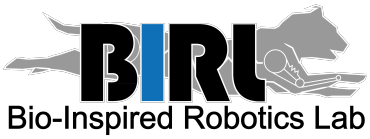
Master's Thesis

Development and Analysis of a Self-Stable, One-Legged Hopping Robot

Author:
Fabio Giardina
fabiogi@ethz.ch

Supervisors:
Fabian Guenther
Prof. Dr. Fumiya Iida



Abstract

In this thesis, a new type of hopping robot with a curved foot has been developed. The robot is self-stable and is operated using feedforward control. By building a detailed physical model using the Newton-Euler equations and implementing it numerically, the robot's abilities and dynamics were investigated. The ground contact model was realized by using a Newtonian kinematic impact- and a Coulomb frictional law. Subsequently, a prototype of the robot was planned and built. Experimental verification showed matching results with the simulation. Parameter analyses based on a series of simulations were performed, and revealed the intrinsic dynamical nature of the non-linear system. The jumping angle of the robot's center of mass at take-off was found to be a highly useful parameter in terms of efficiency- and stability analysis. Furthermore, stable hopping patterns with a small standard deviation of the take-off angle showed the highest locomotion efficiency. Finally, a geometrical parameter for feedforward speed control was discovered and is presented in this thesis.

Acknowledgements

An interesting Master's thesis which covers my interests, where I can pursue my ideas, and where I am supported by knowledgeable supervisors is not easy to find. When coming to the bio-inspired robotics laboratory however, it quickly became clear that this is the right place to dive into a new project for the following and last six months of my studies at ETH Zurich.
First, I'd like to thank my Supervisor Fabian Guenther, who, with his strong physical intuition and his vast building and designing experience, greatly helped me to push forward, rethink, and realize my ideas.
I'd also like to thank Professor Fumiya Iida, who kindly accepted me as a member of his lab and gave me the opportunity and support to work on a highly interesting topic.
Furthermore, I'd like to thank all the other people who are with the bio-inspired robotics lab, for many interesting discussions and inputs. It was a pleasure to work in a lab with such a great spirit and atmosphere.

Contents

Chapter 1

Introduction

In recent years, legged locomotion has been given more and more attention in the robotics community. Scientists have started to analyze animals such as kangaroos or cheetahs in order to rebuild a robot model, developed robots that can walk without a motor, or built legged robots which can maintain their balance even if they are pushed severely. The question arises why many institutions are interested in building legged robots. Consdiering that wheeled robots or vehicles, such as cars or trains, are almost unexcelled in terms of locomotion efficiency, this is certainly an important question to discuss. As a matter of fact, human walking is about a factor of 10 times less efficient than wheeled vehicles (Tucker 1970). To make things worse, nowadays most advanced and versatile robots are in the range of 10 times less efficient than human walking (Iida 2012). The reason for the increasing interest in legged locomotion doesn't lie within its efficiency, but within its high flexibility and agility. This can be easily comprehended if one considers the drawbacks of wheeled locomotion in rough terrain, such as forests or mountains. A legged robot on the other hand, has the ability to adapt its mechanical structure to the given environmental conditions, and is hence the preferred choice for explorational ventures.

In the remaining parts of this chapter, an overview of legged animal- and robot locomotion research will be given.

1.1 Human and Legged Animal Locomotion

In terrestrial locomotion, the cost of transport is a crucial factor for the comparison of animals' efficiency. It was used in (Tucker 1970) and relates the energy an animal has to expend to move a certain distance, to the body weight and the travelled distance. The cost of transport was found to vary with different body weight and type of locomotion, e.g. flying, running or swimming. Humans for example show two characteristic gaits which have to be considered separately, namely running and walking. A highly interesting fact with human walking, is its self-stability property. It can be shown with simple models (Garcia 1998), that it is possible to walk down a shallow slope without the use of any kind of control or actuation. This indicates, that humans' walking movement is defined by its intrinsic mechanical properties rather than by controlled signals of the

Figure 1.1: Illustration of different animal gaits. From left to right: Kangaroo hopping[1], ostrich running[2], cheetah with rotary gallop[3].

brain. When walking at increased speed, a transition from the walking gait to the running gait occurs. The speed at which the change happens is at about $\sqrt{gl}$ (Ruina 2005). The reason why changes are occurring are not yet well understood (Raynor 2002). (Diedrich 1995) suggests, that the transition occurs mainly when stability of the gait is lost due to velocity change and therefore, energy expenditure rises.
There are various properties of the leg which influence the energetic cost of locomotion. In (Adamczyk 2006), the influence of the foot radius for human walking was tested with adult subjects. It was found that the radius of the foot significantly influenced the net metabolic rate of walking, where the optimal radius turned out to be 0.3 times the leg length. The reason why there are better and worse ways to walk, can be explained by energy losses due to impacts at touchdown. These are depending on foot and leg orientation, velocity and the material properties (Gerritsen 1995). The impact is followed by the ground contact phase, where the muscles are able to produce an accelerating force on the body. It can be shown, that the ground contact time is another dominant factor for the overall metabolic energy use of humans, as well as quadrupeds (Roberts 1998).

Other than the gaits of bipedal walkers such as humans' running or kangaroos' hopping, quadrupeds show some advanced ways of locomotion. One of the fastest gaits is the gallop, which is used by the horse, as well as the cheetah, the fastest terrestrial animal (Bertram 2009). Similar analysis as for human walking has been done for the quadrupedal gaits in recent years. A simple model (Berkemeier 1998) suggests that the body inertia is pivotal for stability and transition of quadrupedal gaits, such as bounding or pronking. As this type of locomotion is not further investigaitd in this thesis, the reader is referred to the literature, e.g. in (Kar 2003).

A crucial factor of any legged locomotion is the leg's compliant property. If one had rigid legs, all of the kinetic energy at impact in normal direction to the ground would be absorbed, which would decrease the efficiency of locomotion severely. In order to avoid unnecessary energy losses at impact, muscles and tendons are made to act as springs, which can recover energy from the sprung mass at impact. (Alexander 1990) found three different usages of springs in

[1]Picture Source: www.empowernetwork.com, 9.28.2013.
[2]Picture Source: www.featheremporium.com , 9.28.2013.
[3]Picture Source: www.nationalgeographic.com, 9.28.2013.

legged biological systems. First, the named energy retrenchment is realized by harnessing the "pogo stick principle". Second, the swing motion of the leg can be redirected when reaching either end of the rotational movement of the hip. Finally, direct impact losses of the unsprung mass can be prevented by compliant foot pads, which also prevent the foot from chattering. In (Alexander 1995), different jumping techniques, which have the advantage of exploiting the compliant behavior of the leg, are compared. Locusts and fleas are able to preload their tendons and muscles, such that they can apply an explosive force in a short period of time, which results in a high jump. Vertebrates on the other hand, can not make use of a pre-loading system. They use countermovement jumps in order to exploit the benefits of their springy legs. Squat jumps were shown to produce the smallest jumping height, as no elastic energy is stored with this jumping technique.

1.2 Legged Robots

During the last couple of years, two distinct categories have been emerging from the field of legged robots. On the one hand, we have the feedback controlled legged robots, which use sensors to identify the system's state and correct deviations from the desired trajectories by an implemented controller. On the other hand, feedforward controlled legged robots have been given more and more attention. These robots are using no sensory feedback at all, and are thus simpler in terms of control. Passive dynamic walkers even manage to walk down a slope without any kind of actuation. The reason why those robots manage to move stably without feedback lies within their intrinsic mechanical nature. Hence, a careful design of the mechanical structure of the system can greatly simplify the control effort. The two types of robots are presented in more detail in the following sections. Good summaries of the existing robots and their efficiency can be found in (Kuo 2007) and (Sayyad 2007).

1.2.1 Feedback Controlled Robots

One of the most influencing robots, which triggered a rising interest in legged locomotion was presented in (Raibert 1984). The 3D hopping machine first demonstrated the enormous potential of feedback controlled legged robots. Without using any support and hopping on one leg only, the robot was able to jump stably in all spacial direction without falling down. After Raibert, (Gregorio 1997) and (Ahmadi 2006) showed with their ARL-Monopods I and II, that energy efficient and fast one-legged and planar locomotion can be realized by applying Raibert's control strategy. A simpler method to hop was presented in (Zeglin 1999), where a hopper with a bowed leg, that can be pre-loaded by a string, showed high energy efficiency. On the other hand, very complex one-legged robots are investigaitd, such as a robot leg of the cheetah's hind limb (Lewis 2011).

Not only one-legged robots have been constructed in the past. Bipedal robots are often investigaitd, as they can imitate humans' gaits, such as walking or running. One of the most famous humanoid robots is ASIMO (Sakagami 2002), which is capable of moving like a human and interact with subjects around him.

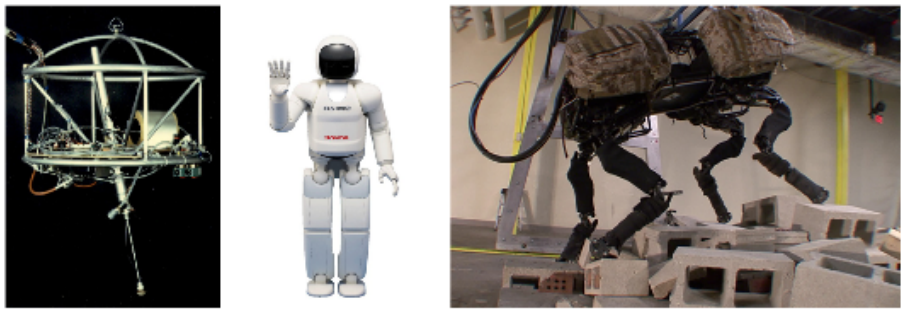


Figure 1.2: Illustration of various feedback controlled robots. From left to right: Raibert's 3D hopper[1] , ASIMO[2], BigDog[3].

The PETMAN robot (Nelson 2012) was constructed to test chemical protective clothing, but has astonishing capabilities as for human like walking and gesturing. Bipedal robots are even being constructed to learn by themselves how to walk, such as the 3D biped in (Tedrake 2002) and (Tedrake 2004), where the robot learns to to adapt to changing ground properties as it is walking.

There is a broad interest in simulating quadrupeds and implementing control strategies for animation ((Coros 2011), (Kokkevis(1995)), but also real world applications harness the advantages of quadrupedal systems. Due to the favorable stability and performance properties, many researchers have been investigating quadrupedal robots, which has made them the most complex legged locomotion machines to date. (Raibert 1990) studied gaits of quadrupedal animals, such as the trot, the pace and the bound. Based on this work, one of the most astounding robots in terms of stability and flexibility was developed, Big Dog (Raibert 2008). This robot is able to surmount highly complex terrain and walk up and down steep slopes. Many other similar quadrupedal robots have been developed during recent years, e.g. StarlETH (Hutter 2012), which is a platform to study fast, efficient and versatile locomotion, or HyQ (Semini 2010), a hydraulically actuated quadruped robot.

1.2.2 Feedforward Controlled Robots

Without any sensory feedback, one is not able to determine what the robot is doing at any time. If we were to construct a hopping robot for example, we would have to make sure that the robot doesn't fall down. This can be performed by designing the robot in a way, such that its mechanical structure is able to converge to a desired system state. If the robot's mechanical structure is preventing it from becoming unstable, it is called self-stable.

One branch of self-stable robots are passive dynamic walkers. The term was introduced in (McGeer 1990) and comprises robots, which are driven by gravitational force. McGeer tested a bipedal robot, which was capable of walking

[1]Picture Source: www.bostondynamics.com, 9.28.2013.
[2]Picture Source: www.asimo.honda.com , 9.28.2013.
[3]Picture Source: www.ai.mit.edu , 9.28.2013.

Figure 1.3: Illustration of various feedforward controlled robots. From left to right: Passive dynamic walker[1] , RHex[2], Curved beam hopper[3].

down a slope with a constant walking velocity and without falling over. No actuator was driving the robot, but only gravity exerted a force. Since McGeer's first study on passive dynamics, many have followed his concept and built more efficient and faster legged walkers (Collins 2005). In (Owaki 2010) and (Owaki 2011), the limits for passive dynamics were pushed further away, as they managed to build the first passive dynamic runner, being able to run stably for 36 steps. By adding feedback control to the hip position only, passive dynamic running can get as fast as $35.4\,\mathrm{km\,h^{-1}}$, as shown in (Cotton 2012). In this publication, the authors present an ostrich-like running robot, based on passive dynamics.
The one drawback of passive dynamic walking and running is its high sensitivity towards disturbances. A small bump on the ground is enough to push the robot to its unstable region. A thorough analysis of stability for running robots has been conducted in (Ringrose 1997). By combining a curved foot with a compliant leg, a simple monopod was built, showing self-stabilizing properties as for hopping height. However, the robot was just hopping in place and no forward movement was achieved.

Moving away from passive dynamic running and walking, more robust and versatile feedforward controlled robots can be found. One example of a simple but highly robust and agile robot is RHex (Saranli 2001). This hexapod robot gives a feedforward signal which drives six legs, each being moved by an individual actuator. The legs are turning full cycle and a swing phase of the legs can therefore be avoided.
The last type of open loop robots that will be discussed, is the one most related to this thesis. The curved beam hoppers are exploiting the natural frequency of the robot's structure in order to hop (Reis 2011) or run (Maheshwari 2012). They are built out of a bent elastic beam, where a DC motor is put on top. The motor is rotating an eccentric mass, which, if excited at the right frequency, makes the robot hop.
The investigations in this thesis will follow up on the research on the curved beam hoppers. Feedforward control and its underlying principles will hence be in the spotlight for the remaining parts of this report.

[1]Picture Source: www.ruina.tam.cornell.edu, 9.28.2013.
[2]Picture Source: www.bostondynamics.com , 9.28.2013.
[3]Picture Source: www.birl.ethz.ch , 9.28.2013.

Chapter 2

Theoretical Background

As mentioned in the last chapter, this thesis will focus on the development of a feedforward controlled robot. This field is broad and requires knowledge of advanced theory in mechanical systems, such as self-stability, which will be explained in section 2.4. In addition, the complexity of the dynamics of legged locomotion and its physical interactions with the ground is very high. Since the equations of motion change their internal structure as the state goes along its predefined trajectory, it is hard to find analytical solutions for the problem at hand. The change in the mathematical structure is caused by the alternation from ground phase to flight phase and vice versa. What makes the problem even more delicate are the impulsive ground reaction forces, which tend to go to infinity for a vanishing time interval at impact. Therefore, a thorough mathematical layout for this problem is the key to success.
In this chapter, simplified models and various aspects of legged locomotion are presented, as well as the mathematical principles needed for the following description of the model used in this thesis (see Chapter 3).

2.1 Spring Loaded Inverted Pendulum Model

One of the simplest model one can possibly use to describe some sort of gait is a spring-mass system hopping in place (Blickhan 1989). If we had a point mass of mass m, a spring attached to the point mass with stiffness k and the only external force is gravity g, the resulting equations of motion in the case that the spring is in contact with the ground looks as follows:

$$m\ddot{y} + ky = mg, \tag{1}$$

where y is the distance of the point mass to the ground. Once the spring reaches its relaxed length while accelerating away from the ground, it detaches and the equations of motion change their internal structure:

$$m\ddot{y} = mg. \tag{2}$$

As can be seen, the solutions of equation (1) and (2) are depending on each other. After the ground phase cycle, the initial conditions need to be passed on to the solution of the flight phase equation of motion and vice versa.

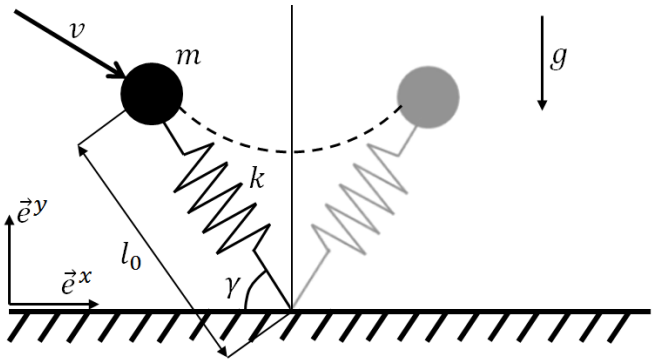


Figure 2.1: Sketch of the spring loaded inverted pendulum (SLIP) model. The landing angle γ is predefined and constant for each hop.

The above mentioned system is only valid for vertical hopping in place. If one is interested in locomotion, forward speed is a crucial factor for a model. By adding another dimension to the problem, we end up with the spring loaded inverted pendulum model (SLIP). The equations of motion become non-linear in the ground phase, as the spring force is no longer acting linearly on the spacial states:

$$\ddot{x} = x\,\omega^2 \left(\frac{l_0}{\sqrt{x^2 + y^2}} - 1 \right) \tag{3}$$

$$\ddot{y} = y\,\omega^2 \left(\frac{l_0}{\sqrt{x^2 + y^2}} - 1 \right) - g. \tag{4}$$

Note that x is the horizontal state of the point mass, $\omega = \sqrt{k/m}$ is the natural frequency of the spring-mass system and l_0 is the relaxed spring length. During flight phase, the equations of motion for the y-direction stay the same as in the simplified hopping model (2). For the x-direction, there is no change in velocity, as no forces are acting in this direction. The switching condition from flight phase to ground phase is given by a predefined landing angle γ, which is set as a parameter in the beginning of the simulation. If γ hits the condition

$$y \leq l_0 \sin\gamma, \tag{5}$$

it is switched from flight- to ground phase. On the other hand, if the relaxed spring length is reached during ground phase, i.e.

$$l_0 < \sqrt{x^2 + y^2}, \tag{6}$$

it is switched back from ground- to flight phase.

The same equations have been derived for a non-linear spring behavior in (Rummel 2008). In this publication, a segmented leg was chosen with a rotational spring stiffness between the thigh and the shank instead of a linear, straight spring. Due to this manipulation of the model, lower speeds for self-stable running was achieved and the tolerated range of the landing angle was increased.

Further literature on the SLIP model can be found in (Holmes 2006) and (Blickhan 2007).

The advantages and limiting assumptions of the SLIP model have to be considered before using it for real world predictions. On the one hand, the model provides a simple prediction of locomotion, which is capable of explaining many phenomenons in animal- and human locomotion. On the other hand, the predefined angle γ can only be guaranteed, if the animal or robot has a hip and is able to control the landing angle by swinging the leg back to the desired position. The terms "control" and "desired position" already imply feedback control. In our case, the robot won't have the opportunity to set the angle γ to a desired state, as we won't have any sensory feedback in our system. Hence, we will need a model which does not require the value of γ at touchdown to be a priori known. A more sophisticated model which satisfies our needs for self-stability analysis will be presented in chapter 3.

2.2 Influence of Impacts on Efficiency

In this section, the influence of the impact on the cost of transport is analyzed. The cost of transport, an important dimensionless measure in terms of locomotion efficiency in legged locomotion, is defined as

$$CoT = \frac{E_{exp}}{m\, g\, x_{end}}, \tag{7}$$

Where E_{exp} is the expended energy, m is the mass of the system, g is the gravitational acceleration, and x_{end} is the moved distance. If we assume now, that at each hop, some energy is lost due to dissipation of kinetic energy at impact, we can estimate the energy loss at each stride. The kinetic energy of an arbitrary rigid body is defined as

$$E_{kin} = \frac{1}{2}\, m\, \boldsymbol{v}^T\, \boldsymbol{v} + \frac{1}{2}\, \boldsymbol{\Omega}^T\, \boldsymbol{\Theta\Omega}, \tag{8}$$

or for the planar case

$$E_{kin} = \frac{1}{2}\, m\, v_x^2 + \frac{1}{2}\, m\, v_y^2 + \frac{1}{2}\, \Theta\Omega^2, \tag{9}$$

where $\boldsymbol{v}$ is the velocity of the rigid body at the center of mass, m is the mass of the body, $\boldsymbol{\Theta}$ is the moment of inertia and $\boldsymbol{\Omega}$ the angular velocity of the body. In order to proceed, one needs to simplify this expression, as the dependence of the tangential force on the normal force makes further simplifications hard to accomplish. The lost energy is dependent on the normal force, the dynamic friction coefficient and the restitution factors in normal- and tangential direction. One simple assumption we can make, is that only energy is lost in the vertical direction to the ground, i.e. hopping motion in place. This would also correspond to hopping with no friction on the ground, or a vanishing relative speed at the touchdown point. This is the case when legged animals accelerate their foot tip velocity to the relative ground speed during stance phase. Assuming a Newtonian kinematic impact law with restitution factor ε_N in normal direction to the contact point, the subsequent energy dissipation per touch down can be

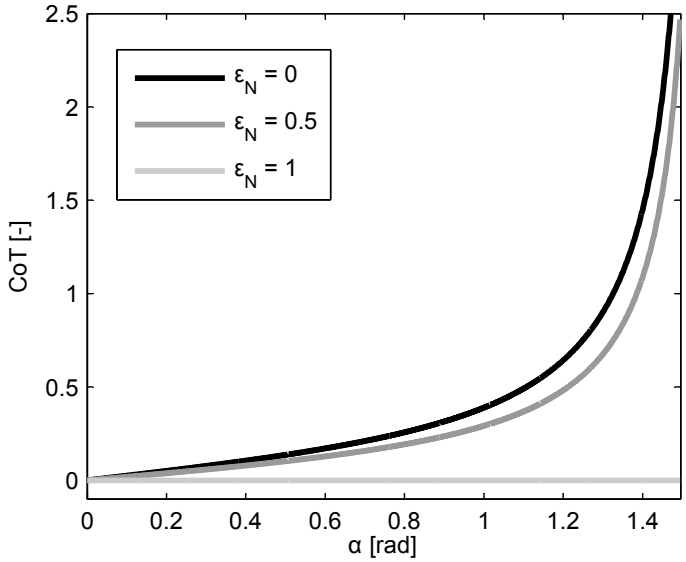


Figure 2.2: Theoretical lower bounds (eq. (13)) of the cost of transport as a function of the jumping angle of the center of mass α. Different normal restitution factors ε_N are plotted.

computed. Note that a detailed description of impact and frictional models is described in section 3.5.

$$E_{exp} = E_{kin}^{-} - E_{kin}^{+} = \frac{1}{2}\, m\, (v_y^{-})^2 \left(1 - \varepsilon_N^2\right), \tag{10}$$

where a + sign indicates a quantity after impact, and - before impact, respectively. If we now form the cost of transport into

$$CoT = \frac{E_{exp}/\Delta t}{m\, g\, x_{end}/\Delta t} = \frac{P_{exp}}{m\, g\, v_x}, \tag{11}$$

and calculate the flight phase time of a body under influence of gravitational acceleration g and with an initial velocity $\boldsymbol{v}$ by

$$\Delta t = \frac{2\|\boldsymbol{v}\| \sin\alpha}{g}, \tag{12}$$

where α is the jumping angle of the center of mass at take off, the cost of transport turns out to be

$$CoT = \frac{\left(1 - \varepsilon_N^2\right)}{4} \tan\alpha. \tag{13}$$

The shape of this function can be found in figure 2.2. Note that it only provides a lower bound for the cost of transport for a real system, as there are more losses than only for vertical hopping, e.g. frictional losses in horizontal direction or internal damping losses in the joints.

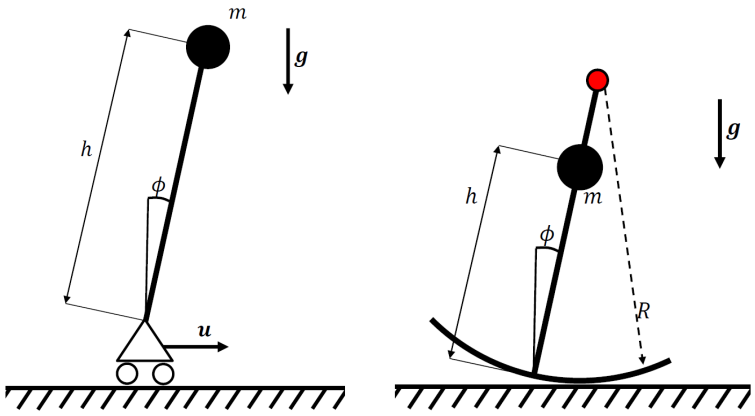


Figure 2.3: (**a**) Sketch of the inverted pendulum model. the control input $\boldsymbol{u}$ is stabilizing the system under the influence of gravity. Sketch (**b**) shows the self-stable version of the inverted pendulum. The mechanical structure guarantees a stabilization around the system's equilibrium point.

2.3 Self-Stable Inverted Pendulum

A popular example for the possibilities of feedback control is the inverted pendulum. It consists basically of a rod, which is being balanced on a moving platform such that it doesn't fall over. A simple mechanical solution to provide self-stability for this system is to fix the rod on a platform, or to add a curved structure at its bottom. The latter one should be analyzed in this chapter. In figure 2.3, the set up of the structure is depicted. The rod is assumed to have its center of mass m at its top end with distance to the ground h, and the radius of the curved structure should be denoted with R. If we were to deflect to rod from its equilibrium position ϕ_0, there is a resetting force, which pulls the rod back to its equilibrium position. The equations of motion of this system are

$$\theta\,\ddot{\phi} = -m\,g\,(R-h)\,\sin\phi \tag{14}$$

or for small deflections from the equilibrium position

$$\theta\,\ddot{\phi} = -m\,g\,(R-h)\,\phi \tag{15}$$

Starting from the linearized system (15), the solution obviously becomes unstable if h is larger than R, as the resetting force vector pulls the rod away from its equilibrium position. If the radius is equal to the distance of the center of mass to the ground, then for all initial conditions ϕ_0, we have an equilibrium position. This case would correspond to the behavior of a wheel.
In addition, we can calculate the frequency, with which the rod is swinging around its equilibrium position if it is deflected and no dissipation of energy occurs. After forming the second order differential equation to a first order system of differential equations and evaluating the determinant of the state matrix,

we'll get for the rocking frequencies of the linearized rod system to be

$$\omega_r = \sqrt{\frac{m\,g\,(R-h)}{\theta}}.$$

The results of this section can be used to explain static stability and swing time during stance phase of a curved foot robot.

2.4 Stability of a Mechanical System

The last section described a concrete example of a self-stable system, namely the self-stable inverted pendulum. By showing its unstable counterpart, the inverted pendulum, the definition of self-stablity becomes more intuitive. A system is called self-stable if it is able to approach its equilibrium point without the help of feedback control.

A mathematically more rigorous way of looking at stability can be done by using the definition of Lyapunov, which says that if the initial condition $\boldsymbol{x}_0$ of our system start in the neighborhood δ of the equilibrium point $\boldsymbol{x}^*$, there exists a bound ϵ which encloses the solution for all future times t in the state space $\boldsymbol{x}$. Or in other words

$$\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| < \delta \quad \Rightarrow \quad \|\boldsymbol{\psi}\,(t, t_0, \boldsymbol{x}_0) - \boldsymbol{x}^*\| < \epsilon \quad \forall t > t_0, \tag{16}$$

where ψ is the solution of our system's equation and t_0 is the initial time. Another important term is attractivity of an equilibrium. An equilibrium x^* is locally attractive if there exists a $\delta > 0$ such that

$$\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| < \delta \quad \Rightarrow \quad \lim_{t\to\infty} \|\boldsymbol{\psi}\,(t, t_0, \boldsymbol{x}_0) - \boldsymbol{x}^*\| = 0, \tag{17}$$

so if the solution ψ of our system approaches an equilibrium point as time goes to infinity. If an equilibrium point is stable and locally attractive, it is called locally asymptotically stable.

In this thesis, starting from the definition of Lyapunov, the jumping angle of the center of mass at take-off α was found to be a good indicator of the system's state. If the jumping angle crosses a limit angle δ, the robot will fall and hence, the system is unstable. Furthermore, if the jumping angle is converging to an equilibrium state, the standard deviation σ of the average jumping angle will be low. This means that σ can be used for an estimation of the system's local attractivity. Note that even though its mathematically not always correct to call an attractive equilibrium point stable, small σ-valued simulations will be called stable, as they won't include simulations where the robot fell.

2.5 Mathematical Principles

2.5.1 Set-Valued Functions

A set-valued function can have one or more elements of the function's domain, whose image is not one single element, but a subset of the codomain. Or in other words, given a function

$$f(x) = y, \tag{18}$$

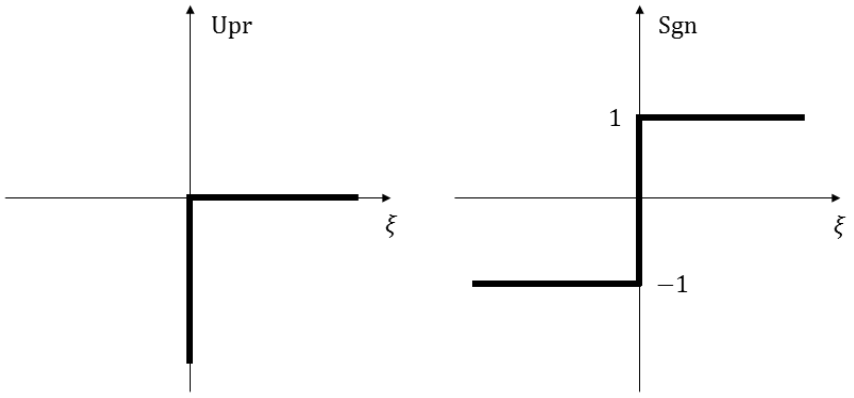


Figure 2.4: Graph of the set-valued Upr- and Sgn functions. They will be needed in section 3.5 to define ground contact forces.

where $y \in Y$ and $x \in X$, a subset of Y might be assigned to one and the same x.
From this definition, two important set-valued functions can be described:

$$\mathrm{Upr}(\xi) = \begin{cases} (-\infty, 0] & \text{if } \xi = 0 \\ 0 & \text{if } \xi > 0 \end{cases}, \tag{19}$$

and

$$\mathrm{Sgn}(\xi) = \begin{cases} -1 & \text{if } \xi < 0 \\ [-1, 1] & \text{if } \xi = 0 \\ 1 & \text{if } \xi > 0 \end{cases}. \tag{20}$$

Upr is called a unilateral primitive, and Sgn is the set-valued signum function, as described in (Glocker 2005). Those two functions will be used later on to describe impacts and frictional forces mathematically.

2.5.2 The Linear Complementarity Problem

In optimization theory, the linear complementarity problem is an inequality-constrained equation, which will be used in our case to relate the kinematic states of our system during ground contact to the acting forces. It is defined as

$$\mathbf{y} = \mathbf{A}\,\mathbf{x} + \mathbf{b} \tag{21}$$

$$\mathbf{y} \geq 0, \qquad \mathbf{x} \geq 0, \qquad \mathbf{y}^T\mathbf{x} = 0, \tag{22}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b}$, $\mathbf{y}$, $\mathbf{x} \in \mathbb{R}^{n\times 1}$. If we define $\mathbf{E} := (\mathbf{e_1}, ..., \mathbf{e_n})$ the identity matrix and $\mathbf{A} := (\mathbf{a_1}, ..., \mathbf{a_n})$, equations (10) and (11) can be rewritten as

$$(\mathbf{e_1}, ..., \mathbf{e_n}, -\mathbf{a_1}, ..., -\mathbf{a_n}) \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \end{pmatrix} = \mathbf{b}, \tag{23}$$

$$y_i \leq 0, \qquad x_i \leq 0, \qquad y_i x_i = 0 \quad \forall i = (1, ..., n)\,. \tag{24}$$

Now be $\mathbf{c_i} \in \{\mathbf{e_i}, -\mathbf{a_i}\}$ the i-th complementary pair of vectors, and $z_i \in \{\mathbf{y_i}, \mathbf{x_i}\}$ the i-th complementary pair of variables such that

$$z_i = \begin{cases} y_i & \text{if } \mathbf{c_i} = \mathbf{e_i} \\ x_i & \text{if } \mathbf{c_i} = -\mathbf{a_i} \end{cases}. \tag{25}$$

Now, given that $\mathbf{c_i}$ can either be equal to $\mathbf{e_i}$ or $-\mathbf{a_i}$, there are many different combinations how we can build a matrix $\mathbf{C_k} := (\mathbf{c_1}, ..., \mathbf{c_n})$ and a corresponding vector $\mathbf{Z_k} := (z_1, ..., z_n)^T$. In fact, 2^n different combinations are possible and will eventually limit the performance of the numerical procedure to solve the linear complementarity problem. The derived results will be used in chapter 4.

Chapter 3

Physical Model

After having covered the relevant theoretical principles in the last chapter, the physical set-up of the robot can be tackled. In this chapter, after stating the given knowledge and findings of previous work at the bio-inspired robotics lab and extracting the needed objective for this project, the new physical model of the robot to build is presented. Subsequently, system parameters, equations of motions as well as ground contact interactions are defined.

3.1 Previous Work and Objective

During the last couple of years, curved beam hoppers have been found to satisfy both, easy construction properties and high locomotion efficiency. Having said that, the non-linear elastic behavior of the curved beam is an obstruction for further numerical analysis and development. In addition, the actuation input has to be chosen such that the natural frequency of the system is excited. In (Mathis 2013), a hopper based on the curved beam is realized with rigid elements and variable motor input, as can be seen in figure 3.1. The resulting segmented robot gets close to the efficiency and speed of the curved beam hopper. Even though the robot was designed using rigid elements, several parts were still adding some unpredictable non-linearity to the system, such as the bent aluminium coupling between the body and the foot. Due to the given problems, the objective of this thesis cleared up: A hopping robot made out of rigid elements should be designed, whose mechanical features allow a simple mathematical and numerical analysis. Out of the simulation, real world behavior should be predicted accurately enough to show matching results to an experiment. General conclusions for legged locomotion is an additional goal to establish.

3.2 Approach

Biological systems come in handy if one is to optimize a desired task in robotics, which has been already realized in nature. As legged locomotion is one of those, it makes sense to study animal gaits and check if the prevailing laws can be discovered and adapted to a robotic system. Since the hopping gait should be

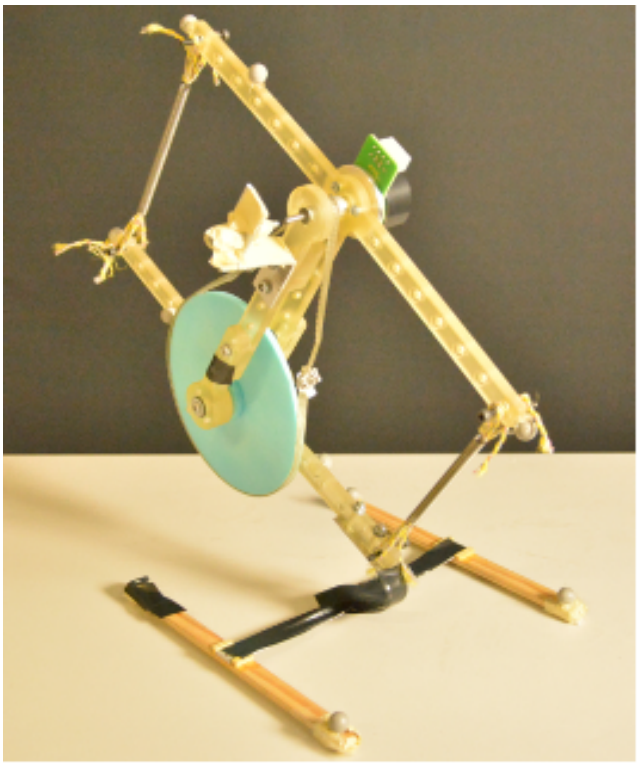


Figure 3.1: Illustration of the segmented beam hopper. Its model consists of 3 rigid bodies, two linear springs, one torsional spring, and is driven by a sinusoidal input torque.

realized, mainly two legged vertebrates are of interest, such as kangaroos, humans or birds. By studying the mechanics of hopping kangaroos (Alexander 1975), it became clear that a hip is crucial for defining the landing angle while hopping. Without a hip, stable standing is hard to realized except if a stabilizing structure like a wide foot is added, which will cause high impact losses at touchdown. Having said that, by using light and stiff materials, the impact losses are moderate and can be tolerated. In previous work, a large foot was mounted to the robot's body by a torsional spring. However, by using a spring, the number of rigid bodies increases by one and renders the system analysis more difficult. The advantage of the torsional spring was that first, it can help to reduce impact losses by storing elastic energy and second, it guarantees a necessary flexibility as for the varying landing- and jumping angle.
As this thesis aims at reducing the system's complexity, an altered approach for the design of the foot was sought. By choosing light materials for the shank of the robot, direct impact losses become less important. The flexibility of the jumping- and landing angle can be achieved without using a torsional spring: If one uses a well designed curved foot, which is firmly attached to the shank, a pitching motion of the robot during stance phase can be realized without the usage of a spring and without expenditure of additional energy. For the aforementioned reasons and for the sake of reduction of complexity, it was decided

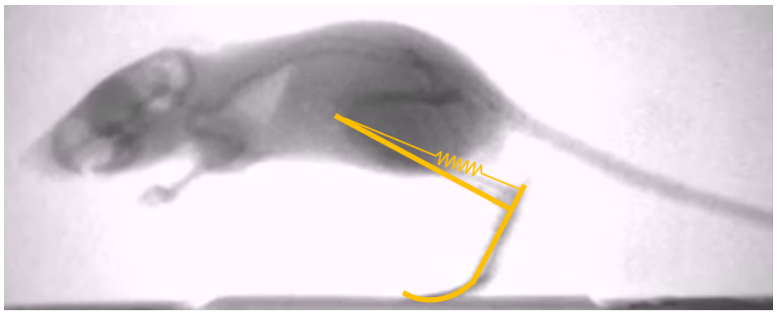


Figure 3.2: X-ray of a jumping kangaroo rat[1]. The orange drawing indicates a possible model for the lower part of the hind leg.

to drop the torsional spring and examine the aptitude of a curved foot.

The remaining parts of the robot were defined easily. It should consist of a thigh, where the main mass is added, and a light shank, which is rigidly attached to a curved foot. A muscle-tendon like spring should be providing elastic suspension of the main mass on top of the thigh. Note that the lower leg needs to have a certain mass and inertia, since the pitching motion of the upper leg during flight phase is depending on it, as was found by running various simulations. This renders a challenging task of finding an optimal lower leg weight for low impact losses and desired pitching motion during flight phase.

The force driving the robot is applied as a torque at the joint between the lower leg and the upper leg. The trajectory of the motor torque was chosen to be sinusoidal, which leads to the following function:

$$T_M(t) = A_T \sin(\omega_T t), \tag{26}$$

where A_T is the amplitude of the motor torque, and ω_T is its angular frequency.

3.3 System Parameters

Before the analysis of the equations of motion, the fixed parameters of the system will be defined. Note that most of them have been set due to anatomical properties of animals such as the kangaroo rat or intuition. see table 3.1 for the list of parameters.
The spring stiffness k was chosen, such that the mass of the upper body could be carried under the influence of gravity. The damping coefficient d was initially chosen as in the previous project (Mathis 2013), but was later altered to match the real world experiment. the geometrical parameters l_l, l_u, and l_h were first chosen to have the same proportions as a kangaroo (Alexander 1975). Later they were iteratively adapted by running several simulations. The lower body mass m_l and m_u, as well as the moments of inertia θ_l and θ_u were first assigned by considering possible materials to use for the design and the known weight

[1]Picture Source: www.sci.uidaho.edu/McGowanLab, 9.29.2013.

Table 3.1: Parameters and Variables of the robot model. The column "Values" shows the used numbers for the simulation and the experiment.

Fixed Parameters		
Letter	**Name**	**Value**
k	Spring stiffness	$3022\,\mathrm{N\,m^{-1}}$
d	Damping coefficient	$0.06\,\mathrm{N\,m\,s}$
l_l	Lower body length	$0.2\,\mathrm{m}$
l_u	Upper body length	$0.16\,\mathrm{m}$
l_h	Moment arm spring	$0.05\,\mathrm{m}$
m_l	Lower body mass	$0.239\,\mathrm{kg}$
m_u	Upper body mass	$0.481\,\mathrm{kg}$
A_T	Motor torque amplitude	$0.5\,\mathrm{N\,m}$
s_f	Foot length	$0.2\,\mathrm{m}$
x_{s1}	Horizontal coordinate of lower leg center of mass	$-0.063\,\mathrm{m}$
y_{s1}	Vertical coordinate of lower leg center of mass	$0.045\,\mathrm{m}$
θ_l	Lower body inertia	$0.0015\,\mathrm{kg\,m^2}$
θ_u	Upper body inertia	$0.0016\,\mathrm{kg\,m^2}$
φ_0	Initial Knee angle for relaxed spring	$0.7853\,\mathrm{rad}$
Tunable Parameters		
f_T	Motor torque angular frequency	$2\,\mathrm{Hz}$ - $6\,\mathrm{Hz}$
R	Foot Radius	$0.2\,\mathrm{m}$ - $0.6\,\mathrm{m}$
β	Lower leg angle	$0.4\,\mathrm{rad}$ - $1.2\,\mathrm{rad}$
Minimal coordinates		
$x(t)$	Position x-value of lower leg COM	m
$y(t)$	Position y-value of lower leg COM	m
$\varphi_l(t)$	Lower leg angle	rad
$\varphi_u(t)$	Upper leg angle	rad
Computed Variables		
$T_M(t)$	Motor moment function	$\mathrm{N\,m}$
$\alpha(t)$	Take off angle of lower leg	rad
$\varphi(t)$	Knee angle	rad
$\omega_T(t)$	Motor torque angular velocity	$\mathrm{rad\,s^{-1}}$

of the motor. The foot length s_f was chosen intuitively. The motor torque amplitude A_T was first varied within a range of 0 to 1 Nm and was then later found to show clear hopping heights for a value of 0.5 Nm. The coordinates of the lower leg center of mass were needed, as the geometry of the robot is referenced by those values. Finally, the initial knee angle φ_0 for the relaxed spring lenght concludes the list of fixed parameters.
All the remaining parameters were being changed for parameter studies during the simulations, as will be described in section 7.2.

3.4 Equations of Motion

Having defined the parameters of the system, one can start to derive the equations of motion. One of the simplest ways in doing so, might be by exploiting

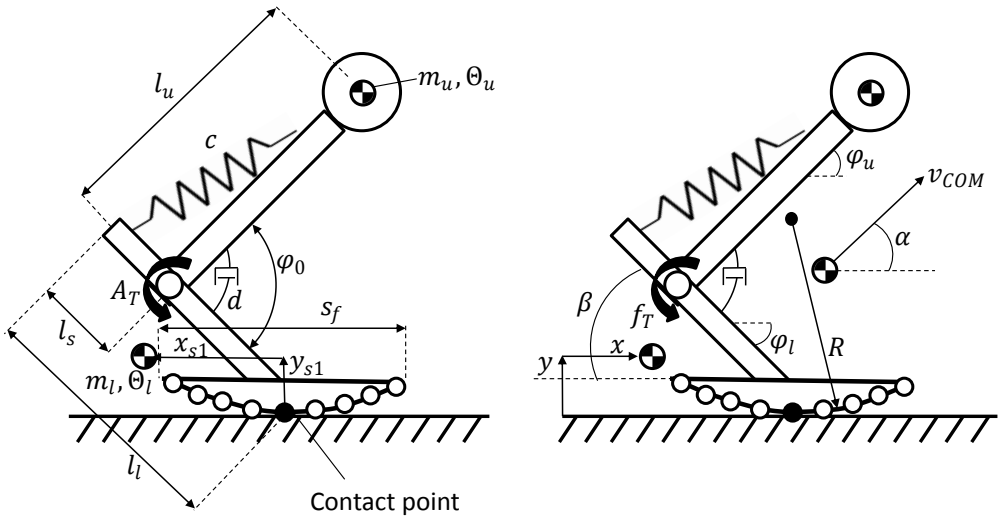


Figure 3.3: Sketch of the the curved foot hopper model including the constant system parameters in the left picture, and the changeable parameters and system states in the right picture.

the simple structure of the Newton-Euler equations:

$$0 = \sum_{i=0}^{n} \left[\left(\mathbf{J}_S^T \, \dot{\mathbf{p}} + \mathbf{J}_R^T \, \dot{\mathbf{N}}_S \right) - \left(\mathbf{J}_Q^T \, \mathbf{F}^A + \mathbf{J}_R^T \, \mathbf{M}_Q \right) \right], \tag{27}$$

where $\mathbf{J}_S$ is the Jacobian of the center of mass of the ith rigid body, $\mathbf{p}$ is the momentum of the same body, $\mathbf{J}_R$ is its rotational Jacobian and $\dot{\mathbf{N}}_S$ its spin. The external forces acting on the body are $\mathbf{F}^A$ with its Jacobian $\mathbf{J}_Q$ and the moments $\mathbf{M}_Q$. Note that n is the number of rigid bodies in the system, and the kinematics of the system are described in minimal coordinates $\mathbf{q}$. The model from which the equations were derived is shown in figure 3.3. The resulting full equations of motion can be found in the numerical code of appendix C.2.

3.5 Ground Contact Interactions

The most tricky part in reaching a meaningful mathematical model is the interaction with the ground. As this provides an instant change in the governing equations of motion and in addition, leads to impulsive forces acting on the robot, a sophisticated approach is crucial in order to obtain reasonable results. (Glocker 2005) presented in his publication how two dimensional systems, which are under the influence of structural changes in the differential equations due to friction and impacts, can be treated in a mathematically elegant way. Note that in this thesis, the system to be examined is assumed to have scleronomic constraints, which simplifies some expressions in the following elaboration. For rheonomically constrained systems, slight extensions need to be included (see (Glocker 2001) and (Glocker 2005)).

Given the set of minimal coordinates

$$\mathbf{q} = \begin{pmatrix} x \\ y \\ \varphi_l \\ \varphi_u \end{pmatrix} \tag{28}$$

of our system, where x and y are the horizontal- and vertical position of the lower leg center of mass, respectively, φ_l is the lower leg absolute angle, and φ_u is the upper leg absolute angle, as can be seen in figure 3.3. The equations of motion have the form

$$\mathbf{M}\,\ddot{\mathbf{q}}(\mathbf{q}, t) - \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}, t) = 0. \tag{29}$$

By including impulsive forces in our system, we can find the following measure equality for a dynamical system

$$\mathbf{M}\,\mathrm{d}\mathbf{u} - \mathbf{h}\,\mathrm{d}t - \mathrm{d}\mathbf{R} = 0, \tag{30}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{u}$ is the velocity of the system in minimal coordinates, $\mathbf{h}$ are the gyroscopic accelerations as well as smooth, generalized forces and $\mathrm{d}\mathbf{R}$ is the measure of the contact forces. By defining force laws for $\mathrm{d}\mathbf{R}$, which depend on the system's kinematic states,

$$\mathrm{d}\mathbf{R} = \sum_{i \in \mathbb{H}} \mathbf{w}_N\,\mathrm{d}\Lambda_N + \mathbf{w}_T\,\mathrm{d}\Lambda_T, \tag{31}$$

where $\mathbb{H}$ is the set of closed contacts, $\mathbf{w}_N$ and $\mathbf{w}_T$ are the generalized normal- and tangential force directions, respectively, and $\mathrm{d}\Lambda_N$ and $\mathrm{d}\Lambda_T$ are the contact impulse measures in normal- and tangential direction, all the required dynamics of the system are defined. What is missing are the kinematic force laws, that express the acting impulses in terms of the kinematic states of the system. The velocity difference before and after impact are given by

$$\xi_N := \gamma_N^+ + \varepsilon_N\,\gamma_N^-, \qquad \xi_T := \gamma_T^+ + \varepsilon_T\,\gamma_T^-, \tag{32}$$

and

$$\gamma_N = \mathbf{w}_N^T\,\mathbf{u}, \qquad \gamma_T = \mathbf{w}_T^T\,\mathbf{u}, \tag{33}$$

where γ_N^+ and γ_N^- are the normal velocities to the ground after impact and before impact, respectively, and ε_N is the restitution coefficient in normal direction at impact. Note that the subscript T describes the same properties in tangential direction to the ground contact point.
With the derived functions, it is possible to define a Newtonian kinematic impact- as well as a coulomb frictional law for our system:

$$-\mathrm{d}\Lambda_N \in \mathrm{Upr}(\xi_N), \qquad -\mathrm{d}\Lambda_T \in \mu\,\mathrm{d}\Lambda_N\,\mathrm{Sgn}(\xi_T), \tag{34}$$

where μ is the dynamic friction coefficient of the system. Please recall the two set-valued functions Upr and Sgn from section 2.5.1. Equations (28) – (34) provide all relevant equations for the problem at hand. The only thing left is to reformulate the equations in a way, which will make them easy to solve numerically. The reader is referred to (Glocker 2005) for a detailed derivation of the following linear complementarity problem for dynamical, non-smooth systems:

$$\begin{pmatrix} \boldsymbol{\xi}_N \\ \boldsymbol{\xi}_R \\ \boldsymbol{\Lambda}_N \end{pmatrix} = \begin{pmatrix} \mathbf{W}_N^{\mathrm{T}}\,\mathbf{M}^{-1}\,(\mathbf{W}_N - \mathbf{W}_T\,\boldsymbol{\mu}) & \mathbf{W}_N^{\mathrm{T}}\,\mathbf{M}^{-1}\,\mathbf{W}_T & 0 \\ \mathbf{W}_T^{\mathrm{T}}\,\mathbf{M}^{-1}\,(\mathbf{W}_N - \mathbf{W}_T\,\boldsymbol{\mu}) & \mathbf{W}_T^{\mathrm{T}}\,\mathbf{M}^{-1}\,\mathbf{W}_T & \mathbf{I} \\ 2\,\boldsymbol{\mu} & -\mathbf{I} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\Lambda}_N \\ \boldsymbol{\Lambda}_R \\ \boldsymbol{\xi}_N \end{pmatrix} + \begin{pmatrix} \mathbf{W}_N^{\mathrm{T}}\,\mathbf{M}^{-1}\,\mathbf{h}\,\Delta t + (\mathbf{I} + \boldsymbol{\varepsilon}_N)\,\boldsymbol{\gamma}_N^A \\ \mathbf{W}_T^{\mathrm{T}}\,\mathbf{M}^{-1}\,\mathbf{h}\,\Delta t + (\mathbf{I} + \boldsymbol{\varepsilon}_T)\,\boldsymbol{\gamma}_T^A \\ 0 \end{pmatrix} \tag{35},$$

$$\begin{pmatrix} \boldsymbol{\xi}_N \\ \boldsymbol{\xi}_R \\ \boldsymbol{\Lambda}_N \end{pmatrix} \succeq 0 \qquad \begin{pmatrix} \boldsymbol{\Lambda}_N \\ \boldsymbol{\Lambda}_R \\ \boldsymbol{\xi}_N \end{pmatrix} \succeq 0 \qquad \begin{pmatrix} \boldsymbol{\xi}_N \\ \boldsymbol{\xi}_R \\ \boldsymbol{\Lambda}_N \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \boldsymbol{\Lambda}_N \\ \boldsymbol{\Lambda}_R \\ \boldsymbol{\xi}_N \end{pmatrix} = 0. \tag{36}$$

Equations (35) and (36) form the linear complementarity problem needed for the ground contact forces. Hereby,

$$\boldsymbol{\Lambda}_R := \boldsymbol{\mu}\,\boldsymbol{\Lambda}_N + \boldsymbol{\Lambda}_T, \tag{37}$$

$$\boldsymbol{\xi}_T := \boldsymbol{\xi}_R - \boldsymbol{\xi}_L, \tag{38}$$

$$\boldsymbol{\xi}_R := \mathbf{W}_N^{\mathrm{T}}\,\mathbf{u}^E + \boldsymbol{\varepsilon}_T\,\boldsymbol{\gamma}_T^A, \tag{39}$$

where the superscript A in $\boldsymbol{\gamma}_T^A$ refers to the initial point of the numerical step, and E as in $\mathbf{u}^E$ to the end point, as will be discussed in the next chapter.
Note that most of the entries are now vector- or matrix valued, depending on the number of closed contact points, i.e. if k contacts are closed in a system with f minimal coordinates:

$$\begin{aligned} \mathbf{W}_N &:= (\mathbf{w}_{Ni_1}, ..., \mathbf{w}_{Ni_k}), & \mathbf{W}_T &:= (\mathbf{w}_{Ti_1}, ..., \mathbf{w}_{Ti_k}) \in \mathbb{R}^{f\times k} \\ \boldsymbol{\Lambda}_N &:= (\Lambda_{Ni_1}, ..., \Lambda_{Ni_k})^{\mathrm{T}}, & \boldsymbol{\Lambda}_T &:= (\Lambda_{Ti_1}, ..., \Lambda_{Ti_k})^{\mathrm{T}} \in \mathbb{R}^k \\ \boldsymbol{\gamma}_N^A &:= (\gamma_{Ni_1}^A, ..., \gamma_{Ni_k}^A)^{\mathrm{T}}, & \boldsymbol{\gamma}_T^A &:= (\gamma_{Ti_1}^A, ..., \gamma_{Ti_k}^A)^{\mathrm{T}} \in \mathbb{R}^k \\ \boldsymbol{\xi}_N &:= (\xi_{Ni_1}, ..., \xi_{Ni_k})^{\mathrm{T}}, & \boldsymbol{\xi}_T &:= (\xi_{Ti_1}, ..., \xi_{Ti_k})^{\mathrm{T}} \in \mathbb{R}^k \\ \boldsymbol{\varepsilon}_N &:= \mathrm{diag}(\varepsilon_{Ni_1}, ..., \varepsilon_{Ni_k}), & \boldsymbol{\varepsilon}_T &:= \mathrm{diag}(\varepsilon_{Ti_1}, ..., \varepsilon_{Ti_k}) \in \mathbb{R}^{k\times k} \end{aligned}$$
$$\boldsymbol{\mu} := \mathrm{diag}(\mu_{i_1}, ..., \mu_{i_k}) \in \mathbb{R}^{k\times k}.$$

Chapter 4

Numerical Model

The last chapter described the physical set-up and models for the new robot. Due to the system's complexity, it quickly became clear, that the problem needs to be tackled by a numerical simulation rather than analytical analysis. In this chapter, the equations of motion (30) are reformulated in order to reach an expression which allows finding a numerical solution for the given problem. Furthermore, a procedure for finding a solution of the derived linear complementarity problem (35) and (36) will be presented.

4.1 Ground Contact Discretization

The crucial parameters for the ground interaction are the friction coefficient μ, and the normal- and tangential restitution factors ε, respectively. Those two parameters eventually define the forces acting from the ground on the robot during stance phase, namely friction- and impact forces. The forces are transmitted through the geometry of the interaction spot between the robot and the ground. Those spots are called contact points and need to be placed on discrete positions. In this simulation, n contact points were distributed equally on the arced foot and on each point touching the ground, the forces had to be calculated. Note that each contact point is assigned to a dynamic friction coefficient μ, a normal restitution factor ε_N, and a tangential restitution factor ε_T. Since it makes no sense to assign different values to the n contact points, the named parameters were kept constant. See table 4.1 for the assigned values of the contact point parameters.

4.2 Initial Conditions

For each simulation, the initial conditions were precalculated in order to start from the robots static equilibrium position. This approach makes sense, since the initial condition of the simulation will match with the experimental one. The way the calculation was performed can be seen in appendix B. Note that static equilibrium of the robot therefore becomes a necessary condition for the simulation to be stable. If a given parameter set does not allow a statically stable position of the robot, the simulation will classify this parameter set as unstable, even though there might be a dynamically stable solution of this configuration.

4.3 Time Stepping Algorithm

By defining a time stepping algorithm, it is possible to find the end states of an iteration within six steps, as described in this section. A flow diagram depicting the algorithm is shown in figure 4.1.

1. After choosing a time interval Δt, define the midpoint time $t^M := t^A + \frac{1}{2}\Delta t$, and end time $t^E := t^A + \Delta t$, where t^A is the initial time of the current iteration.

2. Define the midpoint displacement $\mathbf{q}^M := \mathbf{q}^A + \frac{1}{2}\Delta t\, \mathbf{u}^A \in \mathbb{R}^f$.

3. Calculate matrix properties at midpoint time

- **(a)** $\mathbf{M}(\mathbf{q}^M, t^M) \in \mathbb{R}^{f\times f}$, $\mathbf{h}(\mathbf{q}^M, \mathbf{u}^A, t^M) \in \mathbb{R}^f$
- **(b)** Out of $i = 1, ..., n$ contact points, set up the index set $\mathbb{H}$ of all k contacts $i_1, ..., i_k$ that are currently closed
- **(c)** Then, combining all $i \in \mathbb{H}$, calculate $\mathbf{W}_N(\mathbf{q}^M, t^M) \in \mathbb{R}^{f\times k}$, as well as $\mathbf{W}_T(\mathbf{q}^M, t^M) \in \mathbb{R}^{f\times k}$

4. Solve the linear complementarity problem (35) and (36) in order to get $\mathbf{\Lambda}_N$ and $\mathbf{\Lambda}_R$

5. Compute the end velocity of the iteration step u^E

$$\mathbf{u}^E = \mathbf{M}^{-1}(\mathbf{W}_N - \mathbf{W}_T\,\boldsymbol{\mu})\mathbf{\Lambda}_N + \mathbf{M}^{-1}\,\mathbf{W}_T\,\mathbf{\Lambda}_R + \mathbf{M}^{-1}\,\mathbf{h}\,\Delta t + \mathbf{u}^A \quad (40)$$

6. Compute the final state of the iteration $\mathbf{q}^E := \mathbf{q}^M + \frac{1}{2}\Delta t \cdot \mathbf{u}^E \in \mathbb{R}^f$

4.4 Index Allocation

As described in the previous section under step 3, the contacts that are currently closed need to be defined. First, a condition for a closed contact needs to be set. The property of interest is in this case the distance between the ground and the contact point of the robot, which will be denoted by g_N. In the numerical function (see appendix C.3), all the contact points are checked for their g_N value. If it is smaller than zero for the midpoint displacement, i.e. $g_N(\mathbf{q}^M) < 0$, the contact needs to be included in the set $\mathbb{H}$. Once all the closed contacts have been found, the matrices $\mathbf{W}_N(\mathbf{q}^M)$ and $\mathbf{W}_T(\mathbf{q}^M)$ can be formed. The size of the linear complementarity problem will depend only on the size of $\mathbb{H}$.

4.5 Solving the Linear Complementarity Problem

After the index-allocation, the linear complementarity problem can be formed with equations (35) and (36). In section 2.5.2, it was discussed how the linear complementarity problem can be rewritten. Recall that

$$\mathbf{C}_k\,\mathbf{Z}_k = \mathbf{b}. \quad (41)$$

Table 4.1: Parameters and Parameter values needed for the numerical procedure and contact force computation.

Ground Contact Parameters		
Letter	**Name**	**Value**
μ	Dynamic friction coefficient	0.3
ε_N	Normal restitution factor	0
ε_T	Tangential restitution factor	0
Numerical Parameters		
N	Number of iteration steps	10000
n	Number of contact points	10
Δt	Integration step	0.0005

As we've seen, $\mathbf{C}_k$ and $\mathbf{Z}_k$ can be formed in 2^n different ways, and some of these combinations solve the linear complementarity problem. The easiest but most tedious way, is to compute all possibilities enumeratively, i.e. by trying. As in our model it is assumed that only two contact points can be touching the ground simultaneously, the numerical effort to solve the linear complementarity problem is reasonable, and simulations were found to be performed faster than real time on a normal PC. The Matlab code for solving the LCP can be found in appendix C.4.

4.6 Simulation Summary

On first glance, the physical equations and numerical procedures look utterly confusing and complicated. It takes undeniably some time to get used to the linear complementarity problem and non-smooth mechanics, but the benefits of such a method can be highly rewarding. Once the framework is set up, one can simply change the basic equations of motion (29) and redefine the contact points in order to run the new simulation. Furthermore, and this point might be the most valuable, one knows exactly what the code is doing and where its restrictions are. To sum up, a review of all steps for the simulation is presented in this section.
We've started with a model of our robot made out of rigid bodies, springs and dampers. Once the model is set, the equations of motion have to be derived in minimal coordinates $\mathbf{q}$, using the Newton-Euler equations or alternatively, the Lagrange equations. From this, we can extract the mass matrix $\mathbf{M}$ and the gyroscopic acceleration- and smooth force terms, $\mathbf{h}$. The contact points which will interact with the ground (or any other wall restricting the robot's motion) have to be defined by a distance function g_N which defines the contact to be active if it is negative or equal to zero. Now, the numerical code is executed and is checking for closed contacts. If some contacts happen to be closed, the index set $\mathbb{H}$ needs to be formed, which contains information on closed contacts. The generalized force direction matrices $\mathbf{W}_N$ and $\mathbf{W}_T$ will be computed, out of which the linear complementarity problem (LCP) is be formed. The LCP solver tries to find a solution for the given problem in order to obtain the contact forces in tangential- and normal direction. If no solution is found, the time step of the last simulation is altered slightly and is recomputed. The resulting forces

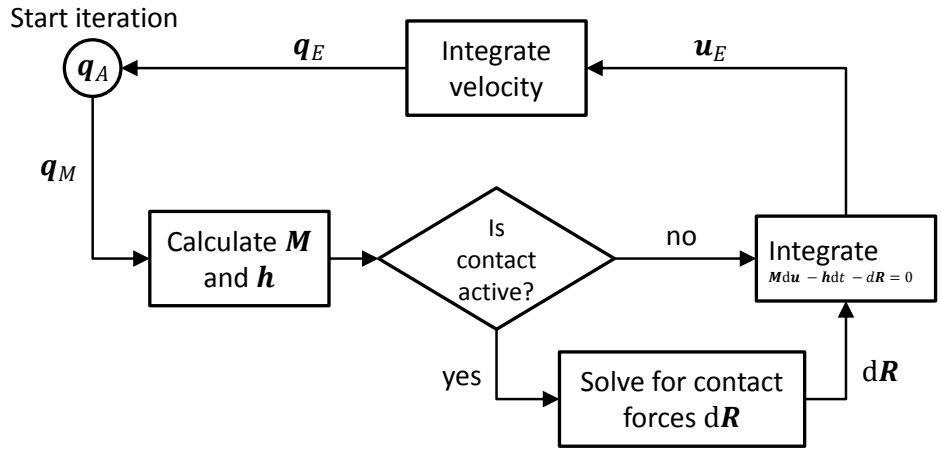


Figure 4.1: Illustration of the numerical procedure for the simulation. If the robot is in flight phase, the described time-stepping algorithm is simply integrating the equations of motion numerically. The non-trivial part comprises the computation of the contact forces d$\mathbf{R}$, given that the robot is touching the ground.

can then be inserted into the equations of motion, which can now be integrated numerically. Finally, the new states for the next iteration step are available and the procedure starts over again.
The full code for the numerical simulation is presented in appendix C.

Chapter 5

Realization of CHIARO

After having implemented the theoretical framework for the simulation, first results indicated promising behavior of the robot. This was the hint needed that a real world robot of the described system could actually perform the desired hopping motion.
In this chapter, the realization of the model to a real world system is presented. First, the computer-aided design (CAD) model is introduced and second, the materials and devices used for this robot are described.

5.1 CAD Model

The theoretical model of a system is often hard to design, as certain assumptions, simplifications and ideal properties are hard to realize in a robot prototype.
First of all, the model assumed only a two dimensional world, where no sideway-motion is possible. However, since our world is three dimensional, we can not simply neglect this spacial direction. In fact, not only the lateral displacement and motion of the robot is neglected, but also the pitching and rolling motion. In order to suppress unstable movements of those degrees of freedom, the robot is added two parallel feet, which make it hard for the robot to turn and fall sideways. The distance between the two feet was chosen sufficiently long, such that stability was guaranteed, but enough short to avoid excessive material usage. A distance of 0.2 meters was assumed to lead to desirable results.
The lower body of the robot was designed in a way, which provides light construction and robust properties. As this part is exerted to pure impacts with the ground without restitution of energy by a spring, light and stiff construction is key. Two pipes are connecting the lower leg with the two feet. Inside the pipes, husks were inserted on each side with a M5 thread, in order to quickly mount different kind of foot shapes to the robot. The two connecting pipes of the feet are attached to the lower leg, which is connected by the rotational joint, realized with two ball bearings, to the upper leg, in order to reduce friction losses. Furthermore, a gear-wheel is rigidly attached to the lower leg, which ensures a high transmission from the motor linked to it with a gear belt. The gear ratio is necessary, since the used EC motor (see next section) runs with a higher efficiency at higher angular velocities. Two springs are connected symmetrically to the lower leg. One is attached at the very top of the pipe and the other one

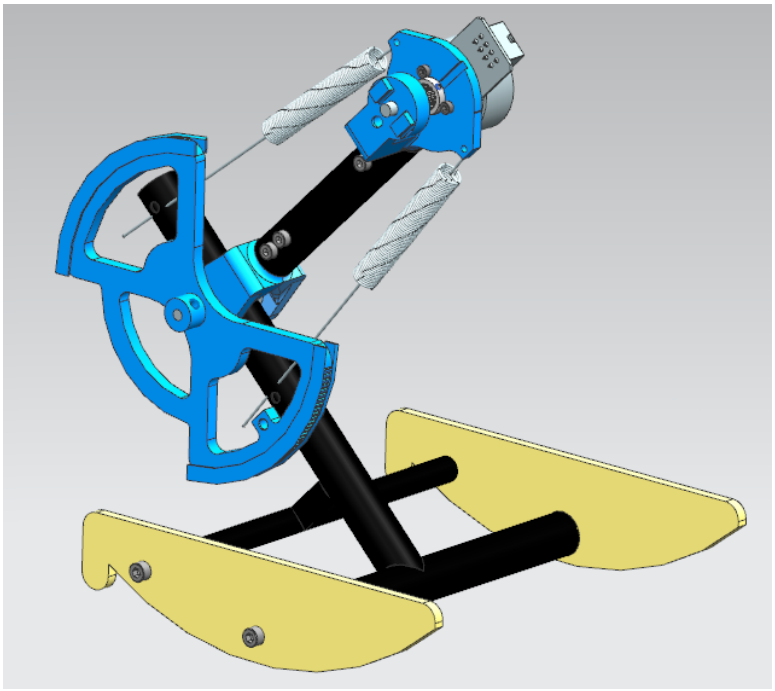


Figure 5.1: Illustration of the CAD model of CHIARO.

in the middle. Together, they form a push-pull spring, which is equivalent to the theoretical model as will also be discussed in the next section.
The upper leg is attached to a linking body, which encloses the two ball bearings. On the other side of the upper leg, the motor holding structure is mounted.
The motor hold is adapted to the used motor, such that it can be attached firmly to it. The shaft is supported on its tip with a ball-bearing in order to reduce radial load from the gear belt. On the shaft of the motor, a pulley is fixed with a grub screw, enabling transmission of the motor torque from the upper leg to the lower leg by using a gear belt. In addition, two screws are mounted to the motor hold, which are the supporting structures for the springs, originating from the lower leg. See figure 5.1 for an illustration of the described model. Note that all parameters stayed the same as in section 3.3.

5.2 Used Material

As light and stiff properties are required for the lower foot, it was decided to use thin walled carbon fiber tubes. Their low density and high robustness offers desirable properties for the required specifications. The feet were chosen to be made out of plywood, as it can be processed easily and has stable and light material properties.
Next to this, there was the opportunity to use a 3-D printer. For parts which do not require a high tensile strength and have a complex structure, the material (polylactic acid, PLA) was used for the printed parts. Namely the joint between the lower leg and the upper leg, as well as the gear-wheel (150mm diameter)

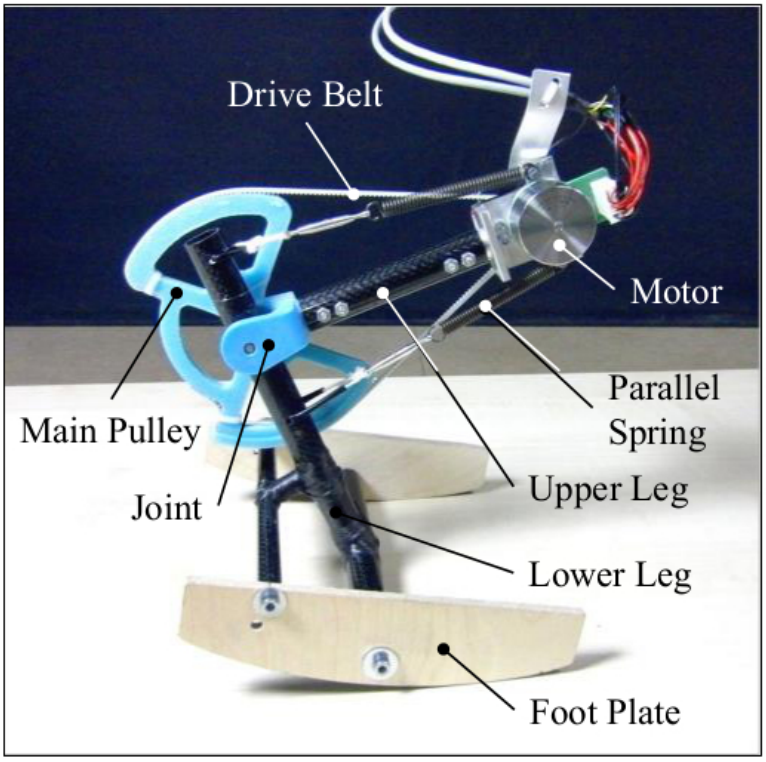


Figure 5.2: Illustration of real robot prototype.

were formed with that method.
The upper leg was again built with carbon fiber tubes, as there was material left from the lower leg.
For the motor hold, first attempts with PLA didn't reach the required precision. Since the mass of the upper leg is suspended with a spring, a denser material won't necessarily change the overall efficiency of the robot. Due to this fact, it was decided to use an aluminium structure instead of PLA, which will provide both, the required stability and precision.
As might be remembered from the previous chapters on the model description, a push-pull spring was assumed for the realization of the linear suspension. As it is hard to find push-pull springs in reality with a linear force-displacement relationship, two pull steel springs were symmetrically mounted instead, as can be seen in figure 5.2.
The most important part of the robot is its actuation system. The used motor is a Maxon EC 45 Flat with 70 Watts of power. The motor current is controlled by a ESCON 50/5 module. The pulley to transmit the force from the motor to the gear belt is made out of aluminium (12mm diameter).

Chapter 6

Experimental Conditions and Set-Up

In the last chapters, theoretical models have been described, and the realization of the first robot prototype was shown. Everything is ready to plan a proper experiment to analyze the influence of the robot's design on its dynamics.
In this chapter, the settings and conditions for the experiment, as well as the simulation are explained.

6.1 Simulation

The parameters for the simulation were kept as described in section 3.3. However, several other parameters come into play when one is to run the numerical simulation. As mentioned in previous chapters, the model assumes a Newtonian kinematic impact- and a Coulomb frictional model. Those two assumption add additional three parameters to each contact point, namely μ, the dynamic friction coefficient, ε_N, the normal restitution factor, and ε_T, the tangential restitution factor. A rigid body without elastic properties is assumed. As has already been listed in table 4.1, $\varepsilon_N = \varepsilon_T = 0$ seems an appropriate assumption for the given problem. The estimation for the friction coefficient is a harder task to accomplish. It was set to a value of $\mu = 0.3$, as this can be related to wood sliding on stone or wood.
Due to the curved shape of the robot's foot, the contact points on the foot need to be arranged in a way, such that they form an arced shape. The discretization can't be performed arbitrarily exact, as this would require a large number of contact points. However, assuming that the foot's shape is convex, only two neighbouring contact points can touch the flat ground at the same time. Due to this fact, it makes no difference for the implementation of the numerical procedure as for how many contact points are mounted to the foot. Having said that, when the midpoint displacement is calculated with a incremental time step Δt, more than two contact points might fall below the ground, depending on the given initial velocity $\mathbf{q}^A$. In order to avoid this problem, either the contact point number needs to be lowered, which will lead in polygonal shape rather than an arc, or the incremental time step Δt needs to be refined, in order to make sure that not more than two contact points get closed at the same time. The simu-

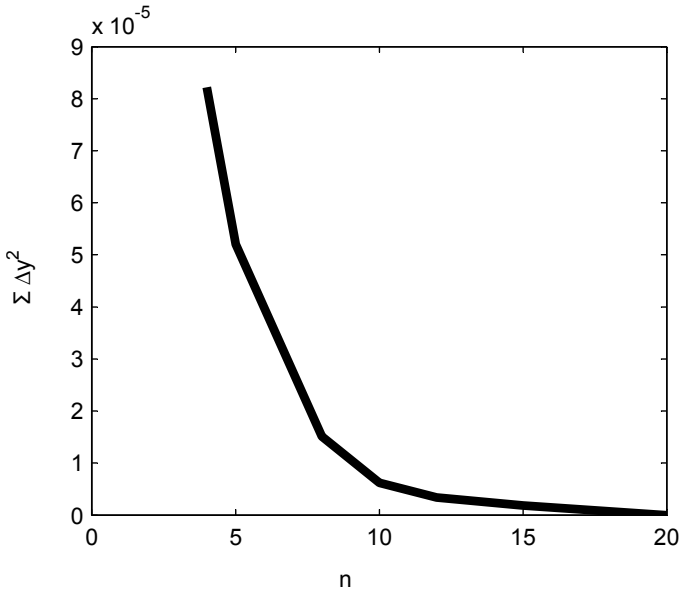


Figure 6.1: Convergence of the mean squared error of the hopping height y as a function of the number of contact points n at the robot's foot. Reference value is a simulation with $n = 20$. A contact point number of $n = 10$ was therefore found to lead to accurate results.

lation showed appropriate convergence and performance for a $\Delta t = 0.0005$ and a number of contact points on the curved foot of $n = 10$.
The last property of the simulation is the number of time steps, which defines the overall simulation time. A number of $N = 10'000$ was found to provide fast simulation without slowing down the computer due to storage allocation.

Note that for each simulation, and therefore also each experiment, the initial position of the robot is its static equilibrium. Hence, a transition phase from standstill to stable hopping occurs at every simulation.
Furthermore, energy losses occurred only in the rotational joint due to joint friction, and when impacts influenced the system. Motor- and gear losses, friction in the spring joints and aerodynamic drag were neglected.

6.2 Experiment

After having simulated the robot's motion, it is compared with experimental measurements. So far, it is not necessarily known if the simulated results have any relevance for the real physical system and if the made assumptions hold. In this section, the experimental environment and conditions are described.

As shown in the previous chapter, the motor used is a Maxon EC 45 Flat with 70 Watts of power, controlled by an ESCON 50/5 module. The trajectory of the motor current is set by a desktop computer, executed by Matlab, which sends its signal to a National Instruments SCB 68 digital-analogue converter. The ground on which the robot is hopping is wooden and flat. The energy source of the mo-

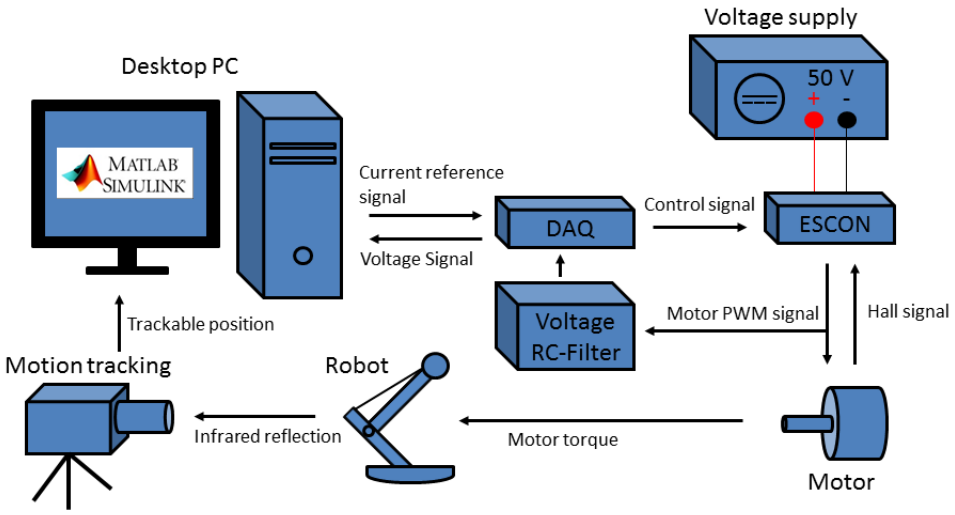


Figure 6.2: Experimental set-up of the robot.

tor is provided by a power source of 50 Volts. The signals and power sources are attached to the robot by a long cable, which enables the robot to move freely. An overview of the set up can be found in figure 6.2. Note that the cable from the voltage supply to the motor was suspended during the experiment, as otherwise it would have had a non-negligible influence on the measured results.

The cost of transport of the system can be measured by taking the energy expended by the motor and dividing it by the weight of the robot times the travelled distance, as has been presented in section 2.2.
The expended energy of the motor can be computed by taking the current flowing through the motor times the voltage drop. A peculiar thing about electronically commutated (EC, or brushless DC motors) is its input signal. A pulse-width modulation (PWM) signal is used to establish a desired voltage signal in the three motor windings, as is described in detail in appendix E. In the real world experiment, the voltage was measured of only one motor winding. As always two windings are turned on at the same time, the resulting power or one motor winding needs to be multiplied by two, which leads to the following cost of transport:

$$CoT = \frac{2 \int_t V_w \, I_c \, dt}{m \, g \, x_{end}}, \tag{42}$$

where V_w is the voltage measured at one motor winding and I_c is the controlled current. Note that the input current was thought to match the actual current, which seemed to be an appropriate assumption. The trajectory of the current was chosen to be sinusoidal, such that a sinusoidal torque trajectory would drive the robot:

$$I_c = A_c \, \sin \omega_c \, t \tag{43}$$

Since the PWM signal measurement of the motor winding was corrupted by noise due to its fast switching, the signal needed to be filtered. See also Furthermore, the SCB 68 digital-analogue converter (also DAQ for data acquisition)

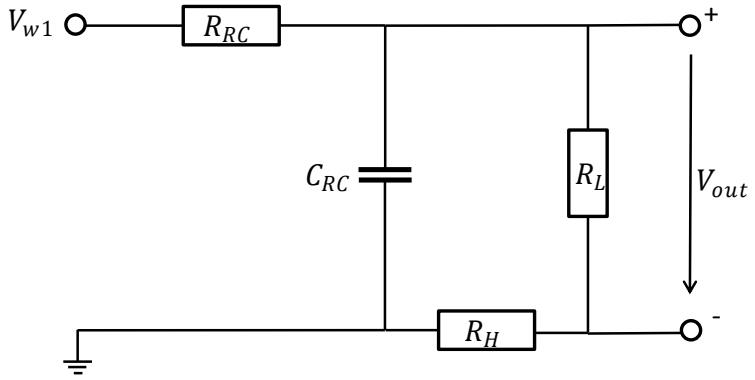


Figure 6.3: RC-Filter and voltage divider of the experimental set-up. $R_{RC} = 1.2\,\mathrm{k\Omega}$, $C_{RC} = 47\,\mu\mathrm{H}$. See also appendix E for its application.

can only measure up to 10 volt, which made it necessary to add a static voltage divider. An overview of the electronics can be found in figure 6.3. Note that the voltage divider decreases the voltage by a factor of 5:

$$V_{out} = V_{w1} \frac{R_L}{R_L + R_H}, \tag{44}$$

where V_{w1} is the motor winding voltage, $R_H = 100\,\mathrm{k\Omega}$, $R_L = 22.1\,\mathrm{k\Omega}$, and V_{out} is the measured voltage in the DAQ.

In order to get the motion of the robot into digital data, an OptiTrack motion tracking system was used. Using 12 cameras placed on top of a 10 cubic meter room and placing 9 trackables on the robot's structure, the motion was recorded.

Chapter 7

Results

In this chapter, the findings of the simulation described in the last chapter are presented. First however, to show that the simulation has any significance, it is compared to corresponding experimental measurements.

7.1 Verification of the Simulation

Two experimental measurements and two simulations with the same parameter settings are compared in this section. It is assumed, that if the two measurements match the simulation results, further simulation can be assumed to provide significant results without having to show the corresponding experiment. As measurements take a much longer time to conduct, simulations are the preferable choice for analysis. In fact, a simulation of five seconds in real time, takes only about three seconds on a standard PC. On the other hand, setting up a proper experiment can take up to one hour and more including the evaluation of the states during five seconds of experiment. Hence, given that a simulation takes 1200 times less time to conduct, it is much easier to analyze large sets of parameters, which will provide insights into the dynamics of the robot.

As there are 17 parameters (see table 3.1) which influence the dynamics of our system, a non-feasible amount of simulations would need to be performed if one wants to find out the influence of the parameters. A reduction of the parameters to analyze is a necessary condition to find powerful statements. It was decided to choose parameters for the examination, which are easily changed in the experiment. For example, exchanging the pipe of the lower leg is hard to accomplish, as it is firmly fixed to other parts of the robot. On the other hand, we have parameters such as the frequency of our sinusoidal current input, which can be changed within a second. From this perspective, the following variable parameters were chosen to be analyzed for their impact on the dynamics:

- Current frequency f_T
- Foot radius R
- Foot to lower leg angle β

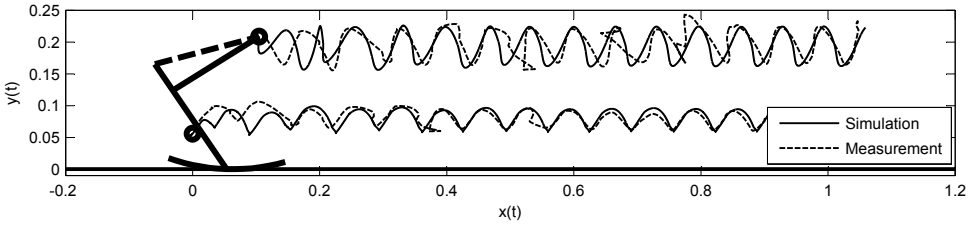


Figure 7.1: Comparison of the simulation trajectory with the corresponding experimental findings. $f_T = 3.4\,\text{Hz}$, $R = 0.3\,\text{m}$ and $\beta = 1.05\,\text{rad}$. Other parameters as in table 3.1.

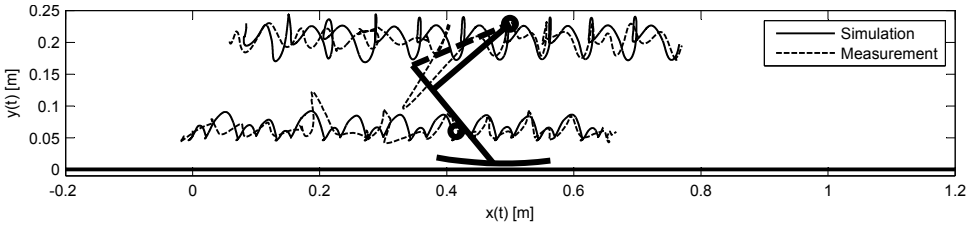


Figure 7.2: Comparison of the simulation trajectory with the corresponding experimental findings. $f_T = 4.3\,\text{Hz}$, $R = 0.6\,\text{m}$ and $\beta = 0.9\,\text{rad}$. Other parameters as in table 3.1.

Note that the robot was built in a way, which enables it to easily mount and dismount the feet. By manufacturing feet with different radii R, they can be changed in a short period of time. Additionally, the lower leg angle β depends on the through hole on the foot and is therefore quickly changed by drilling a new hole. The amplitude A_T of the torque signal was not chosen to be one of the changeable parameters due to missing time.

Two parameter sets have been simulated and verified experimentally. The first parameter set with $f_T = 3.4\,\text{Hz}$, $R = 0.3\,\text{m}$ and $\beta = 1.05\,\text{rad}$, as can be seen in figure 7.1, shows a matching trajectory with the motion tracking measurement. The given conditions lead to a stable hopping pattern after a short transition phase from standstill. Due to high noise in the OptiTrack system, some inconsistencies arised in the measured tracking position. Nevertheless, the measurement strongly supports the simulated result.
The second parameter set showed more chaotic properties than the first one. With $f_T = 4.3\,\text{Hz}$, $R = 0.6\,\text{m}$ and $\beta = 0.9\,\text{rad}$, the transition phase took much longer to change into a stable motion. As shown in figure 7.2, the measurement and the experiment start to show the same pattern not before a distance of 0.4 meters was covered. the reason for these diverging results might come from manufacturing imperfections, parameter variances or unmodeled effects. Having said that, the matching results after some steps even with this chaotic transition phase, strongly indicate that the simulation is able to describe the real robot's motion.

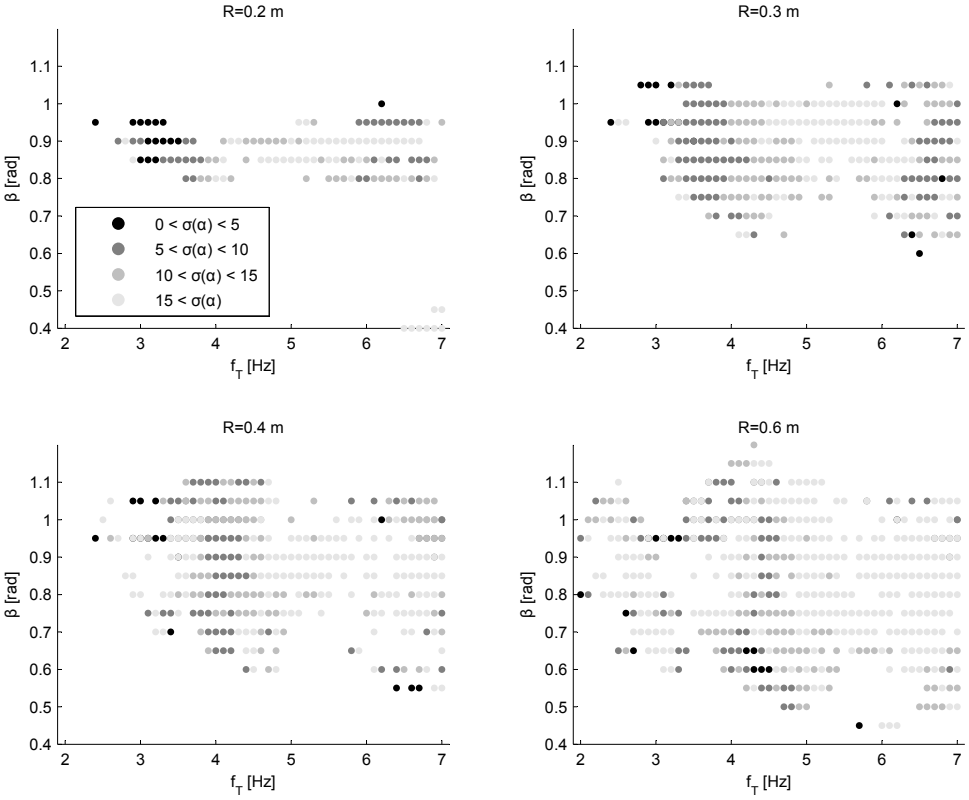


Figure 7.3: Simulation of the whole parameter space. Dots indicate stable runs, white areas unstable ones. The intensity of the dots' color is dependent on the standard deviation σ of the jumping angle α at takeoff of each hop, which gives a measure for attractivity of the fixed-point of α.

7.2 Simulation Results

Having shown that the experiment supports the simulated results, one can start to further investigate in simulation studies. Each of the three parameters mentioned in the last chapter was assigned to a discrete set of values, which spanned a parameter space to be tested, i.e.

- $f_T \in \{2, 2.1, ..., 6.9, 7\}$ Hz
- $R \in \{0.2, 0.3, 0.4, 0.5, 0.6\}$ m
- $\beta \in \{0.4, 0.45, ..., 1.15, 1.2\}$ rad

These parameter sets lead to a total number of 4000 combinations, each of which takes approximately three seconds to simulate. After each simulation, the results were saved in a data file. The most important measures for the following analysis were the jumping angle of the center of mass at take-off α (see figure 3.3), its standard deviation σ, the cost of transport CoT and the average forward velocity component of the robot, v_x.
Figure 7.3 shows the obtained results for different R, f_T and β. Each plot shows results for a different foot radius R, and value ranges of the standard deviation

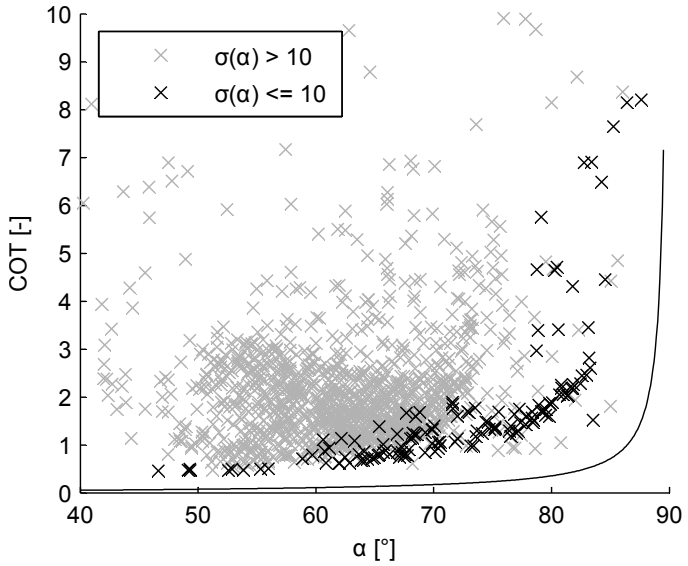


Figure 7.4: Jumping angle α versus the cost of transport. Black crosses indicate a faster convergence to the fixed point of α. The full line indicates the theoretical limit of the cost of transport, derived in section 2.2.

σ are indicated, which give a measure for stability as has been discussed in section 2.4. Note that the whole parameter space has been computed. Areas in the plots where no dots are drawn indicate a fall of the robot at any time during the simulation. Drawn dots, on the other hand, indicate a run, where the robot was able to stand throughout the simulation. What is most conspicuous is the broadening cloud depending on the foot radius. At smaller radii (or larger curvature), a narrow band of hopping without falling can be observed. As we go to larger radii, areas where the robot fell become smaller and smaller. This can be easily understood by the theoretical background from section 2.3. As the distance of the center of mass to the ground shifts below the foot radius, the system becomes statically unstable.
Another interesting behavior is the migrating stable hopping region with increasing foot radius. While the stablest hopping motion of the $R = 0.2\,\mathrm{m}$ plot, which is indicated by a small standard deviation of the jumping angle, is around $f_T = 3.2\,\mathrm{Hz}$ and $\beta = 0.9\,\mathrm{rad}$, the stable region moves gradually to higher frequencies and smaller β angles. In fact, the most stable region for the $R = 0.6m$ plot turns out to be around $f_T = 4.3\,\mathrm{Hz}$ and $\beta = 0.6\,\mathrm{rad}$.

The most interesting simulations are the ones indicating stable hopping patterns. The standard deviation of the jumping angle of the center of mass is a good measure to reveal stable hopping if a period-one pattern is created. However, it doesn't necessarily show period-two patterns, i.e. motions where every second jump shows the same jumping angle, but also odd numbers have a different but constant jumping angle. In the following analysis, only period-one patterns were uncovered.
By filtering the simulations for their standard deviation of the jumping angle, namely by ignoring all $\sigma \leq 10$, and requiring that the simulation shows at least

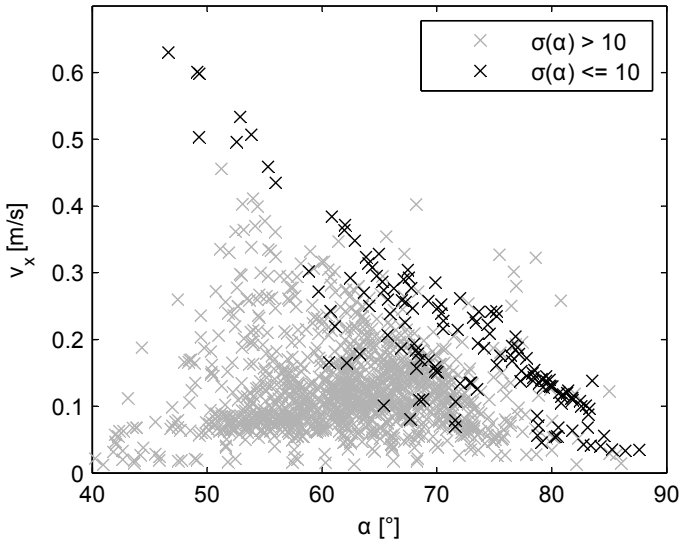


Figure 7.5: Jumping angle α versus the average forward speed v_x of a run. Black crosses indicate a faster convergence to the fixed point of α.

10 consecutive hops, truly stable hopping patterns can be found.
Figure 7.4 shows the distribution of those points in a graph, related to the jumping angle α and the cost of transport CoT. Furthermore, the derived lower bound for the cost of transport from section 2.2 is included. Obviously, simulations with stable hopping patterns seem to follow the path of the predicted lower bound. Furthermore, stable hopping simulations showed the highest energy efficiency.
In figure 7.5, the relation between the jumping angle α and the average forward velocity v_x is plotted. As can be seen, an almost linear relationship for stable hopping motion is visible.
An interesting question to ask is, if there were any parameters that have a direct influence on the jumping angle. If so, a feedforward controlling parameter could be found which can set the desired end velocity of the robot. As can be seen in figure 7.6, the relative angle between the lower leg and the foot, β comprises exactly those properties. For example for a foot radius of 0.3, the relationship between the jumping angle of the center of mass and the β angle is almost linear. As the frequency range of the stable hopping solutions is rather narrow, it can be assumed, that the β angle can be changed without having to adapt the motor frequency f_T. This would mean, that a feedforward tuning parameter has been found, which ensures a certain hopping velocity with a stable hopping pattern.

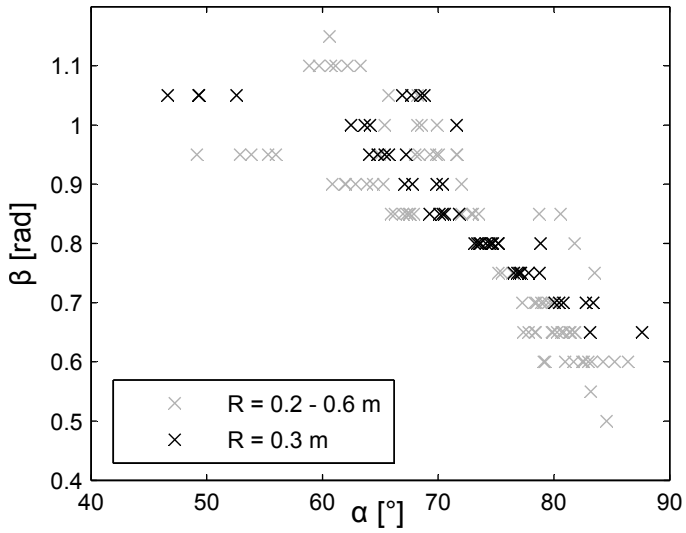


Figure 7.6: Jumping angle α versus the foot to lower leg angle β. All data points have a standard deviation σ of the jumping angle α smaller than 10. Black crosses indicate results of the parameter space, where $R = 0.3$.

Chapter 8

Conclusion

The main objective of this thesis was to investigate the self-stable properties of a hopping robot. Starting from previous, similar robots, the structure was simplified as much as possible, in order to making the dynamics of the system more transparent as for parameter influences. The reduction of the foot spring to a curved foot lead in fact to a reduction of the dimension of the equations of motion, but it added the foot radius R to the long list of system parameters.
After studying possible methods of analyzing non-smooth mechanical systems analytically as presented in (Leine 2004) or (Leine 2008), it quickly became clear, that even for a simple system as it is dealt with here, it is nearly impossible to read the intrinsic, self-stable properties from the equations of motion directly.
To gain insights into the dynamics of the system, a more heuristic approach was needed. As one of the goals is to achieve high energy efficiency of the robot's locomotion, it makes sense to focus on the cost of transport and find its influencing parameters. By assuming that there are mainly impact losses in normal direction while running, a relationship between the jumping angle α of the robot's center of mass and the cost of transport has been found. The derived function provides a lower bound for the cost of transport, which can not be overpassed in a real system.
Subsequently, a numerical model was developed to analyze the motion of the designed robot. As was shown in the last chapter, it was possible to validate the simulation's result with an experiment. Stable hopping patterns were found to be bound by a lower limit of the cost of transport, as predicted in the theoretical, simplified model of section 2.2. This strongly indicates, that even though the considered problem inhibits a detailed analysis of the equations of motion due to its structural changes of the differential equation and its high degree of non-linearity, it still obeys basic rules and principles.
Starting from there, one can ask the question if it is possible to find parameters, which show stable hopping for certain regions of the parameter set. By defining the standard deviation of the jumping angle α as a measure for stable hopping, It was possible to find such sets. What is most intriguing, is the fact that stable hopping patterns show the lowest cost of transport, i.e. are the most efficient ways of locomotion. Consistency of the jumping angle α seems to be the key to efficient locomotion.
The speed of the robot was found to be hard to control by changing the energy

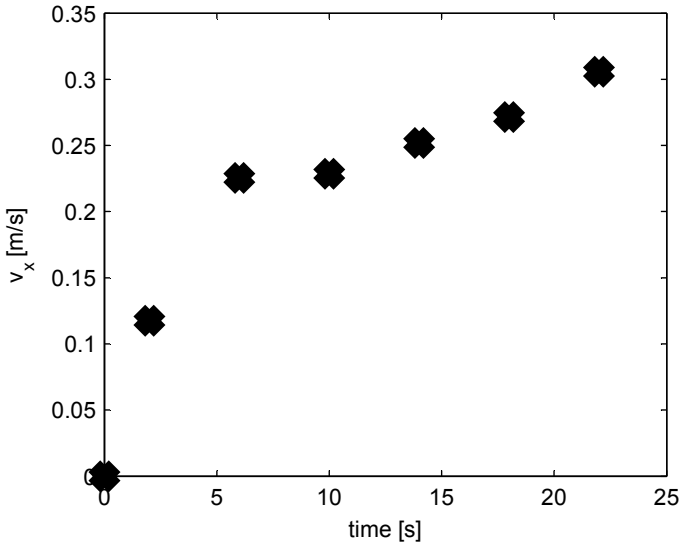


Figure 8.1: Speed of ongoing simulation with gradually changing lower-leg to foot angle β. Every four seconds, the angle is increased instantaneously by 0.05, starting from $\beta = 0.8\,\text{rad}$ to $\beta = 1.1\,\text{rad}$. The indicated data points show the average speed of constant β-intervals.

put into the system. For example the relation between forward speed and energy put into the system is nowhere near to linear, and it is not even clear, if an increase of the motor torque will lead to more stable- or unstable hopping pattern. However, by changing geometrical system parameters such as the foot radius R or the angle between the lower leg and the foot β, which corresponds to a shift of the center of mass in static equilibrium, the speed can be varied with an almost linear relationship and without the use of any feedback control.

Possible future projects might want to analyze the influence of the center of mass relative to the ground contact point, as this parameter was found to have a significant influence on the resulting motion pattern. Furthermore, it might be highly interesting to add a hip to the system to stabilize slight changes of the jumping angle by a feedback controller. The system might then be operated in a hybrid way, i.e. is using the hip correction for rough terrain and when the robot is hopping on a flat path, switch to the energy efficient feedforward control with the self-stable properties of the robot.

Feedforward controlled robots have the ability to move with an efficiency close to biological systems. Even though they are lacking of flexibility and versatility, understanding their dynamics can help feedback controlled systems to minimize the effort of controlling the locomotion of a highly non-linear system, such as a hopping robot.

Bibliography

(Adamczyk 2006) P. G. Adamczyk, S. H. Collins and A. D. Kuo. 2006. *The advantages of a rolling foot in human walking.* The Journal of Experimental Biology 209: 3953-3963.

(Ahmadi 2006) M. Ahmadi, M. Buehler. 2006. *Controlled Passive Dynamic Running Experiments With the ARL-Monopod II.* IEEE Transactions on Robotics 22(5):Page.

(Alexander 1975) R. McN. Alexander and Alexandra Vernon. 1975. *The mechanics of hopping by kangaroos (Macropodidae).* Journal of Zoology 177: 265-303.

(Alexander 1990) R. McN. Alexander. 1990. *Three Uses for Springs in Legged Locomotion.* The International Journal of Robotics Research 9(2): 53-61.

(Alexander 1995) R. McN. Alexander. 1995. *Leg Design and Jumping Technique for Humans, other Vertebrates and Insects.* Philosophical Transactions: Biological Sciences 347(1321): 235-248.

(Berkemeier 1998) M. D. Berkemeier. 1998. *Modeling the Dynamics of Quadrupedal Running.* The International Journal of Robotics Research 17(9): 971-985.

(Bertram 2009) J. E. A. Bertram and Anne Gutmann. 2009. *Motions of the running horse and cheetah revisited: fundamental mechanics of the transverse and rotary gallop.* Journal of the Royal Society Interface 6: 549-559.

(Blickhan 1989) R. Blickhan. 1989. *The spring-mass model for running and hopping.* Journal of Biomechanics 22(11/12): 1217-1227.

(Blickhan 2007) R. Blickhan, A. Seyfarth, H. Geyer, S. Grimm, H. Wagner, M. Guenther. 2007. *Intelligence by mechanics.* Philosophical Transactions of the Royal Society A 365: 199-220.

(Collins 2005) S. Collins, A. Ruina, R. Tedrake, M. Wisse. 2005. *Efficient Bipedal Robots Based on Passive-Dynamic Walkers.* Science 307: 1082-1085.

(Coros 2011) S. Coros, A. Karpathy, B. Jones, L. Reveret, M. van de Panne. 2011. *Locomotion skills for simulated quadrupeds.* ACM Transactions on Graphics (TOG) 30(4): Article 59.

(Cotton 2012) S. Cotton, I. M. C. Olaru, M. Bellman, T. van der Ven, J. Godowski, J. Pratt. 2012. *FastRunner: A Fast, Efficient and Robust Bipedal Robot. Concept and Planar Simulation.* Bio-Inspired Robotic Lab, Institute for Human and Machine Cognition, Pensacola, Florida.

(Diedrich 1995) F. J. Diedrich and W. H. Warren, Jr. 1995. *Why Change Gaits? Dynamics of the Walk-Run Transition.* Journal of Experimental Psychology: Human Perception and Performance 21(1): 183-202.

(Garcia 1998) M. Garcia, A. Chatterjee, A. Ruina, M. Coleman. 1998. *The Simplest Walking Model: Stability, Complexity, and Scaling.* Journal of Biomechanical Engineering 120: 281-288.

(Gerritsen 1995) K. G. M. Gerritsen, A. J. van den Bogert and B. M. Nigg. 1995. *Direct dynamics simulation of the impact phase in heel-toe running.* Journal of Biomechanics 28(6): 661-668.

(Glocker 2001) C. Glocker. 2001. *Set-Valued Force Laws.* Lecture Notes in Applied and Computational Mechanics 1. Springer Verlag. ISBN 978-3-540-41436-0.

(Glocker 2005) C. Glocker and C. Studer. 2005. *Formulation and Preparation for Numerical Evaluation of Linear Complementarity Systems in Dynamics.* Multibody System Dynamics 13: 447-463.

(Gregorio 1997) P. Gregorio, M. Ahmadi, and M. Buehler. 1997. *Design, Control, and Energetics of an Electrically Actuated Legged Robot.* Publication IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 27(4): 626-634.

(Holmes 2006) P. Holmes, R. J. Full, D. E. Koditschek and J. Guckenheimer. 2006. *The Dynamics of Legged Locomotion: Models, Analyses, and Challenges.* SIAM Review 48(2): 207-304.

(Hutter 2012) M. Hutter, C. Gehring, M. Bloesch, M. A. Hoepflinger, C. D. Remy and R. Siegwart. 2012. *StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion.* 15th International Conference on Climbing and Walking Robot, July 23-26, 2012, Johns Hopkins University, USA.

(Iida 2012) F. Iida, M. Reis, N. Maheshwari, K. Gunura, and S. Hauser. 2012. *Legged Robot Locomotion Based on Free Vibration.* 12th International Workshop on Advanced Motion Control, March 25-27, Sarajevo, Bosnia and Herzegovina.

(Kar 2003) D. C. Kar, K. Kurien Isaac, K. Jayarajan. 2003. *Gaits and energetics in terrestrial legged locomotion.* Mechanism and Machine Theory 38: 355-366.

(Kokkevis 1995) E. Kokkevis, D. Metaxas, N. I. Badler. *Autonomous Animation and Control of Four-Legged Animals.* Graphics Interface.

(Kuo 2007) A. D. Kuo. 2007. *Choosing Your Steps Carefully: Trade-Offs Between Economy and Versatility in Dynamic Walking Bipedal Robots.* IEEE Robotics & Automation Magazine: 18-29.

(Leine 2004) R. I. Leine, and H. Nijmeijer. 2004. *Dynamics and Bifurcations of Non-Smooth Mechanical Systems.* Lecture Notes in Applied and Computational Mechanics 18, Berlin Heidelberg New-York, Springer-Verlag.

(Leine 2008) R. I. Leine, and N. vand de Wouw. 2008. *Stability and Convergence of Mechanical Systems with Unilateral Constraints.* Lecture Notes in Applied and Computational Mechanics 36, Berlin Heidelberg New-York, Springer-Verlag.

(Lewis 2011) M. A. Lewis, M. R. Bunting, B. Salemi, and H. Hoffmann. 2011. *Toward Ultra High Speed Locomotors: Design and Test of a Cheetah Robot Hind Limb.* IEEE International Conference on Robotics and Automation, May 9-13, Shanghai, China: 1990-1996.

(Mathis 2013) B. Mathis. 2013. *Efficient locomotion of a segmented beam hopper based on free vibration.* MS thesis ETH Zurich, Zurich, Switzerland.

(McGeer 1990) T. McGeer. 1990. *Passive Dynamic Walking.* The International Journal of Robotics Research 9(2): 62-82.

(Nelson 2012) G. Nelson, A. Saunders, N. Neville, B. Swiling, J. Bondaryk, D. Billings, C. Lee, R. Playter, and M. Raibert. 2012. *PETMAN: A Humanoid Robot for Testing Chemical Protective Clothing.* Journal of the Robotics Society of Japan 30(4): 372-377.

(Owaki 2010) D. Owaki, M. Koyama, S. Yamaguchi, S. Kubo, and A. Ishiguro. 2010. *A Two-Dimensional Passive Dynamic Running Biped with Knees.* IEEE International Conference on Robotics and Automation, May 3-8, Anchorage, Alaska, USA: 5237-5242.

(Owaki 2011) D. Owaki, M. Koyama, S. Yamaguchi, S. Kubo, and A. Ishiguro. 2011. *A 2-D Passive-Dynamic-Running Biped with Elastic Elements.* IEEE Transactions on Robotics 27(1): 156-162.

(Raibert 1984) M. H. Raibert, H. B. Brown, and M. Chepponis. 1984. *Experiments in Balance with a 3D One-Legged Hopping Machine.* The International Journal of Robotics Research 3(2): 75-92.

(Raibert 1990) M. H. Raibert. *Trotting, Pacing and Bounding by a Quadruped Robot.* Journal of Biomechanics 23(1): 79-98.

(Raibert 2008) M. Raibert, K. Blankespoor, G. Nelson, and R. Playter. 2008. *BigDog, the Rough-Terrain Quadruped Robot.* Proceedings of the 17th World Congress The International Federation of Automatic Control, July 6-11, Seoul, Korea: 10822-10825.

(Raynor 2002) A. J. Raynor, C. J. Yi, B. Abernethy, Q. J. Jong. 2002. *Are transitions in human gait determined by mechanical, kinetic or energetic factors?.* Human Movement Science 21: 785-805.

(Reis 2011) M. Reis and F. Iida. 2011. *Hopping Robot Based on Free Vibration of an Elastic Curved Beam.* IEEE/ASME International Conference on Advanced Intelligent Mechatronics, July 3-7, Budapest, Hungary: 892-897.

(Maheshwari 2012) N. Maheshwari, X. Yu, M. Reis, and F. Iida. 2012. *Resonance Based Multi-Gaited Robot Locomotion.* IEEE/RSJ International Conference on Intelligent Robots and Systems, October 7-12, Vilamoura, Algrave, Portugal: 169-174.

(Ringrose 1997) R. P. Ringrose. 1997. *Self Stabilizing Running.* Ph.D. thesis, Massachusetts Institute of Technology, Boston, USA.

(Roberts 1998) T. J. Roberts, R. Kram, P. G. Weyand, and R. Taylor. 1998. *Energetics of Bipedal Running.* The Journal of Experimental Biology 201: 2745-2751.

(Ruina 2005) A. Ruina, J. E. A. Bertram, M. Srinivasan. 2005. *A collisional model of the energetic cost of support work qualitatively explains leg sequencing in walking and galloping, pseuto-elastic leg behaviour in running and the walk-to-run transition.* Journal of Theoretical Biology 237: 170-192.

(Rummel 2008) J. Rummel and A. Seyfarth. 2008. *Stable Running with Segmented Legs.* The International Journal of Robotics Research 27(8): 919-934.

(Sakagami 2002) Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. 2002. *The intelligent ASIMO: System overview and integration.* IEEE/RSJ International Conference on Intelligent Robots and Systems, October, EPFL, Lausanne, Switzerland: 2478-2483.

(Saranli 2001) U. Saranli, M. Buehler, and D. E. Koditschek. 2001. *RHex: A Simple and Highly Mobile Hexapod Robot.* The International Journal of Robotics Research 20(7): 616-631.

(Sayyad 2007) A. Sayyad, B. Seth, and P. Seshu. 2007. *Single-legged hopping robotics research - A review.* Robotica 25: 587-613.

(Semini 2010) C. Semini. 2010. *HyQ - Design and Development of a Hydraulically Actuated Quadruped Robot.* Ph.D. thesis, University of Genoa and Italian Institute of Technology, Italy.

(Tedrake 2002) R. Tedrake and H.S. Seung. 2002. *Improved Dynamic Stability Using Reinforcement Learning.* Leg Laboratory, MIT, Cambridge MA, USA.

(Tedrake 2004) R. Tedrake, T. W. Zhang, and H. S. Seung. 2004. *Stochastic Policy Gradient Reinforcement Learning on a Simple 2D Biped.* Intelligent Robots and Systems Proceedings 3, 28. September - 2. October: 2849-2854.

(Tucker 1970) V. A. Tucker. 1970. *Energetic Cost of Locomotion in Animals.* Comparative Biochemistry and Physiology 34: 841-846.

(Zeglin 1999) G. Zeglin. 1999. *The Bow Leg Hopping Robot.* Ph.D. thesis, The Robotics Institute, Carnegie Mellon University, Pittsburhg, Pennsylvania, USA.

Appendix A

List of Symbols and Abbreviations

A.1 Superscripts

+	property right after impact
-	property right before impact
*	Equilibrium
A	property at initial iteration
M	property at midpoint iteration
E	property at end of iteration

A.2 Subscripts

0	initial condition or relaxed spring position
i	iteration or element counter in vector or matrix
k	complementary element counter
l	lower body
N	Normal direction
s1	lower body center of mass
T	Tangential direction
u	upper body

A.3 Greek Letters

α	jumping angle of the center of mass at takeoff
β	lower body to foot angle
γ	landing angle of SLIP model or contact point speed
$\boldsymbol{\gamma}$	contact point speed vector
δ	neighbourhood equilibrium point
ε	restitution factor
$\boldsymbol{\varepsilon}$	diagonal restitution factor matrix
ϵ	bound of equilibrium point
ζ	turning angle for static equilibrium
θ	planar moment of inertia
$\boldsymbol{\Theta}$	3D moment of inertia
$\mathrm{d}\Lambda$	contact impulse measure
$\boldsymbol{\Lambda}$	contact impulse measure vector
μ	dynamic friction coefficient
$\boldsymbol{\mu}$	diagonal dynamic friction coefficient matrix
ξ	velocity difference before- and after impact
$\boldsymbol{\xi}$	velocity difference before- and after impact vector
σ	standard deviation of jumping angle α
ϕ	state angle SSIP
φ	robot knee angle
φ_0	initial knee angle for relaxed spring
φ	robot body absolute angle
$\boldsymbol{\psi}$	system solution
ω	natural frequency of spring-mass system
ω_r	rocking frequency of the linear SSIP model
ω_T	torque angular frequency
Ω	planar angular velocity
$\boldsymbol{\Omega}$	3D angular velocity

A.4 Latin Letters

a_i	entry i of matrix $\mathbf{A}$
$\mathbf{A}$	quadratic matrix of the linear complementarity problem
A_T	torque amplitude
$\mathbf{b}$	vector of the linear complementarity problem
$\mathbf{c_i}$	complementarity cone element i
$\mathbf{C_k}$	complementarity cone k
d	damping coefficient
e_i	entry i of matrix $\mathbf{E}$
$\mathbf{E}$	identity matrix
E_{exp}	expended energy
E_{kin}	kinetic energy
f_T	motor torque frequency
f	minimal degrees of freedom of the system
$\mathbf{F}$	force
g	gravitational acceleration
g_N	distance between ground and contact point of robot
h	center of mass to foot height
$\mathbf{h}$	gyroscopic accelerations and smooth forces
$\mathbf{J}$	jacobian
k	stiffness coefficient
l_0	relaxed spring length
l	robot body length
l_h	robot moment arm spring
m	mass
$\mathbf{M}$	mass matrix
$\mathbf{M_Q}$	moment
n	number of contact points on arced foot
N	number of iterations in simulation
$\mathbf{N}$	spin

$\mathbf{p}$	momentum
P_{exp}	expended power
$\mathbf{q}$	system state in minimal coordinates
$\mathbf{r}$	position vector
R	radius
$\mathrm{d}\mathbf{R}$	atomic non-smooth force measure
s_f	foot length
Δt	time increment
t	time
T_M	motor torque
$\mathbf{u}$	velocity of system state in minimal coordinates
v	speed
$\boldsymbol{v}$	3D velocity
$\mathbf{w}$	generalized force direction
$\mathbf{W}$	generalized force matrix
x	horizontal displacement
$\boldsymbol{x}$	system state
x_{end}	traveled distance
$\mathbf{x}$	complementary variable
X	set including x
y	vertical displacement
$\mathbf{y}$	complementary variable
Y	set including y
z_i	complementarity vector element i
$\mathbf{Z_k}$	complementarity vector k

A.5 Special Symbols

$\mathbb{H}$	set of closed contacts
$\mathbb{R}$	real numbers

A.6 Abbreviations

BLDC	Brushless DC Motor
CAD	Computer Aided Design
CHIARO	C-shaped Hopping and Impact Adapting RObot
CoM	Center of Mass
CoT	Cost of Transport
DAQ	Data Acquisition (device)
DC	Direct Current
EC	Electronically Commutated (motor)
EoM	Equations of Motion
LCP	Linear Complementarity Problem
PC	Personal Computer
PLA	Polylactic Acid
PWM	Pulse Width Modulation (signal)
RC	Resistor-Capacitor (filter)
Sgn	Set valued Signum function
SCB	Shielded desktop Connector Block
SLIP	Spring Loaded Inverted Pendulum (model)
SSIP	Self Stable Inverted Pendulum
Upr	Unilateral primitive function

Appendix B

Static Equilibrium Position

Given a foot to lower leg angle β, the static equilibrium position of the robot changes. The position is needed for the start of the numerical simulation. Also, the equilibrium position is the starting point for the experimental measurements.
The equilibrium point of the foot is located, where an extension of the vector from the curved foot rotational center to the center of mass of the robot touches the foot arc.

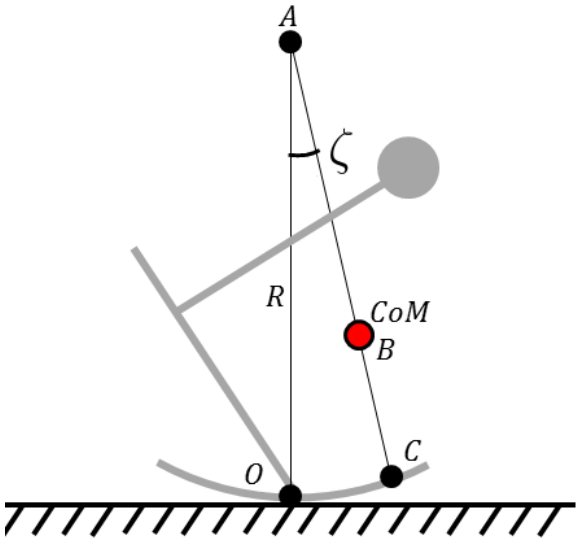


Figure B.1: Illustration of the new equilibrium position C of the curbed foot for a changing center of mass (CoM).

As shown in figure B.1, we'll need to find the angle ζ, that describes the rotational angle between the standard initial condition, where the center of the curved foot O is touching the ground, and the static equilibrium position, where

the new point C is touching the ground. Given

$$\mathbf{r}_O = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mathbf{r}_{OA} = \begin{pmatrix} 0 \\ R \end{pmatrix}, \quad \mathbf{r}_{CoM} = \mathbf{r}_B, \quad \mathbf{r}_C,$$

we can find the angle ζ easily by

$$\zeta = \cos^{-1}\left(\frac{\mathbf{r}_{AO} \cdot \mathbf{r}_{AC}}{\|\mathbf{r}_{AO}\|\|\mathbf{r}_{AC}\|}\right)$$

Appendix C

Numerical Code

The numerical code for the simulation was implemented in Matlab. One parameter file, an executive file, a function for the index allocation and another one for solving the LCP form all needed parts. If all four files are saved in the executive folder and the Run_DS.m file is executed, the simulation will start automatically. in the Parameters.m file, system properties can easily be changed.

C.1 Parameters.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Robot Leg with Curved Foot  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Fabio Giardina 2013         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Parameters                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
close all
clc

% All values in SI units

%% Simulation settings
dt_i = 0.0005;              % Time increment [s] and integration step
N = 10000;                  % Number of iterations
zz_top = 20;                % Number of time to change time increment when
                            % no solution for the LCP was found

%% Animation settings
kk = 400;                   % Number of pictures to be taken throughout
     simulation
ww = 2.2;                   % Window width
wh = 0.4;                   % Window height

%% Geometry
l_l = 0.2;                  % Length, lower leg
l_u = 0.16;                 % Length, upper leg
l_h = l_l/4;                % Length, moment arm spring
l_D = l_l/2 - l_h;          % Length, distance CoM lower leg to knee joint
phi_l_0 = pi/4;             % Angle, (relaxed spring) lower leg
phi_u_0 = pi/4;             % Angle, (relaxed spring) upper leg
dff = 0;                    % Shift, foot relative to lower rod

A_IL0 = [cos(phi_l_0),sin(phi_l_0);...      % Rotational matrix, lower leg
        -sin(phi_l_0),cos(phi_l_0)];        % to inertial coordinate frame

A_IU0 = [cos(phi_u_0),-sin(phi_u_0);...     % Rotational matrix, upper leg
         sin(phi_u_0),cos(phi_u_0)];        % to inertial coordinate frame
```

```
r_sl = A_IL0^-1*[-0.0628469;0.044985];      % Position vector, CoM CAD data

r_ml = A_IL0^-1*[-l_l/2*cos(phi_l_0);...    % Position vector, middle
                  l_l/2*sin(phi_l_0)];      % of lower tube

r_G = r_ml + [-l_l/2;0];    % Position vector, end of lower tube
l = r_ml - r_sl;            % Vector, CoM lower leg to middle of lower tube
l_G = l_u;                  % Length, CoM upper leg

%% Inertias
m_l = 0.239;                % Mass, lower leg
m_u = 0.481;                % Mass, total upper body

J_l = 1471*10^-6;           % Moment of inertia, lower leg
J_u = 1579*10^-6;           % Moment of inertia, upper leg

%% Force elements
k = 3022;                   % Linear stiffness, spring
d = 0.06;                   % Linear dissipation, damper
g = 9.81;                   % Acceleration, gravity
r_g = 155/12;               % Ratio, gear
Am = 4/10*0.95;             % Current, motor current amplitude
t_c = 0.131;                % Constant, torque constant
A_T = r_g*t_c*Am;           % Moment, amplitude of motor moment
T_del = 0;                  % Iteration, time delay of motor moment
omega_T = 3.4*2*pi;         % Angular frequency, motor
phi_r = pi;                 % Angle, phase shift motor

%% Contact parameters
eps_N = 0;                  % Normal restitution
eps_T = 0;                  % Tangential restitution
mu = 0.33;                  % Friction coefficients

n = 10;                     % Number of contact points at foot (counted from
    left to right)
R = 0.3;                    % Length, foot radius
s_f = 0.2;                  % Length, secant of the foot
b_i = s_f/n;
alpha_i = b_i/R;            % Angle, angle between contact points on foot arc
beta = 1.05;                % Angle, relative angle between foot and lower
    leg


%% Static Equilibrium
% See appendix XX of Master thesis (ETH Zurich, Fabio Giardina 2013)

r_sl1 = [-0.0628469;0.044985];              % Initial position of CoM lower
                                            % leg in CAD model from ground
A_IL0 = [cos(beta),sin(beta);...            % Rotational matrix, lower leg
        -sin(beta),cos(beta)];              % to inertial coordinate frame

A_IU0 = [cos(pi/2-beta),-sin(pi/2-beta);... % Rotational matrix, upper leg
         sin(pi/2-beta),cos(pi/2-beta)];    % to inertial coordinate frame

A_beta = [cos(pi/4-beta),-sin(pi/4-beta);...% Rotational matrix, lower leg
        sin(pi/4-beta),cos(pi/4-beta)];     % to foot angle coordinate frame

r_om =  A_beta*(-r_sl1)+ A_IL0*[-l_l/2;0];  % Position, middle point lower
                                            % leg
r_op = r_om + A_IL0*[-l_l/4;0];             % Position, joint
r_ol = r_op + A_IU0*[l_u;0];                % Position, motor
r_l_CoM = [0;0];                            % Position, CoM lower leg
r_u_CoM = r_ol;                             % Position, CoM upper leg
r_i_CoM = r_l_CoM*m_l/(m_l+m_u) +...        % Position, CoM system
          m_u/(m_l+m_u)*r_u_CoM;

r_f = A_beta*(-r_sl1);                      % Vector, CoM lower leg to
                                            % contact point turned by beta

r_a = -r_sl1+[0;R];                         % Vector, CoM lower leg to
                                            % middle point of foot arc

r_b = r_i_CoM;                              % Vector, CoM lower leg to CoM
                                            % system
```

```
r_af = r_f-r_a;                             % Vector, middle point arc to
                                            % rotated CoM lower leg

r_ab = r_b -r_a;                            % Vector, middle point arc to
                                            % system CoM

r_ac = r_ab/norm(r_ab)*norm(r_af);          % Vector, middle point arc to
                                            % equilibrium arc position

seq = acos(r_af'*r_ac/(norm(r_af)*...       % Turning angle of system for
        norm(r_ac)));                       % static equilibrium

% Check the turning direction
if r_ab(1) < 0
    seq = -seq;
end

A_seq = [cos(seq),sin(seq);...              % Rotational matrix, static
         -sin(seq),cos(seq)];               % equilibrium position

r_oc = -r_sl1+[0;R]+r_ac;                   % Vector, static equilibrium
                                            % point on arc

r_h = A_seq*(-r_oc);                        % Vector, lower leg CoM to
                                            % static equilibrium ground
                                            % contact point

%% Initial conditions for static equilibrium
x_0 = 0;                    % Lower leg, horizontal displacement
y_0 =  r_h(2);              % Lower leg, vertical displacement
v_y_0 = 0;                  % Lower leg, vertical velocity
v_x_0 = 0;                  % Lower leg, horizontal velocity
omega_l_0 = 0;              % Lower leg, angular velocity
omega_u_0 = 0;              % Upper leg, angular velocity
phi_l_p = beta + seq;       % Lower leg, angular position
phi_u_p = pi/2-phi_l_p;     % Upper leg, angular position

l_0 = sqrt(l_h^2+l_u^2-...  % Length, relaxed spring length
2*l_u*l_h*cos(pi-...
phi_l_0-phi_u_0));

%% Ground Energy
% Energy, standing position at ground. Note that the slacking due to the
% non-rigid spring is not taken into account!
E_0 = m_l*g*l_l/2*sin(phi_l_0) + ...
    m_u*g*(l_l/2*sin(phi_l_0) + ...
    l_D*sin(phi_l_0) + ...
    l_G*sin(phi_u_0))...
    +1/2*v_x_0^2*m_l...
    +1/2*v_y_0^2*m_l;
```

C.2 Run_DS.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Robot Leg with Curved Foot  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Fabio Giardina 2013         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Executive File              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

run Parameters

%% Initial conditions
q_A(:,1) = [x_0;y_0;phi_l_p;phi_u_p];           % Initial position in
                                                % minimal coordinates
u_A(:,1) = [v_x_0;v_y_0;omega_l_0;omega_u_0];   % Initial velocity in
                                                % minimal coordinates
t_A(1) = 0;                                     % Initial time
eps_N = ones(n,1)*eps_N;                        % Restitution vector normal
                                                %(tangential is zero)
mu_i = ones(n,1)*mu;                            % Friction coefficients
dt = dt_i;
```

```
NS = 0;                                             % Check variable for
                                                    % numerical solution
fall = 0;                                           % Check variable fall
k_gcc = 1;                                          % Iteration variable jump
k_gcc_l = 1;                                        % Iteration variable land

% Initialize animation elements
h_ul=line(0,0,'LineWidth',2);
h_ll=line(0,0,'color','b','LineWidth',2);
h_f=line(0,0,'color','b','LineWidth',2);
h_s=line(0,0,'color','k','LineWidth',2);
h_g=line(0,0,'color','k','LineWidth',2);

%% LCP solution check variable
% If no solution of the LCP should be found, zz is increased by 1 which
% leads to a recalculation of the last timestep with altered time increment
% dt. This is repeated maximally 20 times before a warning is displayed.
zz = 1;

%% Animation settings
ii = 1;                 % Initialize getframe parameter
ff = N/kk;              % Variable for getframe loop

%% Implementation of the time-stepping algorithm (Glocker & Studer 2005)
tic
for i = 1:N
    for j = 1:zz
        %% Step (i): Calculate midpoint and end time
        t_M = t_A(i) + 1/2*dt;

        % If no solution of the LCP exists, change dt and recalculate
        if zz >= 1
            dt = dt_i/zz;
            t_E = t_A(i) + dt;
        else
            t_E = t_A(i) + dt;
        end

        %% Step (ii): Calculate midpoint displacement
        q_M = q_A(:,i) + 1/2*dt*u_A(:,i);

        %% Step (iii): Matrix calculations
        % Free moment function
        if i > T_del
            T_t(i) = A_T*sin(omega_T*t_M-phi_r);
        else
            T_t(i) = 0;
        end

        % Jacobians
        % Lower leg translational jacobian CoM
        Ja_l = [1,0,0,0;0,1,0,0;0,0,0,0];

        % Upper leg translational jacobian CoM
        Ja_u = [1,  0   ,    cos(q_M(3))*l(2) - sin(q_M(3))*l(1) + l_D*sin(
            q_M(3)),   -l_G*sin(q_M(4));...
                0,  1   ,   -cos(q_M(3))*l(1) - sin(q_M(3))*l(2) + l_D*cos(
                    q_M(3)) ,   l_G*cos(q_M(4));...
                0,  0   ,          0        ,   0];

       % Upper leg translational jacobian CoM
        Ja_Fu = [1,  0   ,   cos(q_M(3))*l(2) - sin(q_M(3))*l(1) + l_D*sin(
            q_M(3)),   -l_u*sin(q_M(4));...
                0,  1   ,   -cos(q_M(3))*l(1) - sin(q_M(3))*l(2) + l_D*cos(
                    q_M(3)) ,   l_u*cos(q_M(4));...
                0,  0   ,          0        ,   0];

        % Upper leg translational jacobian spring connection point
        Ja_Fl = [1,  0   ,   cos(q_M(3))*l(2) - sin(q_M(3))*l(1) + l_l/2*sin(
            q_M(3)),   0;...        % Upper leg translational jacobian CoM
                0,  1   ,   -cos(q_M(3))*l(1) - sin(q_M(3))*l(2) + l_l/2*cos(
                    q_M(3)) ,   0;...
                0,  0   ,          0        ,   0];
```

```
% Lower leg rotational jacobian
Ja_Rl = [0,0,0,0;0,0,0,0;0,0,1,0];

% Upper leg rotational jacobian
Ja_Ru = [0,0,0,0;0,0,0,0;0,0,0,1];


% Mass matrix
M = [m_l+m_u          , 0 , m_u*(-l(1)*sin(q_M(3))+l(2)*cos(q_M(3))+
     l_D*sin(q_M(3))) , -l_G*sin(q_M(4))*m_u;...
       0               , m_l + m_u, m_u*(-cos(q_M(3))*l(1)-sin(q_M(3))
           *l(2)+l_D*cos(q_M(3)))  ,  l_G*cos(q_M(4))*m_u;...
       m_u*(cos(q_M(3))*l(2) - sin(q_M(3))*l(1) + l_D*sin(q_M(3)))  ,
            m_u*(l_D*cos(q_M(3)) - cos(q_M(3))*l(1) - sin(q_M(3))*l
           (2)) , J_l + m_u*(l(1)^2-2*l(1)*l_D+l(2)^2+l_D^2) , m_u*(
           l_G*l(1)*(sin(q_M(4))*sin(q_M(3))-cos(q_M(4))*cos(q_M(3)))
            + l_G*l_D*(cos(q_M(4))*cos(q_M(3))-sin(q_M(4))*sin(q_M(3)
           )) - l_G*l(2)*(sin(q_M(4))*cos(q_M(3)) + cos(q_M(4))*sin(
           q_M(3))));...
       m_u*(-l_G*sin(q_M(4)))   ,  m_u*(l_G*cos(q_M(4))) ,  m_u*(l(1)*
           l_G*(sin(q_M(4))*sin(q_M(3))-cos(q_M(4))*cos(q_M(3))) +
           l_G*l(2)*(-sin(q_M(4))*cos(q_M(3))-cos(q_M(4))*sin(q_M(3))
           )+l_G*l_D*(-sin(q_M(4))*sin(q_M(3))+cos(q_M(4))*cos(q_M(3)
           ))) ,   J_u + m_u*l_G^2];

% Left side of equations of motion
h_l = [m_u*u_A(3,i)^2*(-l(1)*cos(q_M(3)) - l(2)*sin(q_M(3))+l_D*cos(
     q_M(3))) - l_G*cos(q_M(4))*u_A(4,i)^2*m_u;...
        m_u*u_A(3,i)^2*( l(1)*sin(q_M(3)) - l(2)*cos(q_M(3))-l_D*sin(
            q_M(3))) - l_G*sin(q_M(4))*u_A(4,i)^2*m_u;...
       m_u*u_A(3,i)^2*( l(1)*l_G*(sin(q_M(3))*cos(q_M(4))+sin(q_M(4))
            *cos(q_M(3))) + l(2)*l_G*(sin(q_M(4))*sin(q_M(3))-cos(q_M
           (4))*cos(q_M(3)))) - m_u*u_A(4,i)^2*l_D*l_G*(sin(q_M(3))*
           cos(q_M(4))+sin(q_M(4))*cos(q_M(3)));...
       m_u*u_A(3,i)^2*( l(1)*l_G*(sin(q_M(4))*cos(q_M(3))+cos(q_M(4))
            *sin(q_M(3))) + l_G*l(2)*(sin(q_M(4))*sin(q_M(3))-cos(q_M
           (4))*cos(q_M(3)))  - l_G*l_D*(sin(q_M(4))*cos(q_M(3))+cos
           (q_M(4))*sin(q_M(3))))];

% Right hand side equations of motion
l_F = sqrt(l_h^2+l_u^2-2*l_u*l_h*cos(pi-q_M(4)-q_M(3)));
                        % Actual spring length
F_d = [l_h*cos(q_M(3))+l_u*cos(q_M(4));-l_h*sin(q_M(3))+l_u*sin(q_M
     (4));0]; % Force direction of the spring

%Inhomogeneous terms of equations of motion
h_r = (Ja_Fl'-Ja_Fu')*F_d/norm(F_d)*k*(l_F-l_0)...                  %
      Spring force
         +Ja_l'*[0;-m_l*g;0]+Ja_u'*[0;-m_u*g;0]...                  %
              Gravity
         -(Ja_Rl' + Ja_Ru')*[0;0;d*(u_A(4,i)+u_A(3,i))]...          %
              Damping moment
         +[0;0;T_t(i);T_t(i)];   % Motor moment

% Gyroscopic accelerations of the system (Measure equality: M*du-h*dt
     -dR=0)
h =  -h_l + h_r;

% Rotation matrix of body fixed coordinate frame of the lower leg
A_Il = [cos(q_M(3)-beta),sin(q_M(3)-beta),0;...
    -sin(q_M(3)-beta),cos(q_M(3)-beta),0;0,0,1];

% Calculate contact distances to ground and generalized force
% directions w_T(j) and w_N(j) of the contact j.
for j = 1:n
    %Contact point on curved foot arc computation
    if j <= n/2
        k_x = sin((j-1-n/2)*alpha_i)*R+b_i/2-dff;
        k_y = R*(1-cos((j-1-n/2)*alpha_i));
    else
        k_x = sin((j-1-n/2)*alpha_i)*R+b_i/2-dff;
        k_y = R*(1-cos((j-1-n/2)*alpha_i));
    end
   % Ground distances of each contact point
```

```
   g_N(j) = q_M(2)-l(1)*sin(q_M(3))+l(2)*cos(q_M(3))-l_l/2*sin(q_M(3)
       )+ A_Il(2,1)*k_x + A_Il(2,2)*k_y;

   % Generalized force directions of each contact point
   w_T(:,j) = [1;0;-l(1)*sin(q_M(3))+l(2)*cos(q_M(3))-l_l/2*sin(q_M
       (3))- sin(q_M(3)-beta)*k_x+cos(q_M(3)-beta)*k_y;0];
   w_N(:,j) = [0;1;-l(1)*cos(q_M(3))-l(2)*sin(q_M(3))-l_l/2*cos(q_M
       (3))- cos(q_M(3)-beta)*k_x-sin(q_M(3)-beta)*k_y;0];
end

% Assign indices for the active contacts
[i_1,i_2,index,W_N,W_T,CV] = indX(w_N,w_T,g_N,n);

% If more than two contacts are closed, display a warning (To
% avoid this probelm you might want to decrease the time increment
% vector dt or the number of contact points n).
if sum(CV) > 2
    disp('WARNING: more than two contacts are closed!')
    CV = 0;
end

%% Step (iv): Solve the linear complementarity problem
% if contact is active, rearrange Problem and form an LCP (Glocker &
    Studer 2005)
if i_1 > 0
        E = eye(length(index));
        Z = zeros(length(index));
        mu = eye(length(index));
        eps_Ni = 0;
        for j = 1:length(index)
            eps_Ni(j,j) = eps_N(index(j));
            mu(j,j) = mu_i(j);
        end

        % LCP problem matrix A
        A = [W_N'*M^-1*(W_N-W_T*mu),W_N'*M^-1*W_T,Z;...
             W_T'*M^-1*(W_N-W_T*mu),W_T'*M^-1*W_T,E;...
             2*mu,-E,Z];

        % LCP problem vector b
        b = [W_N'*M^-1*h*dt+(E+eps_Ni)*W_N'*u_A(:,i);...
             W_T'*M^-1*h*dt+(E)*W_T'*u_A(:,i);zeros(length(index),1)
                 ];

        % Solve LCP enumeratively. Note that NS is a check-
        % variable. If NS = 1 then there doesn't exist a solution
        % of the LCP
        [lambda_N,lambda_R,NS] = LCS(A,b,index);

        % In case there is no solution for the LCP, alter zz. The
        % reasons for not finding a solution might be due to a bad
        % modelling.
        if zz >= zz_top
            disp(['Warning!t = ', num2str(t_E),': Linear
                Complementarity Problem has no solution. Check the
                mathematical model.'])
            break
        end

    %% Step (v): Calculate u_E
        if NS == 1
            zz = zz + 1;
        else
            % Calculate u_E, the final velocity of iteration i
            u_E(:,1) = M^-1*(W_N-W_T*E*mu)*lambda_N + M^-1*W_T*
                lambda_R...
                  + M^-1*h*dt + u_A(:,i);
        end
else
    % If index number is zero, integrate u_E normally
    u_E = M^-1*h*dt + u_A(:,i);
end
% Set new time increment
if NS == 0 && zz > 1
```

```
        zz = 1;
        dt = dt_i;
    end

end
    %% Step (vi): Computation of q_E
    q_E = q_M + 1/2*dt*u_E(:,1);
    q_A(:,i+1) = q_E;
    u_A(:,i+1) = u_E;
    t_A(i+1) = t_E;

    %% System Energy
    % Define relevant vectors (Inertial coordinate system)
    A_IL = [cos(q_M(3)),sin(q_M(3));-sin(q_M(3)),cos(q_M(3))];
    A_IU = [cos(q_M(4)),-sin(q_M(4));sin(q_M(4)),cos(q_M(4))];

    r_SL = [q_E(1);q_E(2)];                                          %
        Position, CoM lower leg
    r_Ml = q_E(1:2) + A_IL*(l);                                      %
        Position, middle lower leg
    r_Ml(3,1) = 0;
    r_OP = r_Ml + [-l_l/2*cos(q_E(3));l_l/2*sin(q_E(3));0];          %
        Vector, origin to lower leg spring connection point
    r_OF = r_Ml + [l_l/2*cos(q_E(3));-l_l/2*sin(q_E(3));0];          %
        Vector, origin to lower leg contact point
    r_OG = r_Ml + [-l_D*cos(q_E(3));l_D*sin(q_E(3));0];              %
        Vector, origin to leg joint
    r_OM = r_OG + [l_u*cos(q_E(4));l_u*sin(q_E(4));0];               %
        Vector, origin to upper leg end point
    r_GU = [l_G*cos(q_E(4));l_G*sin(q_E(4));0];                      %
        Vector, leg joint to upper leg CoM

    r_CoM = r_SL*m_l/(m_l+m_u) + (r_OG(1:2)+r_GU(1:2))*m_u/(m_l+m_u); %
        Position, Center of Mass system
    r_CoM_h(i) = r_CoM(2);                                           %
        Distance, CoM from ground
    r_UL = r_OG(1:2) + A_IU*[l_G;0];                                 %
        Position, upper leg

    % Calculate positions of contact points
    for j = 1:n
        if j <=n/2
            r_cf(:,j) = r_OF + A_Il*[sin((j-1-n/2)*alpha_i)*R+b_i/2-dff
                ;...
            R*(1-cos((j-1-n/2)*alpha_i));0];
        else
            r_cf(:,j) = r_OF + A_Il*[sin((j-1-n/2)*alpha_i)*R+b_i/2-dff
                ;...
            R*(1-cos((j-1-n/2)*alpha_i));0];
        end
    end

    % Velocity CoM lower leg
    v_l = [u_E(1);u_E(2);0];
    % Velocity, CoM upper leg
    v_u = [u_E(1)-sin(q_E(3))*l(1)*u_E(3)+cos(q_E(3))*l(2)*u_E(3)+l_D*sin
        (q_E(3))*u_E(3)-l_G*sin(q_E(4))*u_E(4);...
           u_E(2)-cos(q_E(3))*l(1)*u_E(3)-sin(q_E(3))*l(2)*u_E(3)+l_D*cos
               (q_E(3))*u_E(3)+l_G*cos(q_E(4))*u_E(4);0];     % Velocity
               , CoM upper leg

    vl(i) = norm(v_l);
    vu(i) = norm(v_u);
    vlx(i) = u_E(1);
    vly(i) = u_E(2);
    E_pot_l = m_l*g*q_E(2);                                          %
        Potential energy, lower leg
    E_kin_l = 1/2*m_l*v_l'*v_l + 1/2*J_l*u_E(3)^2;                   %
        Kinetic energy, lower leg
    E_pot_u = m_u*g*r_UL(2);                                         %
        Potential energy, upper leg
    E_kin_u = 1/2*m_u*v_u'*v_u + 1/2*J_u*u_E(4)^2;                   %
        Kinetic energy, upper leg
```

```
E_spring = 1/2*k*(l_F-l_0)^2;                                    %
    Potential energy, spring

E_sys(i,:) = [E_pot_l,E_kin_l,E_pot_u,E_kin_u,E_spring];
E_tot(i) = E_pot_l + E_kin_l + E_pot_u...
           + E_kin_u + E_spring - E_0;                           % Total
                Energy, System

E_tot1(i) = E_pot_l + E_kin_l;
E_tot2(i) = E_pot_u + E_kin_u;

%% Actuation power
P_M(i) = abs(T_t(i)*(u_E(3)+u_E(4)));                            % Power
     of the free moment
if i >1
    W_M(i) = W_M(i-1)+P_M(i)*dt;                                 % Work
        performed of free moment
else
    W_M(1) = 0;
end
%% Velocity angle of center of mass
alpha(i) = atan((v_l(2)*m_l/(m_l+m_u)+v_u(2)*m_u/(m_l+m_u))/((v_l(1)*
    m_l/(m_l+m_u)+v_u(1)*m_u/(m_l+m_u))))*180/pi;

%% Ground contact check
if index ~= 0       % Ground contact active
    gcc(k_gcc) = 1;
    t_gcc(k_gcc) = t_A(i);
    i_gcc(k_gcc) = i;
    x_gcc(k_gcc) = q_A(1,i);
    k_gcc = k_gcc + 1;
else                % Ground contact not active
    t_gcc_l(k_gcc_l) = t_A(i);
    i_gcc_l(k_gcc_l) = i;
    if k_gcc_l > 2
        % Get rid of non realistic flight phase points
       if i_gcc_l(k_gcc_l-1)+10 < i_gcc_l(k_gcc_l) && i_gcc_l(k_gcc_l
            -2)+10 < i_gcc_l(k_gcc_l-1)
           i_temp = i_gcc_l;
           clear i_gcc_l
           i_gcc_l = [i_temp(1:k_gcc_l-2),i_temp(end)];
           k_gcc_l = k_gcc_l -1;

       end
    end
    x_gcc_l(k_gcc_l) = q_A(1,i);
    k_gcc_l = k_gcc_l + 1;
end

 %% Animation
 % Set new values for plot objects
 if i >= ff;
 set(figure(1),'units','normalized','outerposition',[0 0 1 1])
 figure(1)
 set(h_ll,'XData',[r_OP(1);r_OF(1)],'YData',[r_OP(2);r_OF(2)]);
 set(h_ul,'XData',[r_OG(1);r_OM(1)],'YData',[r_OG(2);r_OM(2)]);
 set(h_f,'XData',[r_cf(1,:)],'YData',[r_cf(2,:)]);
 set(h_s,'XData',[r_OP(1);r_OM(1)],'YData',[r_OP(2);r_OM(2)]);
 set(h_g,'XData',[-1,ww],'YData',[-0.005,-0.005]);

 % Set Axis
 axis equal                 % Equal x-y ratio
 axis([-0.1,ww,-0.1,wh])    % Set window properties

 F(ii) = getframe;          % Save picture of the current plot
 ii = ii + 1;               % Set next iteration variable getframe
 ff = ff + N/kk;            % Next iteration to be taken a picture
 end

 %Check if robot fell
 if q_E(2)<=0
     disp('%%%%%%%%%%%%%%%%%%%%    Robot Fall
          %%%%%%%%%%%%%%%%%%%%%%%')
     fall = 1;
```

```
            break
        end
end

t_sim = toc        % Simulation time

%% Save avi file if necessary
%movie2avi(F,'C-Shaped Hopper','compression','None','fps',kk/(N*dt))

%% Cost of transport
if fall < 1
    E_exp = W_M(N) + E_tot(1);                        % Total energy spent
    CoT_c = E_exp/((m_u+m_l)*g*(q_A(1,N)-q_A(1,1)));   % Cost of Transport

else
    E_exp = W_M(i) + E_tot(1);
end

 %% Find jumping and landing angles
    ka = 1;
    ka_l = 1;
    for j = 1:k_gcc-2
        % Save jumping angle
        % check if next iteration will change from ground to flight phase
        if i_gcc(j+1)-i_gcc(j) > 10
            if alpha(i_gcc(j))>0
                alpha_gcc(ka) = alpha(i_gcc(j));    % Jumping angle CoM
                alpha_foot(ka) = (q_A(3,i_gcc(j))-phi_l_p)*180/pi;
                t_ka(ka) = t_gcc(j);                % Time at takeoff
                i_ka(ka) = i_gcc(j);                % Iteration at takeoff
                x_ka(ka) = x_gcc(j);                % Position of takeoff
                ka = ka +1;
            end
        end
    end
    for hh = 1:k_gcc_l-2
        % Save landing angle
        % check if next iteration will change from flight to ground phase
        if i_gcc_l(hh+1)-i_gcc_l(hh) > 10
            if alpha(i_gcc_l(hh))<0
                alpha_gcc_l(ka_l) = alpha(i_gcc_l(hh)); % Landing angle CoM
                alpha_foot_l(ka_l) = (q_A(3,i_gcc_l(hh))-phi_l_p)*180/pi;
                t_ka_l(ka_l) = t_gcc_l(hh);         % Time at landing
                i_ka_l(ka_l) = i_gcc_l(hh);         % Iteration at landing
                x_ka_l(ka_l) = x_gcc_l(hh);         % Position of landing
                ka_l = ka_l + 1;
            end
        end
    end
```

C.3 IndX.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Robot Leg with Curved Foot  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Fabio Giardina 2013         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Index Allocation File       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This function assigns index numbers to possible active contacts. This is
%needed, since the number of active contacts defines the dimension of the
%linear complementarity problem.

function [i_1,i_2,index,W_N,W_T,check_v] = indX(w_N,w_T,g_N,n)

    i_1 = 0;                                    %Active contacts index
        initially zero
    i_2 = 0;
    index = 0;
    check_v = zeros(n,1);

    %Check how many contacts are closed
    for j = 1:n
```

```
        if g_N(j) <= 0
            check_v(j) = 1;
        end
    end

    %Check which contact is closed and if the subsequent contact is closed
    %as well. (It is assumed that only two contacts can be closed at the
    %same time. If more than two contacts should be closed, a warning
    %message is being displayed in the command window)
    for j = 1:n
        if g_N(j) <= 0                          %Case contact j is active
            W_N(:,1) = w_N(:,j);                %Assign normal force direction
                 vector
            W_T(:,1) = w_T(:,j);                %Assign normal force direction
                 vector
            i_1 = j;
            if j <n
                if g_N(j+1) <= 0                %Case contacts j and j+1 are
                     active
                    W_N(:,2) = w_N(:,j+1);
                    W_T(:,2) = w_T(:,j+1);
                    i_2 = j+1;
                    break;
                end
            end
        end
    end

    %Index-Contact relationship vector
    if i_1 > 0
        index(1,:) = i_1;
        if i_2 > 0
            index(2,:) = i_2;
        end
    end
    if index == 0
        W_N = 0;
        W_T = 0;
    end
```

C.4 LCS.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Robot Leg with Curved Foot  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Fabio Giardina 2013         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% LCP Solver                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This function solves the assigned linear complementarity problem (LCP)
% enumeratively. By altering the entries of the matrix C in a specified way
% (see literature on LCP's), a complementarity vector z is seeked, which
% comprises only elements larger or equal to zero. Note that the
% enumerative method finds all the solution of the LCP(given that a
% solution exists) but lacks in efficiency. However, since only two
% contacts are considered in this script, the performance is still
    appropriate.

function [lambda_N,lambda_R,NS] = LCS(A,b,index)
%Find all combinations of complementary vectors and matrices
        C = [eye(length(A)),-A];
        Ck = C;
        tt = 0;
        z = zeros(length(index)*6,1);
        NS = 0;     % If NS = 1 then, there doesn't exist a solution of the
             LCP

%% Solution for one closed contact
        if length(index) == 1
            for j = 1:2                                         %1
                Ck(1,1) = 2-j;
                if Ck(1,1) == 1
                    Ck(:,4) = zeros(3,1);
```

```
                else
                    Ck(:,4) = C(:,4);
                end
                for j = 1:2                                         %2
                        Ck(2,2) = 2-j;
                        if Ck(2,2) == 1
                            Ck(:,5) = zeros(3,1);
                        else
                            Ck(:,5) = C(:,5);
                        end

                    for j = 1:2                                     %3
                        Ck(3,3) = 2-j;
                        if Ck(3,3) == 1
                            Ck(:,6) = zeros(3,1);
                        else
                            Ck(:,6) = C(:,6);
                        end
                         if rank(Ck) == 3
                            z_c = Ck\b;
                            if z_c >= 0                             %If z has
                                 full rank and only nonnegative entries, a
                                 solution was found.
                                tt = tt+1;
                                z = z_c;
                            end
                        end
                    end
                    Ck(:,6) = C(:,6);
                end
                Ck(:,5) = C(:,5);
            end
%% Solution for two closed contacts
        % If two contacts close at the same time, the number of iteration
        % is increased times 2^3 -> 2^3*2^3 = 2^6
        elseif length(index) == 2
             for j = 1:2                                            %1
                Ck(1,1) = 2-j;
                if Ck(1,1) == 1
                    Ck(:,7) = zeros(6,1);
                else
                    Ck(:,7) = C(:,7);
                end
                for j = 1:2                                         %2
                        Ck(2,2) = 2-j;
                        if Ck(2,2) == 1
                            Ck(:,8) = zeros(6,1);
                        else
                            Ck(:,8) = C(:,8);
                        end

                    for j = 1:2                                     %3
                        Ck(3,3) = 2-j;
                        if Ck(3,3) == 1
                            Ck(:,9) = zeros(6,1);
                        else
                            Ck(:,9) = C(:,9);
                        end
                        for j = 1:2                                 %4
                            Ck(4,4) = 2-j;
                            if Ck(4,4) == 1
                                Ck(:,10) = zeros(6,1);
                            else
                                Ck(:,10) = C(:,10);
                            end
                            for j = 1:2                             %5
                                    Ck(5,5) = 2-j;
                                    if Ck(5,5) == 1
                                        Ck(:,11) = zeros(6,1);
                                    else
                                        Ck(:,11) = C(:,11);
                                    end

                            for j = 1:2                             %6
```

```
                                Ck(6,6) = 2-j;
                                    if Ck(6,6) == 1
                                        Ck(:,12) = zeros(6,1);
                                    else
                                        Ck(:,12) = C(:,12);
                                    end
                                        if rank(Ck) == 6
                                            z_c = Ck\b;
                                            if z_c >= 0         %If z has
                                                 full rank and only
                                                 nonnegative entries, a
                                                 solution was found.
                                                tt = tt+1;
                                                z = z_c;
                                            end
                                        end
                                    end
                                Ck(:,12) = C(:,12);
                            end
                            Ck(:,11) = C(:,11);
                        end
                        Ck(:,10) = C(:,10);
                    end
                    Ck(:,9) = C(:,9);
                end
                Ck(:,8) = C(:,8);
             end
        end

        %If no solution was found, set NS to 1
        if tt == 0
            NS = 1;
        end

%% Repeat calculation in depth in case no solution was found
        % If no solution was found, look for solutions with deficient
        % C-Matrix ranks (Multiple solutions may occur. Note that the
        % accelerations are still defined even though multiple solutions
        % were found. The impact forces however are not.)
        if NS == 1 && length(index)==1
            %The rank of the C matrix is likely to be deficient. Since it
            %has no effect on the accelerations, the warning message is
            %turned off.
            warning('off','MATLAB:rankDeficientMatrix');
             for j = 1:2                                            %1
                Ck(1,1) = 2-j;
                if Ck(1,1) == 1
                    Ck(:,4) = zeros(3,1);
                else
                    Ck(:,4) = C(:,4);
                end
                for j = 1:2                                         %2
                        Ck(2,2) = 2-j;
                        if Ck(2,2) == 1
                            Ck(:,5) = zeros(3,1);
                        else
                            Ck(:,5) = C(:,5);
                        end

                    for j = 1:2                                     %3
                        Ck(3,3) = 2-j;
                        if Ck(3,3) == 1
                            Ck(:,6) = zeros(3,1);
                        else
                            Ck(:,6) = C(:,6);
                        end
                            z_c = Ck\b;
                            if z_c >= 0                         %If z has
                                 full rank and only nonnegative entries, a
                                 solution was found.
                                tt = tt+1;
                                z = z_c;
                        end
                    end
```

```
            Ck(:,6) = C(:,6);
        end
        Ck(:,5) = C(:,5);
     end
     warning('on','MATLAB:rankDeficientMatrix');
elseif NS == 1 && length(index)==2
    warning('off','MATLAB:rankDeficientMatrix');
    for j = 1:2                                          %1
        Ck(1,1) = 2-j;
        if Ck(1,1) == 1
            Ck(:,7) = zeros(6,1);
        else
            Ck(:,7) = C(:,7);
        end
        for j = 1:2                                      %2
                Ck(2,2) = 2-j;
                if Ck(2,2) == 1
                    Ck(:,8) = zeros(6,1);
                else
                    Ck(:,8) = C(:,8);
                end

            for j = 1:2                                  %3
                Ck(3,3) = 2-j;
                if Ck(3,3) == 1
                    Ck(:,9) = zeros(6,1);
                else
                    Ck(:,9) = C(:,9);
                end
                for j = 1:2                              %4
                    Ck(4,4) = 2-j;
                    if Ck(4,4) == 1
                        Ck(:,10) = zeros(6,1);
                    else
                        Ck(:,10) = C(:,10);
                    end
                    for j = 1:2                          %5
                            Ck(5,5) = 2-j;
                            if Ck(5,5) == 1
                                Ck(:,11) = zeros(6,1);
                            else
                                Ck(:,11) = C(:,11);
                            end

                    for j = 1:2                      %6
                        Ck(6,6) = 2-j;
                            if Ck(6,6) == 1
                                Ck(:,12) = zeros(6,1);
                            else
                                Ck(:,12) = C(:,12);
                            end
                                z_c = Ck\b;
                                   if z_c >= 0
                                        tt = tt+1;
                                        z = z_c;
                                        NS = 0;
                                   end
                            Ck(:,12) = C(:,12);
                    end
                    Ck(:,11) = C(:,11);
                end
                Ck(:,10) = C(:,10);
            end
            Ck(:,9) = C(:,9);
        end
        Ck(:,8) = C(:,8);
     end
    end
    warning('on','MATLAB:rankDeficientMatrix');
end

if tt == 0
    NS = 1;
end
```

```
%% Define solution of the LCP
lambda_N = z(length(index)*3+1:length(index)*3+1*length(index));
lambda_R = z(length(index)*4+1:length(index)*4+1*length(index));
```

Appendix D

Function "Find Optimum"

The presented code offers a simple method of finding the natural frequency of the real robot in an experiment. By assuming that the provided power by the motor is highest at the system's natural frequency, and that the expended energy rises gradually when approaching the natural frequency, this code provides fast convergence. Experimental findings showed that, starting from an arbitrarily bad frequency, the algorithm teaches the real robot to hop withing a few seconds.

```
% Write and Read to a NI USB-6008 DAQ device
clear all
close all
clc

%% Digital input enables ESCON
dio = digitalio('nidaq', 'Dev1');
hline = addline(dio, 0:11, 'out');
putvalue(dio, [8 8 8 8 8 8 8 8 8 8 8 8])

%%  Initialization
ai = analoginput('nidaq', 'Dev1');  % Analog Input
ao = analogoutput('nidaq', 'Dev1'); % Analog Output
ao0 = addchannel(ao, 0);            % Add desired channel for output
                                    % (See also NI SCB 68 PIN positions)
ai0 = addchannel(ai, [0,1]);        % Add desired channel for input
                                    % (See also NI SCB 68 PIN positions)

timelength = 20;        % total motor control time (s)
freq = [1;3;6];         % Initial Signal frequencies (Hz)
A = 2;                  % Current amplitude
r_c = 0.95/10;          % Current to voltage ratio. e.g. 0.95/10 -> 0.95
                        % Ampere if the analog output is 10V

ai_value = [0,0];       % Initialize Input value
ao_array = 1;           % Initialize Output value
time_array = 0;         % Initialize time
sampt = 2;              % Sampling time

tic                     % Start time

%% Start Control loop
disp('Start of loop')
kk = 1;
energy=0;
jj = 1;

while time_array(end)<timelength
        ao_value = A*(sin(2*pi*freq(jj)...
            *time_array(end)));             % Define motor current
        ao_array=[ao_array;ao_value*r_c];   % Write current in  vector
        putsample(ao, ao_value)             % Set current at analog output
        time_array=[time_array;toc];        % Write time in vector
```

```
        tmp=getsample(ai);                  % Get motor winding 1 Voltage
        ai_value = [ai_value;tmp];          % Write motor winding 1 Voltage

        dt = mean(time_array(2:end)-...
                  time_array(1:end-1));     % Average time step
        energy = 2*abs(ao_array(end))*...   % Energy expended
                 abs(ai_value(end))*dt+energy;

        % After sampling time is up, check for frequencies which showed a
        % higher energy expenditure:
        if toc + 2*dt>sampt*kk
           results(kk,:) = [freq(jj),energy];
           if jj == 4
               res_temp(2,:) = [energy,freq(2)];
           else
               res_temp(jj,:) = [energy,freq(jj)];
           end
           energy = 0;
           kk = kk + 1;
           if jj >= 3
               % Check which tested frequency is closer to the natural
                    frequency
               if res_temp(1,1)>res_temp(3,1) &&  res_temp(2,1)>res_temp(3,1)
                   f4 = (freq(1)+freq(2))/2;
                   freq = [freq(1);f4;freq(2);f4];
                   res_temp(3,:) = res_temp(2,:);
               elseif res_temp(3,1)>res_temp(1,1) &&  res_temp(2,1)>res_temp
                    (1,1)
                   f4 = (freq(2)+freq(3))/2;
                   freq = [freq(2);f4;freq(3);f4];
                   res_temp(1,:) = res_temp(2,:);
               end
               jj = 3;
           end
           jj = jj + 1;
        end
end
ao_value = 0;
putsample(ao, ao_value)
disp('End of loop')

%% Disable ESCON
dio = digitalio('nidaq', 'Dev1');
hline = addline(dio, 0:11, 'out');
putvalue(dio, [0 0 0 0 0 0 0 0 0 0 0 0])
delete(ai)
delete(ao)

%% Transform Voltage
R2 = 22.2;          % Voltage divider resistance 2 (kOhm)
R1 = 98;            % Voltage divider resistance 1 (kOhm)
K = R2/(R1+R2);     % Voltage reduction constant
VoltIn= -ai_value(:,1)/K;

%% Plot results
plot(time_array,VoltIn)
hold on
plot(time_array,ao_array,'r')
xlabel('Time [s]')
ylabel('V/A')

%% Calculate expended energy
figure(2)
en_sort = sortrows(results);
plot(en_sort(:,1),en_sort(:,2))
```

Appendix E

Brushless DC Motor Control

The motor used for the generation of the motor torque acting on the robot is a brushless DC motor (BLDC). The BLDC motor consists of a permanent magnet located in the center of the motor, which is rotating. Around the rotor, three windings are arranged and connected in a way, that will allow then to produce a variable rotating magnetic field. The motor windings are static, such that no brush is needed for a switch of current, as it is the case for conventional DC motors. The ability to drive the motor in a desired way, needs special control to generate the driving magnetic field in the motor windings. Figure E.1 shows the set-up of such a control system. A voltage supply guarantees the power source for the motor to run. Six switches, which are managed by the controller, are arranged, such that the current can flow either through winding 1-2, 1-3, 2-3, or also in the opposite direction. This will generate a magnetic field, which will drive the rotating permanent magnet in a desired way. In order to control the motion of the magnet, three hall sensors are measuring the actual position. The current signal of the controller through the motor winding is given by a pulse width modulation signal (PWM). The trajectory of this signal can be seen in figure E.1. A peculiar thing is, that the phase current given by the PWM signal, is always positive for one third of a whole cycle, zero for one third, and negative for one third. Knowing this comes in handy if one is to measure the dropping voltage over the motor.

In the experiment with the robot, a Maxon EC motor was used. Due to its high efficiency and versatility in terms of input signal, this seemed a reasonable choice. Having said that, the control of the motor is rather untransparent. The Maxon EC motor came with a ESCON control module, which generated the PWM signal, as has been discussed already in section 6.2. The current controlled by the ESCON can be requested from the module, and it matches the desired reference signal given from the Matlab desktop PC (see figure 6.2). However, the dropping voltage can not be measured directly from the module, and needs to be assessed by using an external measurement. This can be done by measuring the voltage drop of one motor winding to the ground. After having taken several measurements, the resulting PWM signal could not be interpreted

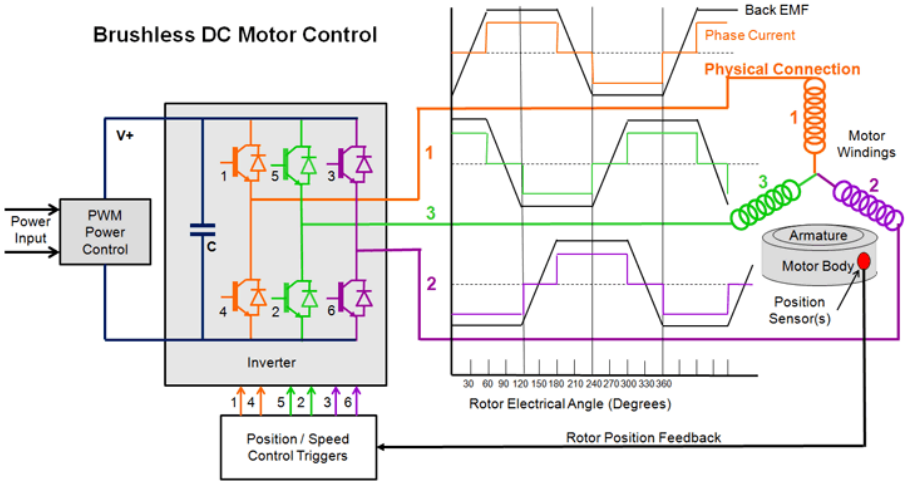
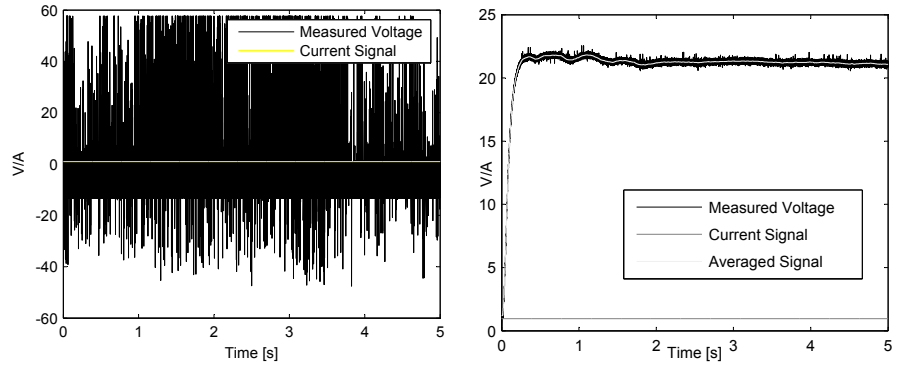


Figure E.1: Illustration of the control of a brushless DC motor (BLDC)[1].

correctly due to its high signal switching (figure E.2).
In order to overcome this problem, a RC filter was designed with a cut-off frequency around 50 kHz, which is the switching frequency of the PWM signal. The elements of the RC filter can be seen in figure 6.3. The resulting check was a nominal voltage measurement of the Maxon motor. As shown in figure E.2, the filtered signal shows a smooth distribution of the voltage. In order to verify the measurement, a nominal voltage experiment was performed. By braking the motor manually while sending the nominal current through the motor windings, the nominal voltage should be reached. The nominal current is given in the motor specification as 0.95A and the voltage as 48V. As can be seen in figure E.2, this is double the voltage measured. The reason is that the voltage drop from only one motor winding to the ground was measured. While the full voltage drops over one third of the PWM period, only half of the voltage is measured over another third, as the winding is not active but still measuring the node current, and no voltage drops over the last third of the period, as the point of the winding measurement is directly attached to the ground. When averaging the measured voltage, this leads to exactly half of the nominal voltage, which corresponds to the measurement.
With this certainty in mind, we are able to compute the used power by the motor as described in section 6.2. A voltage measurement of the motor winding with a sinusoidal current input as a reference can be seen in E.3.

[1]Picture Source: http://www.mpoweruk.com/motorsbrushless.htm, 4.10.2013.



(a) **(b)**

Figure E.2: (**a**) Plot of a unfiltered nominal voltage measurement with the used Maxon EC 45 Flat BLDC motor. Plot (**b**) shows the RC-filtered nominal voltage measurement. The measurement was performed by braking the motor manually while sending the nominal current through the motor windings. The measurement is the voltage drop of one motor winding to the ground.

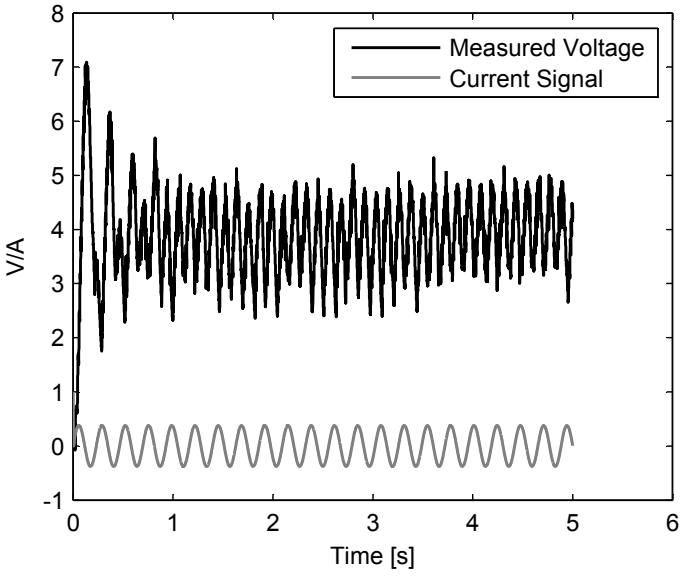


Figure E.3: Plot of voltage measurement with sinusoidal current reference input. Current amplitude is 3.8A and frequency is 4.3Hz.