some disturbances in the bend angle during the rotating motions of the robot. These disturbances are quite significant when the rotating velocity is high (i.e. the step command or ramps with $\dot{\theta} = 1$rad/s and even for $\dot{\theta} = 0.5$rad/s). The increase in the bend angle during the rotating motion suggests that a strong coupling exists between the bending and swivelling motions for high values of $\dot{\theta}$. However, at low rotating velocities (e.g. $\dot{\theta} = 0.2$rad/s), the bend angle remains almost unaffected (0.009rad steady-state error) implying that the coupling is very small.

Maximum errors of the bend angle due to the rotating motions are 0.011rad,

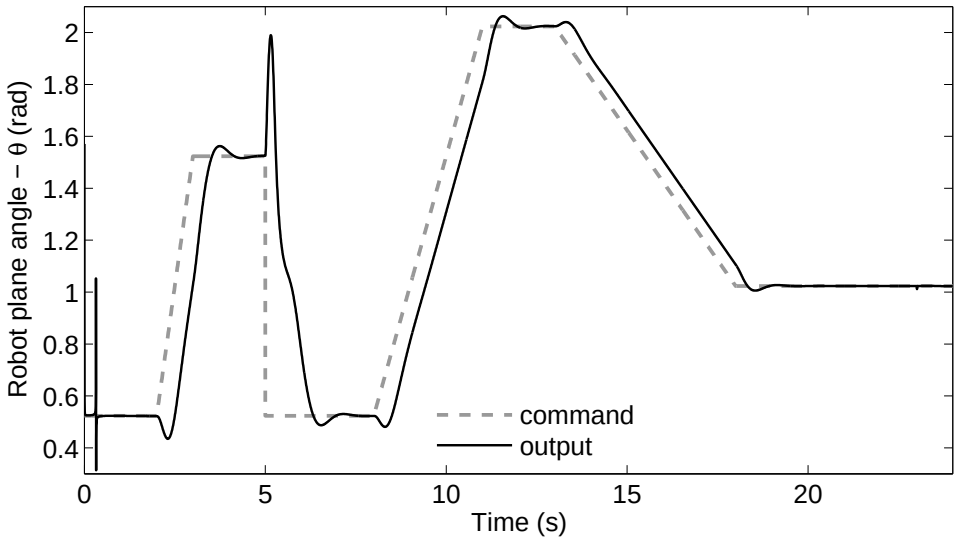


Figure 5.26: Robot plane angle in crouching, rotating and straightening motions of the 3D balancer

0.065rad and 0.258rad at $\dot{\theta} = 0.2\text{rad/s}$, $\dot{\theta} = 0.5\text{rad/s}$ and $\dot{\theta} = 1\text{rad/s}$, respectively. Comparing these errors, it can be seen that they vary with the square of the velocity of the rotating motion ($\dot{\theta}$). Thus, it can be concluded that the disturbances in the bend angle are functions of the square of the rotating velocity and therefore the square of the swivelling velocity. This remark verifies that the bending and the swivelling motions are instantaneously decoupled. Since the bending motion (crouching) does not affect the swivel angle it can be concluded that it is completely decoupled from the swivelling motion. However, according to the simulation results, the swivelling motion is coupled to the bending motion via velocity terms.

5.7 Summary

In this chapter a dynamic model of a 3D under-actuated robot with three degrees of under-actuation is introduced and a new control algorithm is presented to balance the robot. The active joint of the robot is specially designed to decouple the robot's motion instantaneously into bending and swivelling motions. Then an angular momentum based controller is proposed to balance the robot in the bend and swivel directions. The controller is able to stabilize the robot in any unstable balanced configuration at any arbitrary vertical plane, and also rotate the robot plane about the vertical axis without losing balance. Simulation results show the good performance of the controller in straightening, crouching and rotating motions. The last two motions, which are the results of decoupling the robot's motion, are demonstrated here for the first time. According to the simulations, the robot is able to recover its balance from an arbitrary initial unbalanced configuration and crouch at any arbitrary vertical plane. It will be able to crouch, launch and hop in any arbitrary vertical plane using the algorithm introduced in chapter 4.

Appendix

This appendix proves that the bending and the swivelling motions are instantaneously decoupled.[1] Let M_{BS} denote the generalized inertia matrix of the robot in bend-swivel space and τ_{BS} denote its corresponding generalized force vector as

$$\tau_{BS} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_b \\ \tau_s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \tau_b \\ \tau_s \end{bmatrix} , \tag{5.43}$$

where τ_b and τ_s are pure couples about bend and swivel axes, respectively, and τ_1, τ_2 and $\tau 3$ are zero because they belong to the passive joint of the robot. Therefore, if the robot is at rest (i.e. no velocity terms) and there is no gravity, the motion equations of the robot in bend-swivel space become

$$M_{BS}\ddot{q}_{BS} = \tau_{BS} \implies \ddot{q}_{BS} = M_{BS}^{-1}\tau_{BS} , \tag{5.44}$$

where q_{BS} is the matrix of the generalized coordinates of the robot in bend-swivel space as

$$q_{BS} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \psi \\ \sigma \end{bmatrix} . \tag{5.45}$$

The bending and swivelling motions are decoupled at the force-acceleration level if and only if τ_b and τ_s do not cause any acceleration in the swivel and bend directions, respectively. Since M_{BS} is a symmetric matrix, using (5.44) and knowing that $\tau_1 = \tau_2 = \tau_3 = 0$, it is clear that the bending and swivelling motions are instantaneously decoupled if and only if $M_{BS}^{-1}(4,5) = M_{BS}^{-1}(5,4) = 0$ where $M_{BS}^{-1}(i,j)$ is the element in the ith row and jth column of the matrix. Note that, because of the symmetry in the inertia matrix, $M_{BS}^{-1}(4,5)$ is always equal to $M_{BS}^{-1}(5,4)$ implying that if τ_b does not cause any acceleration in the swivel direction then τ_s does not cause any acceleration in the bend direction either.

Fig. 5.27 shows the robot in the robot plane. The robot has a mirror symmetry about this plane which implies that the centers of mass of each link and therefore the CoM of the robot are always in this plane. Also one of the principal axes of inertia of each link is perpendicular to the plane, and so the other principal axes are all in the plane. The spherical joint is also symmetrical with respect to the plane. Since the bend axis ($\tilde{B}$) is always perpendicular to the robot plane, τ_b is an in-plane couple and therefore it causes acceleration in the bend direction only and nothing in the swivel direction (i.e. $\ddot{\sigma} = 0$). This concludes the proof.

[1] This proof is due to Dr. Roy Featherstone.

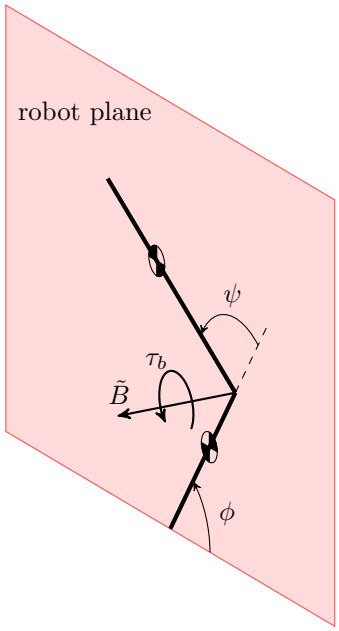


Figure 5.27: Robot in the robot plane

Remark: The proof relies on the robot being physically capable of a pure bending motion. The double Cardan joint guarantees that a pure bending motion (and also a pure swivelling motion) is always possible for any value of q_4 and q_5; but simpler joints, such as the universal joint, do not have this property. This is why it is necessary for the balancer to have a double Cardan joint at the knee.

Chapter 6

Conclusion

6.1 Thesis Contributions

The following are the main contributions of the thesis:

- A new angular momentum based controller for the balancing motion of an under-actuated planar robot. The controller is able to stabilize the robot at any unstable balanced configuration in which the robot is controllable and to control the robot to follow setpoint and motion trajectory commands. It is also able to control the balancing motion of the robot with rolling contact even in the presence of sudden changes in the curvature of the foot.

- A new non-linear normal force model for the contact between a sphere and the ground. The new model is different from the previous ones only in the damping term. This model is able to accurately calculate the coefficient of restitution of the contact between spheres and plates of various materials. The new normal contact model has been tested against real experimental data and found to be more accurate than any of the other models it was compared with, which includes the widely-used Hunt/Crossley model. This model can be used in modelling the contact force between a legged robot's foot and the ground.

- A new control strategy for a single-hop motion of a knee-leg hopping robot. The robot is in fact a two-link robot with one actuator in the knee joint. During a single-hop motion the robot starts from its upright balanced configuration and after crouching, launching and flight phases the robot lands at a desired location and rebalance itself at its upright configuration.

- A new angular momentum based controller for balancing motion of a spatial under-actuated robot. The robot consists of two links connected to each other by an active two DoF joint (knee). The lower link is connected to the ground via a passive spherical joint. The robot has five DoF and is controlled by using only two actuators in the knee joint. Using a constant velocity joint at the knee, decouples the robot's motion instantaneously into bending and swivelling motions (i.e. invented by Dr. Roy Featherstone) which are controlled separately using the proposed control algorithm. The new controller is able to balance

the robot at any unstable balanced configuration at any arbitrary vertical plane. It is also able to rotate the vertical plane of the robot about the vertical axis. Using this controller it is possible to confine the robot's motion to any arbitrary vertical plane. Thus, the hopping motion of the 3D robot becomes a planar problem.

6.2 Future Work

Some possible extensions and research directions are summarized below:

- Solving the sensitivity problem of the planar balance controller to the bias in the sensors (or errors that influence the balanced configuration). This is a common problem in all previous controllers in the literature.
- Extending the planar balance controller to control under-actuated robots with multiple links. This can be used in controlling the balancing motion of multi-legged robots or humanoids.
- Finding a control strategy for continuous hopping motion of the 2D hopper. So the controller would be able to calculate proper (or optimal) hop length between the robot's location and a desired one and perform continuous hopping motions to move the robot to the desired location.
- Modelling the motion of the 3D balancer robot in swivel direction and finding optimal gains for the swivel controller. Also a new controller for its swivelling motion can be derived using the motion equations in swivel direction.
- Building the 2D hopper and the 3D balancer and implementing the proposed controllers on real robots.

Bibliography

Abdallah, M. and Waldron, K., 2009. The mechanics of biped running and a stable control strategy. *Robotica*, 27, 5 (2009), 789. (cited on page 4)

Abdallah, M. E. and Waldron, K. J., 2007. A physical model and control strategy for biped running. In *Robotics and Automation, 2007 IEEE International Conference on*, 3982–3988. IEEE. (cited on page 4)

Ahmadi, M. and Buehler, M., 1995. A control strategy for stable passive running. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 3, 152–157. IEEE. (cited on page 4)

Ahmadi, M. and Buehler, M., 1997. Stable control of a simulated one-legged running robot with hip and leg compliance. *Robotics and Automation, IEEE Transactions on*, 13, 1 (1997), 96–104. (cited on page 4)

Ahmadi, M. and Buehler, M., 2006. Controlled passive dynamic running experiments with the arl-monopod ii. *Robotics, IEEE Transactions on*, 22, 5 (2006), 974–986. (cited on page 4)

Altendorfer, R.; Koditschek, D.; and Holmes, P., 2004. Stability analysis of legged locomotion models by symmetry-factored return maps. *The International Journal of Robotics Research*, 23, 10-11 (2004), 979–999. (cited on page 4)

Amagata, Y.; Nakaura, S.; and Sampei, M., 2008. The running of humanoid robot on uneven terrain utilizing output zeroing. In *SICE Annual Conference, 2008*, 2841–2846. IEEE. (cited on page 5)

Ankarali, M. and Saranli, U., 2010. Analysis and control of a dissipative spring-mass hopper with torque actuation. In *Proceedings of Robotics: Science and Systems, Zaragoza, Spain*. (cited on page 4)

Armstrong-Hélouvry, B.; Dupont, P.; and De Wit, C., 1994. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30, 7 (1994), 1083–1138. (cited on page 18)

Azad, M. and Featherstone, R., 2010. Modeling the contact between a rolling sphere and a compliant ground plane. In *Australasian Conf. Robotics and Automation, Brisbane, Australia*. (cited on page 9)

Azad, M. and Featherstone, R., 2012. Angular momentum based controller for balancing an inverted double pendulum. In *RoManSy 2012, Paris, France*. (cited on page 26)

Azad, M. and Featherstone, R., 2013. Balancing and hopping motion of a planar hopper with one actuator. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2027–2032. IEEE. (cited on page 57)

Berkemeier, M. and Fearing, R., 1992. Control of a two-link robot to achieve sliding and hopping gaits. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, 286–291. IEEE. (cited on pages xiii and 6)

Berkemeier, M. and Fearing, R., 1998. Sliding and hopping gaits for the underactuated acrobot. *Robotics and Automation, IEEE Transactions on*, 14, 4 (1998), 629–634. (cited on pages 5, 6, 7, 57, and 59)

Berkemeier, M. and Fearing, R., 1999. Tracking fast inverted trajectories of the underactuated acrobot. *Robotics and Automation, IEEE Transactions on*, 15, 4 (1999), 740–750. (cited on pages 5, 7, 33, 42, and 44)

Bliman, P. and Sorine, M., 1995. Easy-to-use realistic dry friction models for automatic control. In *Proceedings of 3rd European Control Conference, Rome, Italy*, 3788–3794. (cited on page 18)

Brown, S. C. and Passino, K. M., 1997. Intelligent control for an acrobot. *Journal of Intelligent and Robotic Systems*, 18, 3 (1997), 209–248. (cited on pages 7 and 44)

Canudas de Wit, C.; Olsson, H.; Astrom, K.; and Lischinsky, P., 1995. A new model for control of systems with friction. *Automatic Control, IEEE Transactions on*, 40, 3 (1995), 419–425. (cited on page 18)

Cherouvim, N. and Papadopoulos, E., 2009. Control of hopping speed and height over unknown rough terrain using a single actuator. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2743–2748. IEEE. (cited on page 4)

Chevallereau, C. and Aoustin, Y., 2001. Optimal reference trajectories for walking and running of a biped robot. *Robotica*, 19, 5 (2001), 557–569. (cited on page 5)

Chevallereau, C.; Westervelt, E.; and Grizzle, J., 2005. Asymptotically stable running for a five-link, four-actuator, planar bipedal robot. *The International Journal of Robotics Research*, 24, 6 (2005), 431–464. (cited on page 5)

Cho, B. and Oh, J., 2008. Running pattern generation of humanoid biped with a fixed point and its realization. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, 299–305. IEEE. (cited on page 6)

Cho, B. and Oh, J., 2009. Running pattern generation of humanoid biped in the three-dimensional space and its realization. In *Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on*, 142–149. IEEE. (cited on page 6)

Dahl, P., 1968. A solid friction model. Technical report, DTIC Document. (cited on page 18)

Dupont, P.; Armstrong, B.; and Hayward, V., 2000. Elasto-plastic friction model: contact compliance and stiction. In *American Control Conference, 2000. Proceedings of the 2000, Chicago, Illinois*, vol. 2, 1072–1077. IEEE. (cited on page 18)

Dupont, P.; Hayward, V.; Armstrong, B.; and Altpeter, F., 2002. Single state elastoplastic friction models. *Automatic Control, IEEE Transactions on*, 47, 5 (2002), 787–792. (cited on page 18)

Falcon, E.; Laroche, C.; Fauve, S.; and Coste, C., 1998. Behavior of one inelastic ball bouncing repeatedly off the ground. *The European Physical Journal B-Condensed Matter and Complex Systems*, 3, 1 (1998), 45–57. (cited on page 10)

Featherstone, R., 2008. *Rigid body dynamics algorithms*. Springer NewYork, USA:. (cited on pages 18, 75, and 81)

Featherstone, R., 2012. Analysis and design of planar self-balancing double-pendulum robots. In *Romansy 19–Robot Design, Dynamics and Control*, 259–266. Springer. (cited on pages 30, 31, 34, and 48)

Fernandes, R.; Akinfiev, T.; and Armada, A., 2009. Control of the hop height of a one-legged resonance robot. *Automation and Remote Control*, 70, 1 (2009), 64–73. (cited on page 3)

François, C. and Samson, C., 1998. A new approach to the control of the planar one-legged hopper. *The International Journal of Robotics Research*, 17, 11 (1998), 1150–1166. (cited on page 4)

Fujimoto, Y., 2004. Trajectory generation of biped running robot with minimum energy consumption. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, 3803–3808. IEEE. (cited on page 4)

Gilardi, G. and Sharf, I., 2002. Literature survey of contact dynamics modelling. *Mechanism and machine theory*, 37, 10 (2002), 1213–1239. (cited on page 9)

Goldsmith, W., 1960. *Impact: the theory and physical behaviour of colliding solids*. E. Arnold (London). (cited on pages 10 and 15)

Grizzle, J.; Hurst, J.; Morris, B.; Park, H.; and Sreenath, K., 2009. Mabel, a new robotic bipedal walker and runner. In *American Control Conference, 2009. ACC'09.*, 2030–2036. IEEE. (cited on page 5)

Grizzle, J.; Moog, C.; and Chevallereau, C., 2005. Nonlinear control of mechanical systems with an unactuated cyclic variable. *Automatic Control, IEEE Transactions on*, 50, 5 (2005), 559–576. (cited on pages 6, 7, 25, and 42)

Ha, Y. and Yuta, S., 1994. Trajectory tracking control for navigation of self-contained mobile inverse pendulum. In *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, vol. 3, 1875–1882. IEEE. (cited on page 45)

Haessig Jr, D. and Friedland, B., 1991. On the modeling and simulation of friction. *Journal of Dynamic Systems, Measurement, and Control*, 113 (1991), 354–362. (cited on page 18)

Harbick, K. and Sukhatme, G., 2001a. Height control for a one-legged hopping robot using a one-dimensional model. Technical report, Institute for Robotics and Intelligent Systems. USC. (cited on page 3)

Harbick, K. and Sukhatme, G., 2001b. Height control for a one-legged hopping robot using a two-dimensional model. Technical report, Institute for Robotics and Intelligent Systems. USC. (cited on page 4)

Harnefors, L. and Nee, H.-P., 2000. A general algorithm for speed and position estimation of ac motors. *Industrial Electronics, IEEE Transactions on*, 47, 1 (2000), 77–83. (cited on page 38)

Hattori, K.; Yamaura, H.; and Ono, K., 2004. Torque-based aerial posture control of a two-link acrobat robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 218, 7 (2004), 595–601. (cited on page 6)

Hauser, J. and Murray, R., 1990. Nonlinear controllers for non-integrable systems: The acrobot example. In *American Control Conference, SanDiego, CA*, 669–671. IEEE. (cited on pages 7 and 25)

He, G. and Geng, Z., 2009. Exponentially stabilizing an one-legged hopping robot with non-slip model in flight phase. *Mechatronics*, 19, 3 (2009), 364–374. (cited on page 4)

He, G.; Tan, X.; Zhang, X.; and Lu, Z., 2008. Modeling, motion planning, and control of one-legged hopping robot actuated by two arms. *Mechanism and Machine Theory*, 43, 1 (2008), 33–49. (cited on page 4)

Hodgins, J. and Raibert, M., 1990. Biped gymnastics. *The International Journal of Robotics Research*, 9, 2 (1990), 115–128. (cited on page 3)

Hollis, R., 2008. Ballbots. *Special Editions*, 18, 1 (2008), 58–63. (cited on page 45)

Hunt, K. and Crossley, F., 1975. Coefficient of restitution interpreted as damping in vibroimpact. *J. Appl. Mech.*, 42, 2 (1975), 440–445. (cited on page 10)

Hyon, S. and Emura, T., 2004a. Energy-preserving control of a passive one-legged running robot. *Advanced Robotics*, 18, 4 (2004), 357–381. (cited on page 4)

Hyon, S. and Emura, T., 2004b. Running control of a planar biped robot based on energy-preserving strategy. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, 3791–3796. IEEE. (cited on page 4)

Iida, F.; Dravid, R.; and Paul, C., 2002. Design and control of a pendulum driven hopping robot. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, 2141–2146. IEEE. (cited on page 4)

Inoue, A.; Deng, M.; Hara, S.; and Henmi, T., 2007. Swing-up and stabilizing control system design for an acrobot. In *Networking, Sensing and Control, 2007 IEEE International Conference on*, 559–561. IEEE. (cited on pages 7 and 44)

Johnson, K., 1977. *Contact mechanics*. Cambridge university press. (cited on pages 9 and 13)

Jung, S. and Kim, S. S., 2008. Control experiment of a wheel-driven mobile inverted pendulum using neural network. *Control Systems Technology, IEEE Transactions on*, 16, 2 (2008), 297–303. (cited on page 45)

Kajita, S.; Kaneko, K.; Morisawa, M.; Nakaoka, S.; and Hirukawa, H., 2007a. Zmp-based biped running enhanced by toe springs. In *Robotics and Automation, 2007 IEEE International Conference on*, 3963–3969. IEEE. (cited on page 6)

Kajita, S.; Nagasaki, T.; Kaneko, K.; and Hirukawa, H., 2007b. Zmp-based biped running control. *Robotics & Automation Magazine, IEEE*, 14, 2 (2007), 63–72. (cited on page 6)

Kawabara, G. and Kono, K., 1987. Restitution coefficient in a collision between two spheres. *Jpn. J. Appl. Phys. 1*, 26, 8 (1987), 1230–1233. (cited on pages 10 and 15)

Kawaguchi, M. and Yamakita, M., 2011. Stabilizing of bike robot with variable configured balancer. In *SICE Annual Conference (SICE), 2011 Proceedings of*, 1057–1062. IEEE. (cited on page 7)

Kawamura, A. and Zhu, C., 2006. The development of biped robot mari-3 for fast walking and running. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, 599–604. IEEE. (cited on page 6)

Lai, X.; Wu, Y.; She, J.; and Wu, M., 2005. Control design and comprehensive stability analysis of acrobots based on lyapunov functions. *Journal of Central South University of Technology*, 12, 1 (2005), 210–216. (cited on pages 7 and 44)

Lankarani, P. and Nikravesh, H., 1990. A contact force model with hysteresis damping for impact analysis of multi-body systems. *J. Mechanical Design*, 112, 3 (1990), 369–376. (cited on page 10)

Lauwers, T.; Kantor, G. A.; and Hollis, R. L., 2006. A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, 2884–2889. IEEE. (cited on page 45)

Li, Z. and He, J., 1990. An energy perturbation approach to limit cycle analysis in legged locomotion systems. In *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*, 1989–1994. IEEE. (cited on page 3)

Li, Z. and Montgomery, R., 1990. Dynamics and optimal control of a legged robot in flight phase. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 1816–1821. IEEE. (cited on page 4)

Mahindrakar, A. D. and Banavar, R. N., 2005. A swing-up of the acrobot based on a simple pendulum strategy. *International Journal of Control*, 78, 6 (2005), 424–429. (cited on pages 7 and 44)

Marhefka, D. and Orin, D., 1999. A compliant contact model with nonlinear damping for simulation of robotic systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 29, 6 (1999), 566–572. (cited on pages 10 and 15)

M'Closkey, R. and Burdick, J., 1991. An analytical study of simple hopping robots with vertical and forward motion. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, 1392–1397. IEEE. (cited on page 4)

M'Closkey, R. and Burdick, J., 1993. Periodic motions of a hopping robot with vertical and forward motion. *The International journal of robotics research*, 12, 3 (1993), 197–218. (cited on page 4)

Michalska, H.; Ahmadi, M.; and Buehler, M., 1996. Vertical motion control of a hopping robot. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3, 2712–2717. IEEE. (cited on page 3)

Mita, T.; Hyon, S.; and Nam, T., 2001. Analytical time optimal control solution for a two-link planar aerobot with initial angular momentum. *Robotics and Automation, IEEE Transactions on*, 17, 3 (2001), 361–366. (cited on page 6)

Miyashita, N.; Kishikawa, M.; and Yamakita, M., 2006. 3d motion control of 2 links (5 dof) underactuated manipulator named acrobox. In *American Control Conference, Minneapolis, Minnesota, USA*. IEEE. (cited on page 71)

Morris, B.; Westervelt, E.; Chevallereau, C.; Buche, G.; and Grizzle, J., 2006. Achieving bipedal running with rabbit: Six steps toward infinity. *Fast Motions in Biomechanics and Robotics*, (2006), 277–297. (cited on page 5)

Muskinja, N. and Tovornik, B., 2006. Swinging up and stabilization of a real inverted pendulum. *Industrial Electronics, IEEE Transactions on*, 53, 2 (2006), 631–639. (cited on page 45)

Nagarajan, U.; Kantor, G.; and Hollis, R. L., 2009. Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 3743–3748. IEEE. (cited on page 45)

Nakajima, R.; Tsubouchi, T.; Yuta, S.; and Koyanagi, E., 1997. A development of a new mechanism of an autonomous unicycle. In *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2, 906–912. IEEE. (cited on page 45)

Nam, T.; Fukuhara, Y.; Mita, T.; and Yamakita, M., 2002. Swing-up control and avoiding singular problem of an acrobot system. In *SICE 2002. Proceedings of the 41st SICE Annual Conference*, vol. 5, 2990–2995. IEEE. (cited on page 7)

Ohashi, E. and Ohnishi, K., 2006. Hopping height control for hopping robots. *Electrical Engineering in Japan*, 155, 1 (2006), 64–71. (cited on page 4)

Okawa, A.; Keo, L.; and Yamakita, M., 2009. Realization of acrobatic turn via wheelie for a bicycle with a balancer. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2965–2970. IEEE. (cited on page 7)

Olfati-Saber, R., 2000. Control of underactuated mechanical systems with two degrees of freedom and symmetry. In *American Control Conference, 2000. Proceedings of the 2000*, vol. 6, 4092–4096. IEEE. (cited on page 7)

Olfati-Saber, R., 2002. Normal forms for underactuated mechanical systems with symmetry. *Automatic Control, IEEE Transactions on*, 47, 2 (2002), 305–308. (cited on page 7)

Olfati-Saber, R. and Megretski, A., 1998. Controller design for a class of underactuated nonlinear systems. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 4, 4182–4187. IEEE. (cited on page 7)

Olsson, H.; Åström, K.; Canudas de Wit, C.; Gäfvert, M.; and Lischinsky, P., 1998. Friction models and friction compensation. *European journal of control*, 4 (1998), 176–195. (cited on page 18)

Park, J. and Kwon, O., 2003. Impedance control for running of biped robots. In *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on*, vol. 2, 944–949. IEEE. (cited on page 6)

Playter, R. and Raibert, M., 1992. Control of a biped somersault in 3d. In *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, vol. 1, 582–589. IEEE. (cited on pages 3 and 6)

Poulakakis, I. and Grizzle, J., 2007. Monopedal running control: Slip embedding and virtual constraint controllers. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 323–330. IEEE. (cited on page 5)

Poulakakis, I. and Grizzle, J., 2009a. Modeling and control of the monopedal robot thumper. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 3327–3334. IEEE. (cited on page 5)

Poulakakis, I. and Grizzle, J., 2009b. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *Automatic Control, IEEE Transactions on*, 54, 8 (2009), 1779–1793. (cited on page 5)

Rad, H.; Gregorio, P.; and Buehler, M., 1993. Design, modeling and control of a hopping robot. In *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 3, 1778–1785. IEEE. (cited on page 4)

Raibert, M., 1986. *Legged robots that balance*. Massachusetts Institute of Technology. (cited on pages xiii, 2, 3, 4, and 6)

Raibert, M.; Brown, H.; and Chepponis, M., 1984. Experiments in balance with a 3d one-legged hopping machine. *The International Journal of Robotics Research*, 3, 2 (1984), 75–92. (cited on pages 2 and 6)

Raibert, M. and Brown Jr, H., 1984. Experiments in balance with a 2d one-legged hopping machine. *ASME Transactions Journal of Dynamic Systems and Measurement Control B*, 106 (1984), 75–81. (cited on page 2)

Raibert, M.; Chepponis, M.; and Brown Jr, H., 1986. Running on four legs as though they were one. *Robotics and Automation, IEEE Journal of*, 2, 2 (1986), 70–82. (cited on page 3)

Sayyad, A.; Seth, B.; and Seshu, P., 2007. Single-legged hopping robotics researchâĂŤa review: A review. *Robotica*, 25, 5 (2007), 587–613. (cited on page 3)

Schwind, W. and Koditschek, D., 1995. Control of forward velocity for a simplified planar hopping robot. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 1, 691–696. IEEE. (cited on page 4)

Seipel, J. and Holmes, P., 2005a. Running in three dimensions: Analysis of a point-mass sprung-leg model. *The International Journal of Robotics Research*, 24, 8 (2005), 657–674. (cited on page 6)

Seipel, J. and Holmes, P., 2006. Three-dimensional translational dynamics and stability of multi-legged runners. *The International Journal of Robotics Research*, 25, 9 (2006), 889–902. (cited on page 6)

Seipel, J. E., 2005. The stability of point-mass hoppers with varying morphology and minimal feedback. In *Robotics: Science and Systems*, 305–310. (cited on page 6)

Seipel, J. E. and Holmes, P. J., 2005b. Three-dimensional running is unstable but easily stabilized. In *Climbing and Walking Robots*, 585–592. Springer. (cited on page 6)

Shimizu, T.; Nakaura, S.; and Sampei, M., 2006. The control of a bipedal running robot based on output zeroing considered rotation of the ankle joint. In *Decision and Control, 2006 45th IEEE Conference on*, 6456–6461. IEEE. (cited on page 5)

Spong, M., 1995. The swing up control problem for the acrobot. *Control Systems, IEEE*, 15, 1 (1995), 49–55. (cited on pages 7, 42, and 44)

Spong, M. W., 1994. Swing up control of the acrobot. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, 2356–2361. IEEE. (cited on pages 7 and 44)

Sung, S. and Youm, Y., 2007. Landing motion control of articulated hopping robot. *International Journal of Advanced Robotic Systems*, 4, 3 (2007), 303–312. (cited on page 4)

Takahashi, T.; Yamakita, M.; and Hyon, S., 2006. An optimization approach for underactuated running robot. In *SICE-ICASE, 2006. International Joint Conference*, 3505–3510. IEEE. (cited on page 4)

Ur-Rehman, F., 2005. Steering control of a hopping robot model during the flight phase. *IEE proceedings. Control theory and applications*, 152, 6 (2005), 645–653. (cited on page 4)

Vakakis, A.; Burdick, J.; and Caughey, T., 1991. An" interesting" strange attractor in the dynamics of a hopping robot. *The International journal of robotics research*, 10, 6 (1991), 606–618. (cited on page 3)

Vermeulen, J.; Lefeber, D.; and Verrelst, B., 2003. Control of foot placement, forward velocity and body orientation of a one-legged hopping robot. *Robotica*, 21, 1 (2003), 45–57. (cited on page 4)

Wu, A. and Geyer, H., 2013. The 3-d spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments. *IEEE Trans on Robotics*, 29, 5 (October 2013), 1114–1124. (cited on page 6)

Xin, X. and Kaneda, M., 2001. A new solution to the swing up control problem for the acrobot. In *SICE 2001. Proceedings of the 40th SICE Annual Conference. International Session Papers*, 124–129. IEEE. (cited on pages 7 and 44)

Xin, X.; Mita, T.; and Kaneda, M., 2004. The posture control of a two-link free flying acrobot with initial angular momentum. *Automatic Control, IEEE Transactions on*, 49, 7 (2004), 1201–1206. (cited on page 6)

Xinjilefu, X.; Hayward, V.; and Michalska, H., 2009. Stabilization of the spatial double inverted pendulum using stochastic programming seen as a model of standing postural control. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, 367–372. IEEE. (cited on page 71)

Yamakita, M.; Yonemura, T.; Michitsuji, Y.; and Luo, Z., 2002. Stabilization of acrobat robot in upright position on a horizontal bar. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3, 3093–3098. IEEE. (cited on pages 7 and 71)

Yonemura, T. and Yamakita, M., 2004. Swing up control of acrobot based on switched output functions. In *SICE 2004 Annual Conference*, vol. 3, 1909–1914. IEEE. (cited on pages 7 and 71)