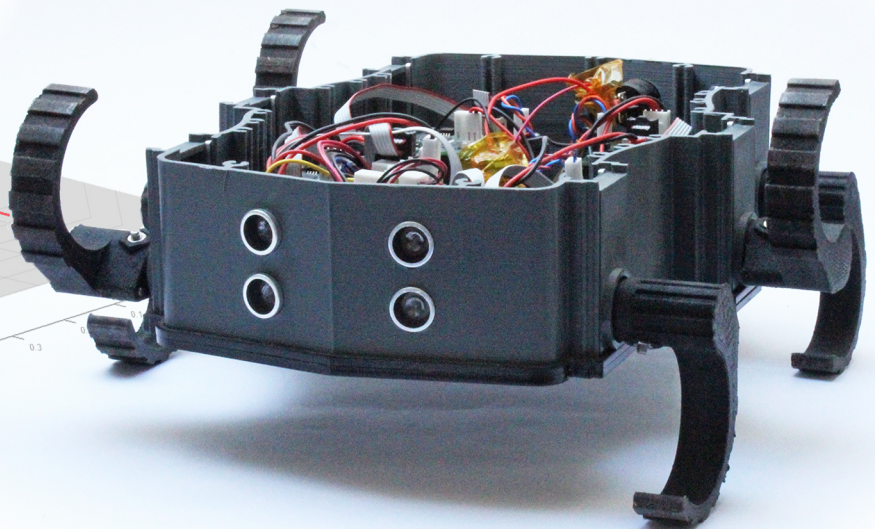
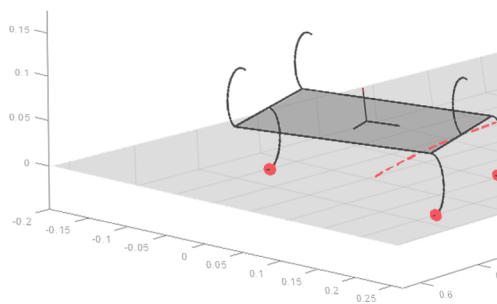
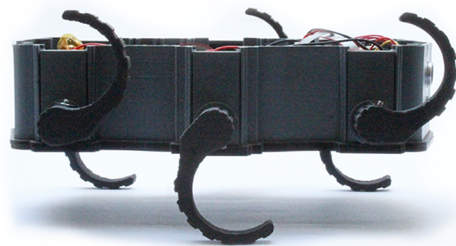


Modelling the hybrid dynamics of a hexapedal robot

Predicting the Zebro's path using an identified leg-ground slippage model

F.P. Erkelens

Master of Science Thesis



Modelling the hybrid dynamics of a hexapedal robot

**Predicting the Zebro's path using an identified leg-ground slippage
model**

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Systems and Control at Delft
University of Technology

F.P. Erkelens

March 13, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of
Technology

Abstract

Legged locomotion is a discrete event system (DES) due to the ever changing contact states of each leg. As such, it requires a nonlinear modelling method to predict the trajectory of a legged robot. One such robot has been focused on throughout this thesis; the six legged “Zebro”. With its half-circular legs, the Zebro is well suited for traversing uneven terrain and even climbing up small steps, making it the robot of choice for a lunar mission in the near future. A previous attempt at modelling the trajectory of a Zebro kinematically fell short when it came to modelling a curved path, with the reasoning behind this being how the Zebro’s legs can visibly be seen slipping over the ground as it walks, the effect of which is never taken into account.

A combination of kinematics and dynamics was used for the model in this thesis. The reason for this is that the Zebro’s legs are actuated using position controlled motors, so the information available is the leg’s angle and the speed at which the leg is rotating, rather than the torque. Therefore, kinematics were used to estimate the vertical motion of the Zebro due to the rotational speed of each leg, which could then be used to estimate the normal contact force on each leg. The traction forces could then be estimated using the normal force and a slippage model which has been experimentally identified in collaboration with a different research team at the TU Delft. With these forces, and the Newton-Euler equations of motion, the Zebro’s path could be predicted.

Applying a slippage model to a half-circular leg, rather than a wheel, required a new method of calculating the slip ratios which was based on Pacejka’s formula, but adapted to also account for the case where a leg is standing on its toe.

Furthermore, a contact detection algorithm was designed to kinematically predict which leg(s) lift off of the ground during a touchdown event. This was used to kinematically model the orientation of the Zebro during a tetrapod gait and was shown to correctly predict the complicated contact transitions during said gait.

Another product of the research in this thesis is a new and improved algorithm for a turning tripod gait, which achieves smoother turning than beforehand by guaranteeing a smooth contact transition between two leg groups. That being said, it cannot yet be applied to the

tetrapod gait, so the current gait scheduling algorithm is still required for a turning tetrapod gait.

The results of the simulations showed the Zebro walking as expected, both in a straight line and when turning, but the simulations could not be validated quantitatively due to current events regarding COVID-19. For a qualitative review of the model, photographs were taken of the Zebro during walking gaits to compare to the simulations and showed that, while straightforward walking was predicted well, the turning circle of the model was significantly sharper than in reality. The reason for this is most likely a problem in the calculation of the slip ratios, since they were consistently unrealistically low, therefore requiring further research in future.

Table of Contents

1	Introduction	1
1-1	Legged Robots	1
1-1-1	Method of modelling	1
1-2	The Zebro	2
1-3	Research objective	3
1-4	Assumptions made	4
1-5	Impact of the research	5
1-6	Outline of the report	5
2	Trajectory Generation	7
2-1	Introduction	7
2-2	Gaits and their parameterisation	7
2-2-1	Gait notation	8
2-3	Gait scheduling	9
2-3-1	Short introduction to max-plus algebra	10
2-3-2	Switching max-plus linear system for straightforward walking	10
2-3-3	Switching max-plus linear system for smooth turning	11
2-4	Kinematic model	12
2-4-1	Results	12
2-5	Challenges	13
2-6	Smoother turning algorithm	14
2-6-1	Calculating the ideal lift-off angle	14
2-6-2	Steering factor	15
2-6-3	Leg speeds	16
2-7	Summary	16

3	Kinematics of the Body	19
3-1	Introduction	19
3-2	Generalised coordinates of the Zebro	19
3-3	Coordinate frames and their transformations	20
3-3-1	Inertial frame	21
3-3-2	Body frame	21
3-3-3	Hip frame	22
3-3-4	Contact frame	23
3-4	Finding the leg's potential contact point	25
3-4-1	Determining the leg's state	26
3-4-2	Calculating $\dot{\phi}_{min}$	27
3-5	Contact state and contact events	28
3-5-1	Contact polygon	28
3-5-2	Detecting contact events	29
3-5-3	Contact transitions in a tetrapod gait	31
3-5-4	Contact transitions in a tripod gait	32
3-5-5	Leg kinematics: hip height and its vertical velocity	33
3-5-6	Body kinematics: vertical velocity of the body and its roll angle	34
3-6	Summary	35
4	Dynamics of the Body	37
4-1	Introduction	37
4-2	Contact forces	38
4-3	Newton-Euler framework	39
4-3-1	Newton's equation	39
4-3-2	Euler's equation	39
4-3-3	General form equations of motion	40
4-4	Weight distribution estimation	41
4-4-1	Distance between a point and a line defined by two points	42
4-4-2	Using an optimisation function to improve the estimate	42
4-5	Slip model	44
4-5-1	Pacejka's magic formula and quantifying the slip of a tire	44
4-5-2	Quantifying the slip of a half-circular leg	47
4-5-3	Model for the lateral traction force	50
4-6	Experimental setup for determining the longitudinal slip curve	51
4-6-1	Results of the experiment	52
4-6-2	Limitations of the experiment	53
4-7	Summary	55

5	Simulations	57
5-1	Introduction	57
5-2	Simulating the max-plus algorithm	57
5-2-1	Modelling the leg speeds	57
5-2-2	Simulating straightforward walking	58
5-2-3	Comparison with real world experiment	59
5-2-4	Simulating turning	60
5-2-5	Key insights	62
5-3	Simulating the novel turning algorithm	62
5-3-1	Simulating 4 seconds of walking with steering factor 1.2	62
5-3-2	Simulating 40 seconds of walking with steering factor 1.25	64
5-4	Experimental results of novel turning algorithm	65
5-4-1	Steering factor 1.1	65
5-4-2	Steering factor 1.3	67
5-4-3	Steering factor 1.5	69
5-4-4	Discussion of the results	70
5-5	Simulating tetrapod kinematics	73
5-5-1	Initial state	74
5-5-2	Simulation of the tetrapod's orientation during walking	74
5-5-3	Comparison with a real Zebro	76
6	Conclusion and Future Work	79
6-1	Future Work	80
A	MATLAB Code	83
A-1	Parameters	83
A-2	Plot zebro	83
A-3	Animate zebro	85
A-4	Main	85
A-5	Gait initiation	95
A-5-1	Init tripod	95
A-5-2	Init Suriana tripod	95
A-6	Rotation Matrices	96
A-7	Homogeneous Transformations	97
A-8	Dynamics	98
A-8-1	Newton-Euler	98
A-8-2	Newton-Euler for optimisation	98
A-9	Slip	99
A-9-1	Slip ratio	99
A-9-2	Pacejka (longitudinal)	100
A-9-3	Pacejka (lateral)	100

A-10 Kinematics	101
A-10-1 Leg angles	101
A-10-2 Body angles	102
A-10-3 Body height estimates	102
A-10-4 Weight distribution	104
A-11 Contact points	106
A-11-1 Contact position	106
A-11-2 Contact velocities	106
A-11-3 Update contact state	106
A-12 Update leg speeds	107
A-13 Save results	108

Chapter 1

Introduction

1-1 Legged Robots

Half of Earth's surface is inaccessible by wheeled or tracked vehicles [1], making legged locomotion a candidate for traversing broken or unstable terrain. However, one limitation of legged robots arises from their mechanical complexity since there will always be a trade-off between freedom of movement and decreased reliability as the number of actuators increases [2]. That being said, recent developments on the well-known Spot built by Boston Dynamics has thrust legged robots into the limelight and displayed how far the technology has come. However, with this advanced technology comes a large price-tag and the demand for simple and cheap, yet versatile, legged robots will remain.

1-1-1 Method of modelling

In this thesis the path of a legged robot is modelled through kinematically determining vertical accelerations of the body, which are used to estimate normal contact forces between the legs and the ground with inverse dynamics. These normal forces are then used to find the traction between the legs and the ground, making it possible to find the body's linear and rotational accelerations using the Newton-Euler framework for equations of motion.

This particular method of combining kinematics to lead to the dynamics of the system was not found in literature, mainly because dynamic models are usually applied to robots with torque-controlled motors, since the contact forces are a simple function of the motor torque. The robot which is handled in this thesis has position-controlled motors, making the contact forces difficult to estimate without using kinematically determined accelerations.

The robot which this dynamic model is applied to is introduced in the following section.

1-2 The Zebro

The name “Zebro” stems from the Dutch words “Zes Benen Robot” [3], meaning “six legged robot”. The Zebro (Figure 1-1) is an autonomous robot designed on the foundations of RHex [4] (pronounced “Rex”), and is produced by a start-up on the campus of the TU Delft. The robot is mechanically simple [5], consisting of six actuated half circular legs. Despite its simple design it can achieve impressive performance having been able to climb stairs and hills [6] and even walk on two legs [7]. Its robustness and versatility has made the Zebro a prime candidate for space exploration, prompting the launch of a team of students at TU Delft hoping to make the first European lunar rover by operating a Zebro on the moon [8].

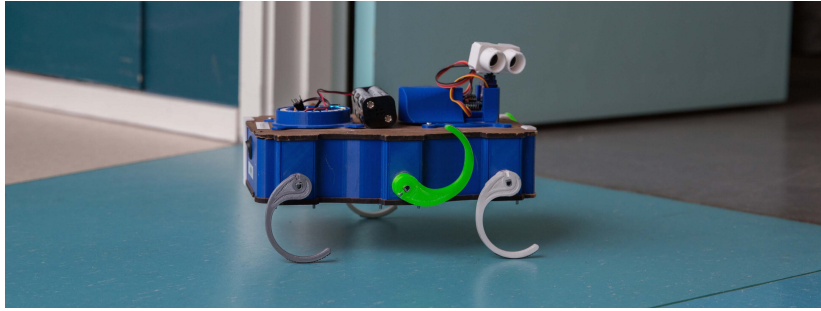


Figure 1-1: The Zebro robot. From [9].

With just one actuator per leg, the Zebro has the benefits of the mechanical simplicity of a six-wheeled vehicle as well as the freedom of movement afforded by legged locomotion.

See Figure 1-2 for some clarification on the nomenclature of the robot. The Zebro consists of a head and a tail with the head facing the forward direction of walking. The six legs are numbered in the manner shown in the figure. Each leg has a “hip” at the point of attachment to the body and a toe at the other end. The motors inside the Zebro’s body rotate the legs to an angle θ_i where the subscript $i \in \{1, 2, 3, 4, 5, 6\}$ represents the leg number. The angle θ_2 is shown as an example in the figure below. The four parameters shown in Figure 1-2 are l , the length; w , the width; h , the height of the robot; and r , the radius of a leg. The final important parameter to note is the mass of the body, m_b , which is not shown in the figure. Two notable angles which will be mentioned often in this thesis are the *touchdown* and *lift-off* angles. The former is the angle at which a leg collides with the ground and the latter is the angle at which the leg lifts up from the ground. They are referred to as θ_{TD} and θ_{LO} respectively.

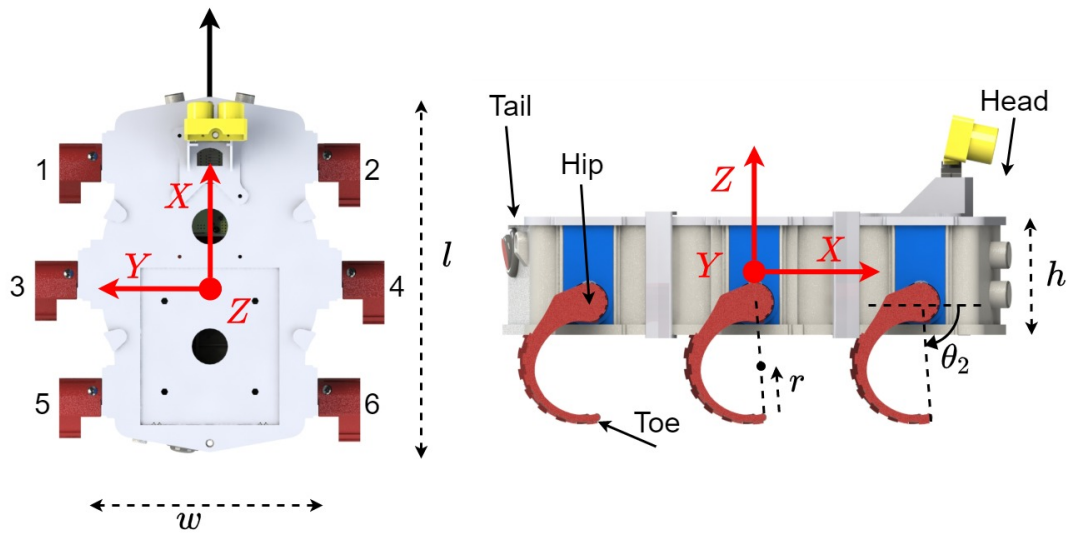


Figure 1-2: On the left: the leg numbering convention, shown from a bird's eye view perspective of the Zebro. On the right: A side view of the Zebro, showing where a leg's hip and toe are located, as well as the leg angle θ for leg 2.

1-3 Research objective

The motivation for research on the Zebro's dynamics stems from the desire to improve the control software governing how the Zebro walks. In its current state, the Zebro can only walk in a straight line. To change direction it must first stop walking, turn on-the-spot, and then continue walking in a straight line. This might be the shortest path from one position to another, but it is not the fastest. Instead, a smooth, curved path would be advantageous for travelling between two positions, because the Zebro would no longer be forced to stop walking.

In recent years, a novel method of designing a gait (walking pattern) has allowed for smooth turning to be implemented in the Zebro [10]. This technique for designing gaits has the potential to greatly improve the Zebro's path planning algorithm, but has not yet been implemented in the current version of the Zebro's software due to its complexity and the lack of understanding within the Zebro team.

One of the problems holding back the development of new path-planning algorithms using this gait-design technique is the lack of an accurate model of the Zebro's motion. An MPC controller, for example, requires a good model in order to predict the Zebro's path and thus control it. Only kinematic models have been used to model the robot [10], [11], but these were deemed insufficient. A portion of the inaccuracy was chalked up to the slip between the Zebro's leg and the ground, which is clearly visible by eye when watching the Zebro walk. As quoted from Suriana in [10]: "The slipping and skidding of the virtual Zebro are not incorporated in the switching kinematic algorithm". A kinematic model uses the velocities of the legs to estimate the motion, but if a portion of this velocity is "lost" through slippage, a kinematic model will never sufficiently predict the Zebro's path.

All in all, the research objective is the following.

To replace the kinematic model with a dynamic model which incorporates the dynamic effects of the leg-ground slippage into the Zebro's equations of motion.

To achieve this, the following three sub-questions must be answered.

1. What are the equations of motion linking the forces on the body to its dynamics?
2. How do the equations of motion change depending on which legs are touching the ground?
3. How much do the legs slip and what effect does this have on the forces on the body?

1-4 Assumptions made

Some assumptions have been made throughout this thesis in order to simplify the research and make it realistically possible.

- The ground is assumed to be completely flat and without obstacles.
- The half-circular legs are assumed to be a perfect semicircular shape.
- Contact between a leg and a ground is assumed to be a point contact.
- The legs are completely rigid.
- The leg angle θ_i can be measured; that is to say, each leg angle is always known.

Using the coordinate frame shown in Figure 1-2, the position of the Zebro's hips can be seen from above as a hexagon with the following coordinates:

$$\text{Leg 1: } \begin{bmatrix} l/2 \\ w/2 \\ 0 \end{bmatrix}, \quad \text{Leg 2: } \begin{bmatrix} l/2 \\ -w/2 \\ 0 \end{bmatrix} \quad (1-1)$$

$$\text{Leg 3: } \begin{bmatrix} 0 \\ w/2 + a \\ 0 \end{bmatrix}, \quad \text{Leg 4: } \begin{bmatrix} 0 \\ -w/2 - a \\ 0 \end{bmatrix} \quad (1-2)$$

$$\text{Leg 5: } \begin{bmatrix} -l/2 \\ w/2 \\ 0 \end{bmatrix}, \quad \text{Leg 6: } \begin{bmatrix} -l/2 \\ -w/2 \\ 0 \end{bmatrix} \quad (1-3)$$

with the parameter a denoting how legs 3 and 4 are further from the x -axis than legs 1,2,5 and 6. That being said, for the duration of this thesis the Zebro is simplified to a uniform cuboid, such that $a = 0$. Due to the assumed uniformity and the symmetrical shape of the body, the robot's Centre of Mass (CoM) is assumed to act at the centre of the cuboid, at coordinates $[0, 0, 0]^T$.

Put simply, the Zebro is assumed to be a uniform cuboid with dimensions w , l , h and mass m_b .

1-5 Impact of the research

A better model for the leg-ground interaction on a flat surface could pave the way for a future model which takes uneven ground into account. Non-flat surfaces make it difficult to predict when a leg will touch the ground because there could be a bump in the ground which will be touched by the leg earlier than expected. A model which can take these bumps into account will be required on terrain such as the moon's surface and the flat-ground model in this thesis could provide a foundation for this.

Furthermore, considering how one of the use-cases of the Zebro is in disaster-struck areas, any improvements in speed through better path-planning could result in faster search and rescue missions, the effect of which could save lives. Better path-planning requires an improved model.

1-6 Outline of the report

First, in Chapter 2 the current method of trajectory generation is discussed. This includes the generation of the leg's rotational velocity reference, as well as the generation of the robot's path. An important development with the Zebro is the "Switching Max-Plus-Linear (SMPL) system" which is used to make the Zebro walk in a smooth circle. Furthermore, a new method for smooth turning is introduced which is easy to model and requires no switching max-plus algorithm to implement.

In Chapter 3, the geometry of the Zebro is presented in the form of coordinate frames and transformations between them. Some important features such as a leg's "potential contact point" are introduced and two contact detection algorithms are described.

Next, in Chapter 4, the Newton-Euler equations of motion for the system are found, as well as a method of estimating the force exerted on a leg by the ground. Using this force estimate, the traction forces which pull the Zebro forward can be calculated and the accelerations are then found with the Newton-Euler equations. Furthermore, the slippage between a leg and the ground is analysed using a method analogous to Pacejka's magic formula, for which a physical experiment was designed and performed to identify its parameters.

After this, in Chapter 5, simulations of a walking Zebro using the new dynamic model are shown. Furthermore, the new smooth turning algorithm is also simulated with the same dynamic model. In order to complete this research fully, validation of this model was supposed to be performed, however due to the current situation surrounding COVID-19, these experiments could not go ahead unfortunately.

Finally, conclusions are made and some future work is recommended in Chapter 6.

Trajectory Generation

2-1 Introduction

The term “trajectory” here refers to two different things: first, the angle reference for a leg’s motor to follow, and second, the path and speed of the Zebro when walking.

The research in this thesis builds upon work done over the past several years, some of which will be mentioned in this chapter. First, in Section 2-2, it is explained what a gait is and how it can be defined. A recent development in the research on the Zebro is the introduction of a novel gait scheduling algorithm which is used to make the Zebro walk in a curved trajectory, as mentioned in Section 1-3. This new gait scheduler is outlined, albeit not in too much detail since no further work will be done on this scheduling algorithm or the mathematics behind it. Next, the results of a kinematic model used in previous research is shown in Section 2-4 followed by a short explanation of the biggest problems faced in the model. A solution to this problem is offered in Section 2-6, which provides a novel turning algorithm which is significantly easier to model.

2-2 Gaits and their parameterisation

Gaits were briefly mentioned in the introduction as a “walking pattern”. As Hildebrand explained “A gait is a manner of moving the legs when walking or running” [12]. For example, there is a clear difference between horse’s trotting gait and galloping gait and these are just two examples. Hildebrand showed a diagram for 16 “of the many” gait sequences [12], implying many more possibilities, but for a six legged system this will be even more.

A clear way of displaying a gait graphically is with a Gantt chart, as shown in Figure 2-1 [13]. The Gantt chart shows grey bars which represent a leg touching the ground and white bars which represent a leg in the air. The gait displayed in the figure is called the “tripod” and will be the main focus of this thesis. The tripod gait is characterised by the two outer legs

on one side moving synchronously with the middle leg of the other side, always maintaining three contact points with the ground in the shape of a triangle.

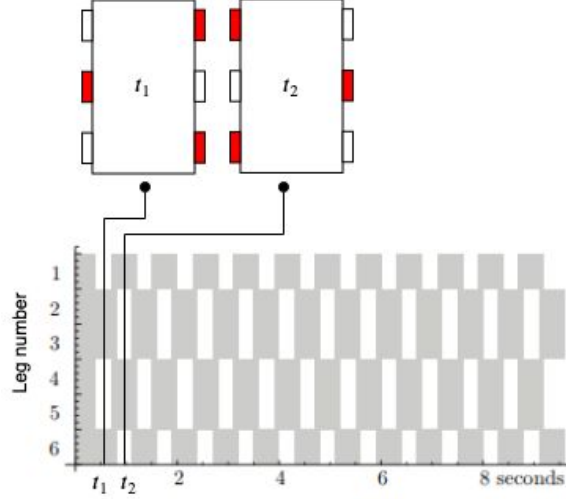


Figure 2-1: Gantt chart representation of a tripod gait, adapted from [13]. Solid bars represent a leg touching the ground and white bars represent a leg in the air. The time markers t_1 and t_2 correspond to the Zebra diagrams above with the same label in which the red legs are the legs in contact with the ground at that time instant.

2-2-1 Gait notation

The gait of an n legged system has been described by Lopes et al. in [13] using a notation showing groups of legs preceding each other. In general,

$$\{L_{1,1}, \dots, L_{1,j}\} \prec \dots \prec \{L_{r,1}, \dots, L_{r,p}\} \quad (2-1)$$

where leg $L_{a,b} \in \{1, 2, 3, 4, 5, 6\}$. All legs in the same group $\{L_{s,1}, \dots, L_{s,j}\}$ rotate synchronously and precede the groups to the right in the notation by only being allowed to lift-off once the succeeding group has touched down.

This notation can be applied to the aforementioned tripod gait, like so,

$$\{1, 4, 5\} \prec \{2, 3, 6\} \quad (2-2)$$

implying that the legs 1,4 and 5 only lift-off from the ground once legs 2,3 and 6 touch the ground, which is in line with what is shown in the Gantt diagram in Figure 2-1. A different gait, named the tetrapod uses three different leg groups, like so:

$$\{1, 4\} \prec \{2, 5\} \prec \{3, 6\}. \quad (2-3)$$

Since a gait is a repeating pattern, it does not matter which leg group is the first in the sequence, provided order does not change. In other words, $\{1, 4\} \prec \{2, 5\} \prec \{3, 6\}$ is the same gait as $\{2, 5\} \prec \{3, 6\} \prec \{1, 4\}$.

2-3 Gait scheduling

A gait scheduler produces a reference path for the robot's legs to follow. An existing method which was used in the software for the RHex is the Buehler clock [14]. In this clock, the leg's phase reference at a specific time is modelled as a continuous piecewise linear function [10] as seen in Figure 2-2.

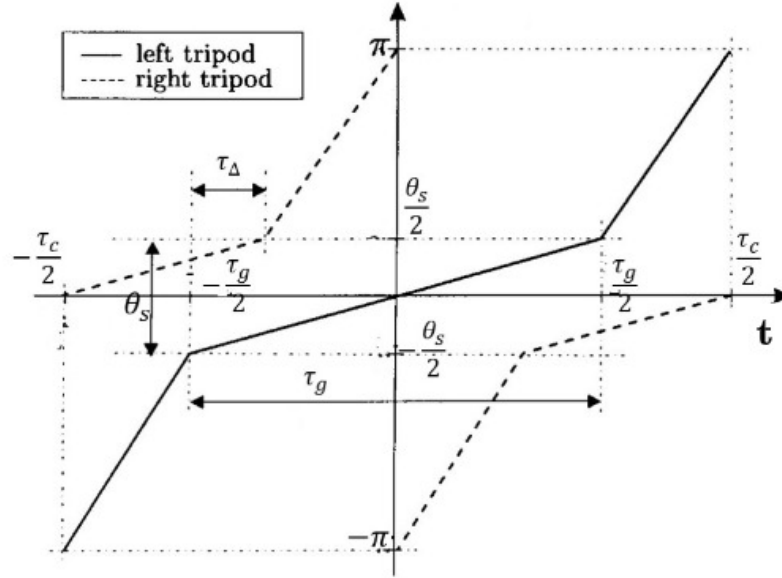


Figure 2-2: Buehler Clock for the tripod gait. From [10].

In the figure, time constant τ_g represents the time spent on the ground and τ_c represents the cycle time, after which the pattern repeats. The angle θ_s represents the stance angle, which in this thesis can be translated to $\theta_s = \theta_{LO} - \theta_{TD}$, and is therefore the angle covered by the leg whilst it is touching the ground. Let

$$\bar{t} = (t + \tau_c/2) \pmod{\tau_c} - \tau_c/2 \quad (2-4)$$

ensuring that $-\tau_c/2 < \bar{t} \leq \tau_c/2$. Now, the mathematical model of the Buehler clock can be written as follows.

$$\theta_{\text{left tripod}}(t) = \begin{cases} \frac{\theta_s}{\tau_g} \bar{t} & \text{if } -\frac{\tau_g}{2} < \bar{t} < \frac{\tau_g}{2} \\ \frac{\pi - \theta_s}{\tau_c - \tau_g} (\bar{t} - \frac{\tau_g}{2}) + \frac{\theta_s}{2} & \text{if } \bar{t} \geq \frac{\tau_g}{2} \\ \frac{\pi - \theta_s}{\tau_c - \tau_g} (\bar{t} + \frac{\tau_g}{2}) - \frac{\theta_s}{2} & \text{if } \bar{t} \leq -\frac{\tau_g}{2} \end{cases} \quad (2-5)$$

$$\theta_{\text{right tripod}}(t) = \theta_{\text{left tripod}}(t + \frac{\tau_c}{2}) \quad (2-6)$$

This clock produces a simple reference trajectory for each leg's angle to follow. However, this clock works for straightforward walking because τ_g is equal on both sides of the robot. Focus in this thesis lies on the turning of a legged robot, so slight alterations must be made. A Switching Max-Plus Linear (SMPL) system is used by Suriana to schedule the times at which each leg should touch the ground and lift-off, allowing for different τ_g per side of the side such that a turning motion can be made.

2-3-1 Short introduction to max-plus algebra

It is possible to describe a certain class of discrete event systems (DES) as a linear system with max-plus algebra. The scheduling of the Zebro's gait falls under this class of system, where each discrete event is the touching down or lifting off of a leg. It has been shown that an n -legged system can be described in a $2n$ -dimensional max-plus linear state space representation [13].

Without delving too deeply into the mathematics, a short introduction on max-plus algebra is appropriate. As the name suggests, the algebraic operations allowed in this framework are max (\oplus) and plus (\otimes), defined as $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$. Let the identity and zero element be $\varepsilon := -\infty$, $e := 0$, respectively and let $\mathbb{R}_{max} = \mathbb{R} \cup \{\varepsilon\}$. Max-plus algebra can now be denoted by $\mathcal{R}_{max} = (\mathbb{R}_{max}, \oplus, \otimes, \varepsilon, e)$ [15].

For matrices, the algebraic functions are defined as

$$[A \oplus B]_{ij} = \max(a_{ij} + b_{ij}) \quad (2-7)$$

and

$$[A \otimes C]_{ij} = \max_{k=1, \dots, m} (a_{ik} + c_{kj}), \quad (2-8)$$

where $A, B \in \mathbb{R}_{max}^{n \times m}$ and $C \in \mathbb{R}_{max}^{m \times p}$.

The $n \times n$ identity matrix is denoted by E_n and has e for each element along the diagonal and ε for all other elements,

$$E_2 = \begin{bmatrix} e & \varepsilon \\ \varepsilon & e \end{bmatrix} \quad (2-9)$$

and $\mathcal{E}_{m \times n}$ is the $m \times n$ max-plus zero matrix and has ε as each element:

$$\mathcal{E}_{2 \times 2} = \begin{bmatrix} \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \end{bmatrix}. \quad (2-10)$$

2-3-2 Switching max-plus linear system for straightforward walking

Let the touchdown event time of leg i be t_i and the lift-off event time of leg i be l_i . Let τ_f be the amount of time a leg spends in flight and τ_g be how long the leg should be on the ground.

$$\begin{aligned} t_i(k) &= l_i(k) + \tau_f \\ l_i(k) &= t_i(k-1) + \tau_g \end{aligned}$$

The following example is taken from [10] and handles the case of a biped for ease of readability, since a biped only has two legs, *left* and *right*. Please note that this example can be expanded to the six-legged case of the Zebro.

Let τ_Δ be the double-stance time, representing the amount of time both legs spend on the ground together. The set of equations that can be used to model the times at which touchdown

and lift-off occur are as follows.

$$\begin{aligned}
t_{left}(k) &= l_{left}(k) + \tau_f \\
&= l_{left}(k) \otimes \tau_f \\
t_{right}(k) &= l_{right}(k) + \tau_f \\
&= l_{right}(k) \otimes \tau_f \\
l_{left}(k) &= \max(t_{left}(k-1) + \tau_g, t_{right}(k-1) + \tau_\Delta) \\
&= t_{left}(k-1) \otimes \tau_g \oplus t_{right}(k-1) \otimes \tau_\Delta \\
l_{right}(k) &= \max(t_{right}(k-1) + \tau_g, t_{left}(k) + \tau_\Delta) \\
&= t_{right}(k-1) \otimes \tau_g \oplus t_{left}(k) \otimes \tau_\Delta
\end{aligned} \tag{2-11}$$

This set of equations can be written in matrix form as a linear system.

$$\begin{bmatrix} t_{left}(k) \\ t_{right}(k) \\ l_{left}(k) \\ l_{right}(k) \end{bmatrix} = \begin{bmatrix} \varepsilon & \varepsilon & \tau_f & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \tau_f \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_\Delta & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} t_{left}(k) \\ t_{right}(k) \\ l_{left}(k) \\ l_{right}(k) \end{bmatrix} \oplus \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_g & \tau_\Delta & \varepsilon & \varepsilon \\ \tau_\Delta & \tau_g & \varepsilon & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} t_{left}(k-1) \\ t_{right}(k-1) \\ l_{left}(k-1) \\ l_{right}(k-1) \end{bmatrix} \tag{2-12}$$

Generalising this for other systems with more legs results in

$$\begin{bmatrix} \mathbf{t}(k) \\ \mathbf{l}(k) \end{bmatrix} = \begin{bmatrix} \mathcal{E} & \tau_f \otimes E \\ P & \mathcal{E} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{t}(k) \\ \mathbf{l}(k) \end{bmatrix} \oplus \begin{bmatrix} \mathcal{E} & \mathcal{E} \\ \tau_g \otimes E \oplus Q & \mathcal{E} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{t}(k-1) \\ \mathbf{l}(k-1) \end{bmatrix} \tag{2-13}$$

where the vectors $\mathbf{t}(k)$ and $\mathbf{l}(k)$ contain the k th touchdown and liftoff times of each leg in the n -legged system.

$$\mathbf{t}(k) = \begin{bmatrix} t_1(k) \\ \vdots \\ t_n(k) \end{bmatrix}, \quad \mathbf{l}(k) = \begin{bmatrix} l_1(k) \\ \vdots \\ l_n(k) \end{bmatrix} \tag{2-14}$$

The matrices P and Q are unique to the chosen gait and are defined as follows, where l_i is a group of legs as shown in Section 2-2-1 and m is the number of unique leg groups in the gait. In the case of a tripod gait, for example, $m = 2$, with $l_1 = \{1, 4, 5\}$ and $l_2 = \{2, 3, 6\}$.

$$\begin{aligned}
[P]_{p,q} &= \begin{cases} \tau_\Delta & \forall j \in \{1, \dots, m-1\}; \forall p \in l_{j+1}; \forall q \in l_j \\ \varepsilon & \text{otherwise} \end{cases} \\
[Q]_{p,q} &= \begin{cases} \tau_\Delta & \forall p \in l_1; \forall q \in l_m \\ \varepsilon & \text{otherwise} \end{cases}
\end{aligned}$$

In summary, the state-space system given in Eq. (2-13) is used to find the times $t_i(k)$ and $l_i(k)$ at which leg i should touchdown or lift-off.

2-3-3 Switching max-plus linear system for smooth turning

The previous section went into the max-plus framework for modelling legged locomotion for a robot walking in a straight line. This section concerns itself with gait scheduling of a robot following a curved path. This work was done by Suriana [10].

A parameter τ_p is introduced and defined as the difference time between the left and right legs. By adding τ_p to the stance time of the “inner” legs, these legs have a longer stance time and thus a slower angular velocity, producing a turning moment and allowing the robot to steer using a technique called “skid steering”[16].

Without going too deep into the details, a similar process is followed as in the previous section, but the different groups of legs are defined by the superscripts i and o signifying *inner* or *outer*. This convention is continued throughout the thesis.

$$\begin{aligned} t_i^i(k) &= l_i^i(k) + \tau_f \\ t_i^o(k) &= l_i^o(k) + \tau_f + \tau_p \\ l_i^i(k) &= t_i^i(k-1) + \tau_g + \tau_p \\ l_i^o(k) &= t_i^o(k-1) + \tau_g \end{aligned}$$

The general form of the max-plus linear state space representation can now be written as follows.

$$\begin{bmatrix} \mathbf{t}(k) \\ \mathbf{l}(k) \end{bmatrix} = \begin{bmatrix} \mathcal{E} & \tau_f \otimes M \\ R & \mathcal{E} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{t}(k) \\ \mathbf{l}(k) \end{bmatrix} \oplus \begin{bmatrix} E & \mathcal{E} \\ (\tau_g \otimes N) \oplus T & E \end{bmatrix} \otimes \begin{bmatrix} \mathbf{t}(k-1) \\ \mathbf{l}(k-1) \end{bmatrix} \quad (2-15)$$

Building the matrices R , M , N and T is done somewhat similarly to P and Q in the previous section, but are significantly more complicated. For the interested reader, the process is described in more detail on pages 22 and 23 of [10].

2-4 Kinematic model

Suriana went on to try and model the Zebro’s path with kinematics as a six-wheeled mobile robot with non-steered wheels. The assumption was made that each leg was essentially a wheel of radius $4r$, resulting in the forward velocity of a hip being $\dot{x}_h = 2r\dot{\theta}$.

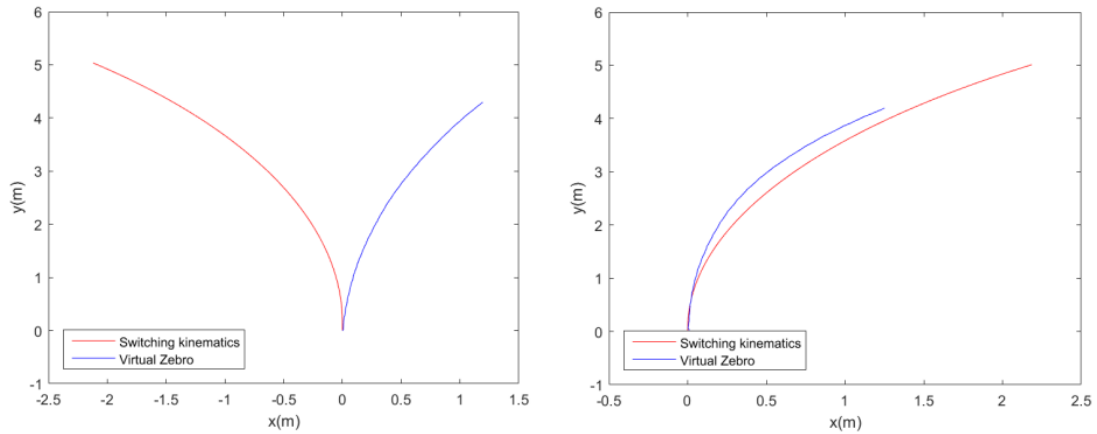
2-4-1 Results

The kinematic model was tested by comparing it to the results of a virtual Zebro in V-REP. As shown in Figure 2-3, with an intended left turn, the results appeared to show the kinematic model correctly predicting a left turn, whereas the virtual Zebro turned to the right. When the intended turning direction was to the right, the kinematic model and the virtual Zebro both appeared to turn in the correct direction.

This result raises the question of whether the virtual Zebro might be incorrectly predicting the path, but Figure 2-4 shows a very similar scenario with a real Zebro. In both cases, the Zebro turns to the right. Three explanations were offered by Suriana.

1. The middle-left leg’s angle was 5° less than the middle-right leg, possibly altering the touchdown and lift-off times and affecting the turning mechanics.
2. The interaction between the leg and the carpet produced an undesired effect.
3. The chosen value for τ_p is not enough to induce turning to the left.

No concrete explanations for the anomalous results were offered in the report. A dynamic model which takes the leg-ground interaction into account would offer more insight into the turning mechanics of the Zebro.



(a) The kinematic model (red) and the virtual Zebro (blue) with an intended left turn.

(b) The kinematic model (red) and the virtual Zebro (blue) with an intended right turn.

Figure 2-3: In (a) the result of an attempted left turn is plotted which shows the virtual Zebro actually turning right. In (b) an attempted right turn is plotted and this time both models do turn right. Both images are taken from [10].



(a) Photographic result of the Zebro's path during an intended left turn.



(b) Photographic result of the Zebro's path during an intended right turn.

Figure 2-4: In (a) one can see the Zebro turn to the right, despite the intended direction being left. In (b) the Zebro does turn to the right, as intended. Note the Zebro starts from the bottle by the white line in the images. Both images are taken from [10].

2-5 Challenges

As Suriana put it “The interaction between the leg and the carpet produced an undesired effect”, which could be interpreted as the leg slipping on the carpet. Slip is a non-linear effect

and is not included in a kinematic model, therefore calling for a dynamic model to accurately take into account the traction forces produced by slippage.

Furthermore, a glaring problem with the current gait scheduling method developed by Suriana is the fact that constant touchdown and lift-off angles are used. Firstly, in a complicated gait such as the tetrapod, the Zebro visibly “wobbles”, resulting in legs touching down at a time which was not scheduled, implying that the touchdown and lift-off angles should actually be a function of the body’s orientation, since pitching forward will significantly affect these angles. In the simple tripod gait the assumption of constant touchdown angles can be valid since the Zebro does not wobble. However, this leads on to a new problem.

The idea that the lift-off and touchdown angle are symmetrical in a tripod gait is simply false. More detail on this topic is gone into in Section 2-6-1. Essentially, the lift-off angle is a physically determined parameter which depends on the distance between a hip and the ground. Suriana’s turning algorithm relies on the presumption that a leg touches the ground at a time t_{TD} due to it being at an angle θ_{TD} , but since this is not necessarily the case in real-life, simulating it would be incorrect.

2-6 Smoother turning algorithm

To combat the aforementioned challenges, a significantly simpler turning algorithm was devised for the tripod gait using carefully chosen touchdown and lift-off *angles*, rather than the max-plus method of calculating touchdown and lift-off *times*. By increasing the difference between the TD and LO angles on one side of the robot compared to the other, a larger distance is travelled by that side, causing the robot to turn. By calculating the LO angles such that the hip height at this angle is exactly equal to the hip height at the TD angle, a perfect tripod rhythm is preserved.

2-6-1 Calculating the ideal lift-off angle

In current Zebro software, and in the previous kinematic model, a lift-off angle was assigned manually. However, this turns out to be a useless parameter to set, since the lift-off angle is a physically determined function of the touchdown angle. The actual lift-off angle is the angle at which the hip height is equal to the hip height at the touchdown angle. The reason this is ideal is because the lift-off angle on one leg will be reached at the exact same time as the touchdown angle on the other.

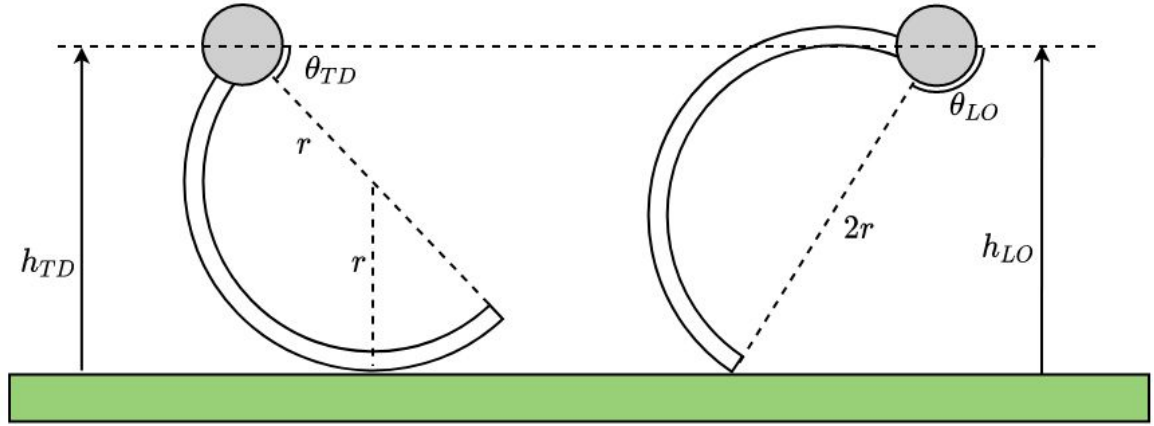


Figure 2-5: Equal hip heights at touchdown and lift-off.

The equations for the heights h_{TD} and h_{LO} shown in Figure 2-5 are given in the following equation.

$$h_{TD} = r + r \sin \theta_{TD}, \quad h_{LO} = 2r \sin \theta_{LO} \quad (2-16)$$

Setting the two heights equal to each other obtains the following expression for θ_{LO} . Note that there are infinite solutions for θ_{LO} , but the solution required is where $\pi/2 \leq \theta_{LO} \leq \pi$, hence the appearance of π and the negative sign in Eq. (2-17).

$$\theta_{LO} = \pi - \arcsin \left(\frac{\sin \theta_{TD} + 1}{2} \right) \quad (2-17)$$

2-6-2 Steering factor

For the Zebro to steer, one side must have a linear velocity larger than the other. Let s be the steering factor such that $v^o = s v^i$, where v^o and v^i are the outer and inner velocities of the Zebro, respectively.

Let θ_{TD}^i and θ_{LO}^i be the touchdown and lift-off angles for the inner side. Also, let θ_{TD}^o and θ_{LO}^o be the touchdown and lift-off angles for the outer side. The difference in velocities of each side is achieved by making the distance travelled by each side different. The distance travelled can be estimated well using the arc length of the circle between θ_{TD} and θ_{LO} .

$$\theta_{LO}^o - \theta_{TD}^o = s(\theta_{LO}^i - \theta_{TD}^i) \quad (2-18)$$

The aim here is to find θ_{LO}^i , θ_{TD}^o and θ_{LO}^o using only a freely chosen angle $-\pi/2 < \theta_{TD}^i < \pi/2$. First, θ_{LO}^i is found using Eq. (2-17). Next, by substituting the expression for θ_{LO} from Eq. (2-17) into Eq. (2-18), a formula for θ_{TD}^o is found.

$$\theta_{TD}^o = \pi - \arcsin \left(\frac{\sin(\theta_{TD}^o) + 1}{2} \right) - s(\theta_{LO}^i - \theta_{TD}^i) \quad (2-19)$$

This non-linear equation can be solved for θ_{TD}^o numerically, since θ_{TD}^i and θ_{LO}^i are already known. Finally, the ideal lift-off angle for θ_{TD}^o can also be found using Eq. (2-17).

With these newfound angles, each side of the Zebro will look somewhat like Figure 2-6 at the touchdown time, where the outer side touches down and lifts off at a lower height than the inner side, allowing for a larger step.

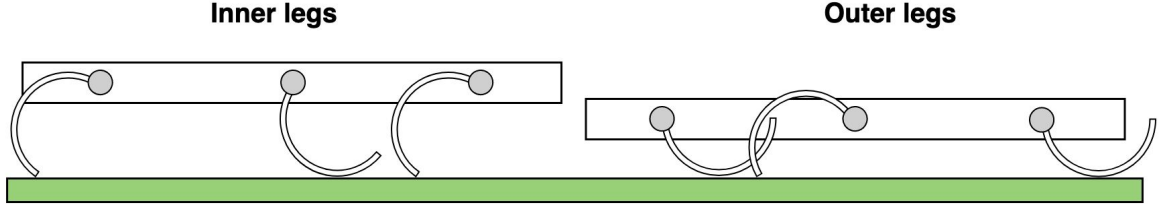


Figure 2-6: Inner and outer side of the Zebro at the touchdown/lift-off event.

2-6-3 Leg speeds

For the legs to maintain the tripod rhythm, the time spent in flight must be equal to the time spent on the ground. Let τ_c be the cycle time (time to complete one full rotation). Now, let $\dot{\theta}_g$ and $\dot{\theta}_f$ be the ground speed and flight speed, respectively. The usual superscripts i and o again represent “inner” and “outer”.

$$\dot{\theta}_g^i = \frac{\theta_{LO}^i - \theta_{TD}^i}{\tau_c/2}, \quad \dot{\theta}_f^i = \frac{2\pi - (\theta_{LO}^i - \theta_{TD}^i)}{\tau_c/2} \quad (2-20)$$

$$\dot{\theta}_g^o = \frac{\theta_{LO}^o - \theta_{TD}^o}{\tau_c/2}, \quad \dot{\theta}_f^o = \frac{2\pi - (\theta_{LO}^o - \theta_{TD}^o)}{\tau_c/2} \quad (2-21)$$

It follows that

$$\begin{cases} \dot{\theta}_j = \dot{\theta}_g & \text{if } \theta_{TD} < \theta_j(\text{mod } 2\pi) \leq \theta_{LO} \\ \dot{\theta}_j = \dot{\theta}_f & \text{else} \end{cases} \quad (2-22)$$

where the appropriate speed is chosen depending on whether leg j is on the inner or outer side of the robot.

These leg speeds are in fact the same as the Buehler Clock described earlier, resulting in the same piecewise linear function for the leg angles, the only difference being that the left and right tripod differ, unlike in Figure 2-2.

2-7 Summary

To summarise, the notation of a gait is introduced and it is shown how a Buehler Clock is used to schedule the trajectory of a leg such that it reaches a certain angle at a certain time. Suriana’s SMPL system for generating the times at which each leg should touchdown and lift-off in order for the Zebro to make a turning motion is touched upon, although the reader is encouraged to find more information on pages 22 and 23 of [10]. The Zebro’s trajectory generated by this gait was modelled kinematically by Suriana, but the results in both a virtual

Zebro and a real-world Zebro did not appear to match with the predicted trajectory from the model.

The possible reasons for the ill performance can be summarised as:

1. Slip is not taken into account in a kinematic model
2. Wobbling of the body makes the prescribed touchdown and lift-off times unreliable
3. The lift-off angles should not be used as an input to the gait scheduling system, since it is a physically determined parameter dependent on the touchdown angle and the body's orientation.

Point 1 is a very large topic which will be discussed extensively throughout the thesis. Points 2 and 3 are addressed with a new turning algorithm designed for the tripod gait which uses variable touchdown angles for each side of the Zebro to allow for steps of different length on either side, causing the robot to turn. By using the body's orientation and leg angles it can be predicted which legs touch the ground, alleviating the problem of the body wobbling. Point 1 is a very large topic which is discussed extensively throughout the thesis.

Kinematics of the Body

3-1 Introduction

This chapter goes through the process of defining and calculating some important kinematic aspects of the system, step by step, starting with the building blocks of any system: its generalised coordinates in Section 3-2. Next, the coordinate frames that were used in the system are introduced in Section 3-3, followed by a section calculating ϕ_{min} , a defining parameter of a leg, and explaining how it is used to determine the leg's state. Next, in Section 3-5, contact events are introduced and the discrete changes in the contact state are modelled in two ways: a contact detection algorithm in Section 3-5-2 designed for complicated gaits in which a lot of pitching and rolling occurs such as the tetrapod; and a hybrid automaton governing prescribed discrete changes in the system which are triggered by certain conditions designed for a simple gait such as the tripod whose state changes are easily predictable.

3-2 Generalised coordinates of the Zebro

The generalised coordinate values must uniquely define any possible position of the system relative to the initial position [17]. The minimum number of generalised coordinates required to specify the position of the system is equal to the number of degrees of freedom of the system. In the case of the Zebro this is 12: three positional coordinates in Cartesian space, three rotations of the body (roll, pitch and yaw) and an angle for each of the six legs.

Let the positional coordinates be

$$x_b, \quad y_b, \quad z_b$$

and the rotational coordinates be

$$\theta_r, \quad \theta_p, \quad \theta_y,$$

where the subscripts represent “body”, “roll”, “pitch” and “yaw” respectively. Finally, as mentioned in Chapter 1, let the angle of leg i , where $i \in \{1, 2, 3, 4, 5, 6\}$, be

$$\theta_i.$$

The generalised coordinate vector \mathbf{q} of the system is therefore

$$\mathbf{q} = [x_b \ y_b \ z_b \ \theta_r \ \theta_p \ \theta_y \ \theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6]^T. \quad (3-1)$$

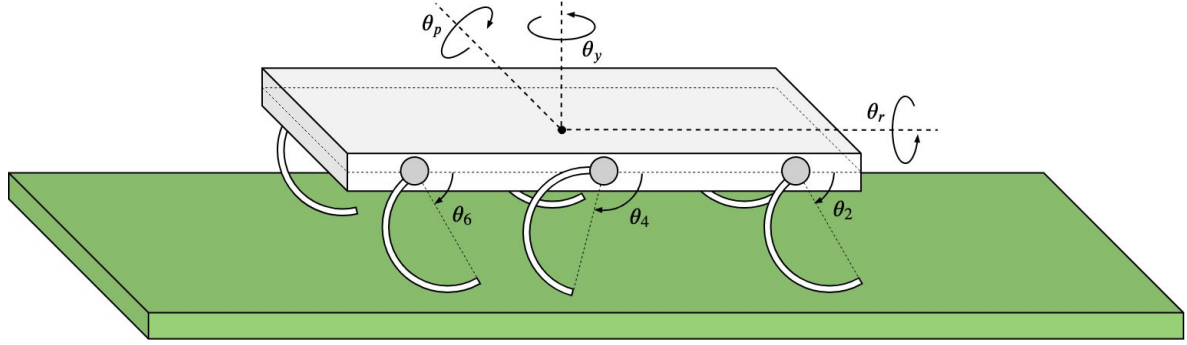


Figure 3-1: A diagram of the Zebro in which the body's rotations θ_r, θ_p and θ_y as well as the leg angles θ_2, θ_4 and θ_6 are shown.

3-3 Coordinate frames and their transformations

Expressing points and vectors in different coordinate frames can be useful when modelling a dynamic system. To transform a point or vector from one frame to another, a homogeneous transformation, consisting of a rotation and a translation, is used.

The set of rotation matrices is also called the special orthogonal group $SO(3)$, formally defined as the set of 3×3 matrices R that satisfy $R^T R = I$ and $\det R = 1$ [18].

A homogeneous transformation matrix H is part of the special Euclidian group $SE(3)$, also known as the group of rigid-body motions in \mathbb{R}^3 , which is the set of all 4×4 real matrices of the form

$$H = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3-2)$$

where $R \in SO(3)$ is the rotation matrix between the two frames and $\mathbf{p} \in \mathbb{R}^3$ is a column vector which describes the translation between the frames. As for notation, let ${}^a\mathbf{p}$ be a point \mathbf{p} expressed in coordinate frame a . Let

$${}^a\mathbf{P} = \begin{bmatrix} {}^a\mathbf{p} \\ 1 \end{bmatrix} \quad (3-3)$$

The general homogeneous transformation from frame b to a , such that ${}^a\mathbf{P} = H_b^{ab}\mathbf{P}$, is given in Eq. (3-4).

$$H_b^a = \begin{bmatrix} R_b^a & {}^a\mathbf{p}_b \\ \mathbf{0}_3^T & 1 \end{bmatrix} \quad (3-4)$$

Where R_b^a is the rotation matrix from frame b to a and ${}^a\mathbf{p}_b$ is the vector from the origin in frame a to the origin in frame b , expressed in frame a .

Transformations can be linked such that if there is a transformation H_b^a from b to a and another transformation H_c^b from c to b , then a point ${}^c\mathbf{P}$ can be expressed in a

$${}^a\mathbf{P} = H_b^a H_c^b {}^c\mathbf{P} \quad (3-5)$$

so $H_c^a = H_b^a H_c^b$. The aim with these transformation matrices is to easily express a point on the Zebro's leg in the inertial frame. This can be done by linking several transformations such as in Eq. (3-5). The links between each frame are shown in the next few sections.

3-3-1 Inertial frame

The inertial coordinate frame I is a stationary coordinate frame attached to the ground. The position of the robot's body $\begin{bmatrix} x_b & y_b & z_b \end{bmatrix}^T$ is always expressed in the inertial frame.

3-3-2 Body frame

The shape of the body has been simplified to just a cuboid. Let the Body-fixed frame B be fixed in the centre of this cuboid. The ${}^B X$ -axis points in the direction of the Zebro's head and the ${}^B Y$ -axis points to its left side. Figure 3-2 shows a diagram of the robot with the body frame attached. The coordinates of this frame in the inertial frame is the position of the body and the rotation required is a rotation of θ_r around the ${}^B X$ -axis, θ_p around the ${}^B Y$ -axis and θ_y around the ${}^B Z$ -axis. Each of these rotations will be referred to as R_r , R_p and R_y respectively.

$${}^I\mathbf{p}_B = \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \quad (3-6)$$

$$R_B^I = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_r & -\sin \theta_r \\ 0 & \sin \theta_r & \cos \theta_r \end{bmatrix}}_{R_r} \underbrace{\begin{bmatrix} \cos \theta_p & 0 & \sin \theta_p \\ 0 & 1 & 0 \\ -\sin \theta_p & 0 & \cos \theta_p \end{bmatrix}}_{R_p} \underbrace{\begin{bmatrix} \cos \theta_y & -\sin \theta_y & 0 \\ \sin \theta_y & \cos \theta_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_y} \quad (3-7)$$

Eq. (3-6) and Eq. (3-7) are used to give the homogeneous transformation to transform a point from the body frame to a point in the inertial frame.

$$H_B^I = \begin{bmatrix} R_r R_p R_y & \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & 1 \end{bmatrix} \quad (3-8)$$

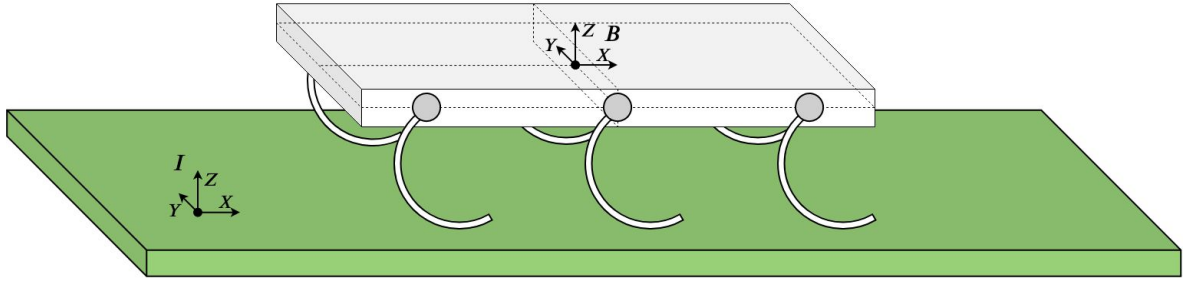


Figure 3-2: The Zebra's body-fixed frame B is shown in the centre of the body, at the position $[x_b, y_b, z_b]$ relative to the inertial frame.

3-3-3 Hip frame

As explained in Section 1-2, the Zebra's hip is a point on the body at which a leg is attached. Each hip frame is located at the hip and rotates with the leg such that the $^H X$ -axis always points towards the i th leg's toe. To find a homogeneous transformation from the B frame to the H_i frame, a rotation of θ_i around the $^H Y$ -axis and a translation to $^B \mathbf{p}_{H_i}$ is required.

$$^B \mathbf{p}_{H_i} = \begin{bmatrix} x_{h_i} \\ y_{h_i} \\ z_{h_i} \end{bmatrix} \quad (3-9)$$

The values of x_{h_i} , y_{h_i} and z_{h_i} are essentially just the coordinates given in Section 1-2.

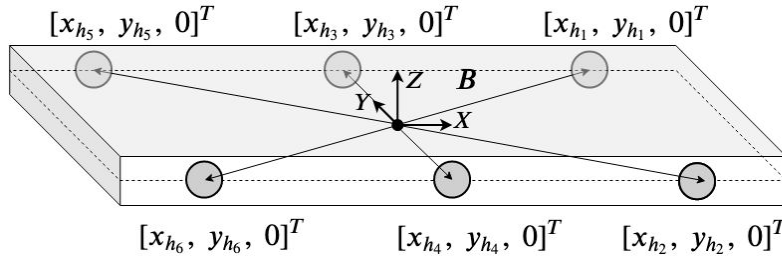


Figure 3-3: Each hip i has a position on the body at $[x_{h_i}, y_{h_i}, 0]$, shown in the diagram.

$$R_{H_i}^B = \underbrace{\begin{bmatrix} \cos \theta_i & 0 & \sin \theta_i \\ 0 & 1 & 0 \\ -\sin \theta_i & 0 & \cos \theta_i \end{bmatrix}}_{R_{\theta_i}} \quad (3-10)$$

These two matrices can be substituted into Eq. (3-4) to produce the homogeneous transformation $H_{H_i}^B$.

$$H_{H_i}^B = \begin{bmatrix} \begin{bmatrix} \cos \theta_i & 0 & \sin \theta_i \\ 0 & 1 & 0 \\ -\sin \theta_i & 0 & \cos \theta_i \end{bmatrix} & \begin{bmatrix} x_{h_i} \\ y_{h_i} \\ z_{h_i} \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & 1 \end{bmatrix} \quad (3-11)$$

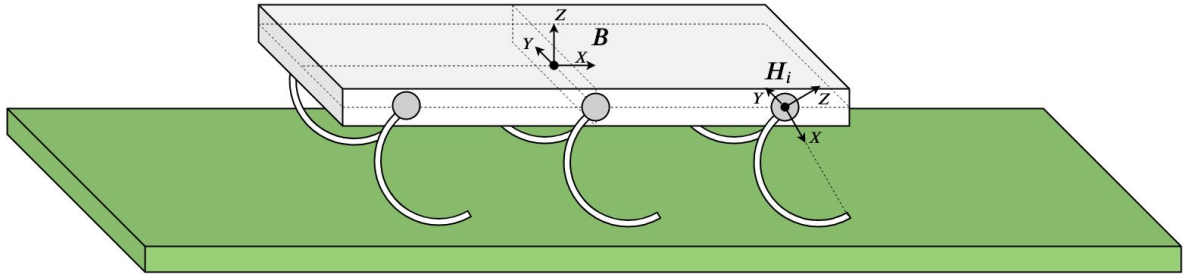


Figure 3-4: The hip frame of leg 2 is shown at the position $[x_{h2}, y_{h2}, 0]$, relative to the B -frame.

3-3-4 Contact frame

The contact frame C_i is a coordinate frame located at the point of contact between leg i and the ground. The C_i Z -axis will always be perpendicular to the ground and the C_i X -axis will always point in the direction the Zebro is heading. This configuration was chosen because the normal force acting on the leg will be purely in the C_i Z -direction and any longitudinal traction force is in the C_i X -direction.

Each leg has three different contact states: rolling-contact, toe-contact and hip-contact. Within the scope of this thesis only rolling and toe-contact are considered, since hip-contact does not occur during a stable walking gait. Each state requires a different transformation to define the contact frame, so it is vital to easily detect which state a leg is in. The next two sections are dedicated to this task.

It is important to note that knowing the leg angle θ_i is not sufficient to define the contact frame. This is shown in Figure 3-5, where θ_p represents a small pitching angle, effectively shifting the position of the contact frame slightly. An angle ϕ_{min} is defined as the angle between the leg's toe and the minimum of the leg's full circle. See Figure 3-5 for some clarification. Note that the angle is taken to be positive in the clockwise direction, meaning that ϕ_{min} in toe-contact (the scenario on the right of Figure 3-5) will always be negative. Using this angle one can find the rotation required to get from the C_i frame to the H_i frame.

It turns out that the value of ϕ_{min} is not just dependent on the leg angle and the pitch angle, but also on the roll and yaw of the body, as will be shown in Section 3-4.

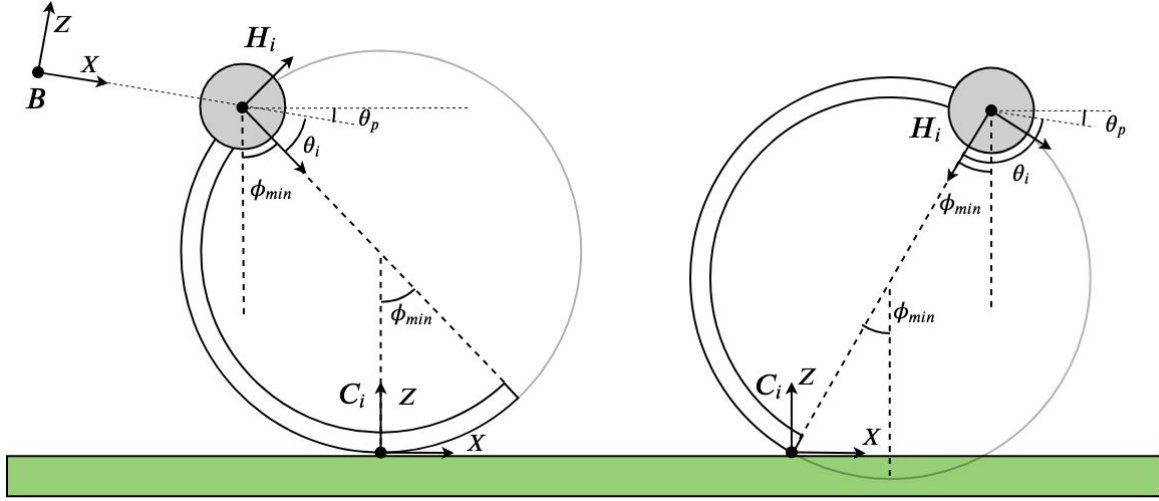


Figure 3-5: Two legs in contact with the ground with the frames H_i and C_i labelled. On the left the leg is in rolling-contact and on the right the leg is in toe-contact. The body is pitched forward at an angle of θ_p .

Homogeneous transformation for rolling-contact

Consider the position of C_i in the H_i frame. This can be deduced from the leg on the left in Figure 3-5 using some trigonometry.

$${}^{H_i}\mathbf{p}_{C_i} = \begin{bmatrix} r + r \cos \phi_{min,i} \\ 0 \\ -r \sin \phi_{min,i} \end{bmatrix} \quad (3-12)$$

Two rotations are required from H_i to C_i , the first of which can be deduced from Figure 3-5. This is a rotation of $-(\frac{\pi}{2} - \phi_{min,i})$ radians around the ${}^{H_i}Y$ -axis, referred to as R_ϕ .

$$R_\phi = \begin{bmatrix} \cos(-(\frac{\pi}{2} - \phi_{min,i})) & 0 & \sin(-(\frac{\pi}{2} - \phi_{min,i})) \\ 0 & 1 & 0 \\ -\sin(-(\frac{\pi}{2} - \phi_{min,i})) & 0 & \cos(-(\frac{\pi}{2} - \phi_{min,i})) \end{bmatrix} \quad (3-13)$$

The second rotation required is less obvious to see in the figure, but arises from the fact that the $C_i Z$ -axis should point in the inertial vertical direction, which is not the case if the body's roll angle $\theta_r \neq 0$. To counteract the roll angle, the inverse of R_{θ_r} from Eq. (3-7) is required.

$${}^{H_i}{}_{C_i} = \begin{bmatrix} R_\phi R_{\theta_r}^{-1} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} r + r \cos \phi_{min,i} \\ 0 \\ -r \sin \phi_{min,i} \\ 1 \end{bmatrix} \quad (3-14)$$

Homogeneous transformation for toe-contact

The rotation from H_i to C_i is exactly the same as in Eq. (3-13) since ϕ_{min} is negative in this case. As for the position of C_i in the H_i frame:

$${}^{H_i}p_{C_i} = \begin{bmatrix} 2r \\ 0 \\ 0 \end{bmatrix} \quad (3-15)$$

This results in the following matrix for $H_{C_i}^{H_i}$ in toe-contact.

$$H_{C_i}^{H_i} = \begin{bmatrix} R_\phi R_{\theta_r}^{-1} & \begin{bmatrix} 2r \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & 1 \end{bmatrix} \quad (3-16)$$

3-4 Finding the leg's potential contact point

A leg's minimum is the lowest point on the leg in the inertial frame. If the leg is in contact with the ground, this is the point at which contact occurs, hence the name “potential contact point”. The C_i -frame is located at this position on the leg. The angle ϕ_{min} which was mentioned previously is now calculated in this section.

Ignore for now the fact that the legs are half-circular. Instead, imagine them to be a full circle, much like the grey circle shown in Figure 3-5 and Figure 3-6. All the points on this circular leg L can be defined in the hip frame using an angle ϕ .

$${}^{H_i}L(\phi) = \begin{bmatrix} r + r \cos \phi \\ 0 \\ -r \sin \phi \end{bmatrix} \quad \text{for } -\pi < \phi \leq \pi \quad (3-17)$$

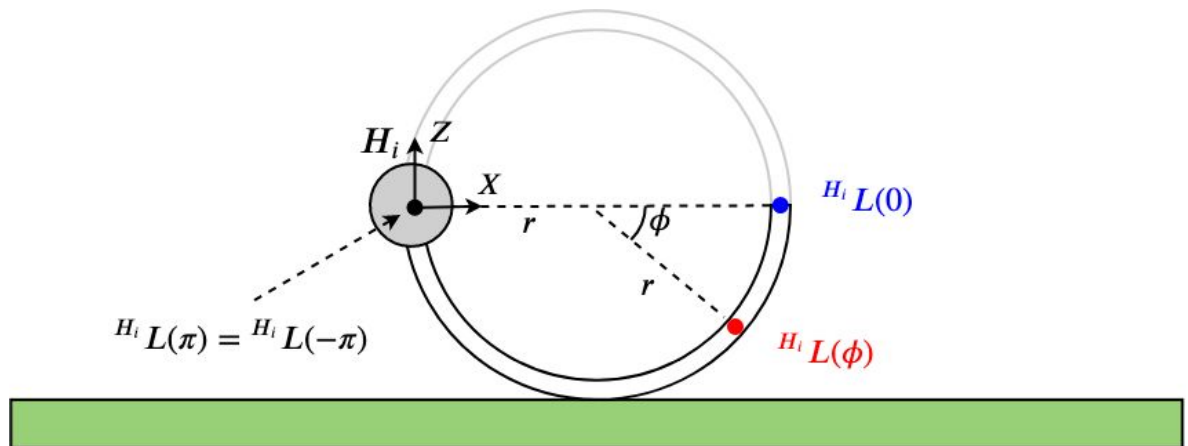


Figure 3-6: The imaginary circular leg is shown with the H_i frame labelled. The blue marker shows the point ${}^{H_i}L(\phi)$ with $\phi = 0$ as can be found in Eq. (3-17). The red marker shows the general point ${}^{H_i}L(\phi)$.

Taking this a step further, the circle can also be defined in the inertial frame using the homogeneous transformations found previously.

$$\begin{bmatrix} {}^I\mathbf{L}(\phi) \\ 1 \end{bmatrix} = H_B^I H_{H_i}^B \begin{bmatrix} r + r \cos \phi \\ 0 \\ -r \sin \phi \\ 1 \end{bmatrix} \quad \text{for } -\pi \leq \phi \leq \pi \quad (3-18)$$

Or, written out fully:

$${}^I\mathbf{L}(\phi) = R_r R_p R_y R_{\theta_i} \begin{bmatrix} r + r \cos \phi \\ 0 \\ -r \sin \phi \end{bmatrix} + R_r R_p R_y \begin{bmatrix} x_{h_i} \\ y_{h_i} \\ z_{h_i} \end{bmatrix} + \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}. \quad (3-19)$$

The two points on ${}^I\mathbf{L}(\phi)$ where the circle is parallel to the ground are the points at which $\frac{dz(\phi)}{d\phi} = 0$.

$$\frac{d^I\mathbf{L}(\phi)}{d\phi} = \begin{bmatrix} \frac{dx(\phi)}{d\phi} \\ \frac{dy(\phi)}{d\phi} \\ \frac{dz(\phi)}{d\phi} \end{bmatrix} = R_r R_p R_y R_{\theta_i} \begin{bmatrix} -r \sin \phi \\ 0 \\ -r \cos \phi \end{bmatrix} \quad (3-20)$$

It is found that $\frac{dz(\phi)}{d\phi} = 0$ is satisfied when

$$\tan(\phi) = \frac{1 + \tan \theta_i (\tan \theta_r \sin \theta_y - \tan \theta_p \cos \theta_y)}{\tan \theta_i - (\tan \theta_r \sin \theta_y - \tan \theta_p \cos \theta_y)} \quad (3-21)$$

which has two unique solutions for $-\pi \leq \phi \leq \pi$, one corresponding to the bottom of the circle (ϕ_{min}) and one corresponding to the top (ϕ_{max}). Which one is which? This can be determined using the second derivative.

$$\frac{d^2 {}^I\mathbf{L}(\phi)}{d\phi^2} = \begin{bmatrix} \frac{d^2 x(\phi)}{d\phi^2} \\ \frac{d^2 y(\phi)}{d\phi^2} \\ \frac{d^2 z(\phi)}{d\phi^2} \end{bmatrix} = R_r R_p R_y R_{\theta_i} \begin{bmatrix} -r \cos \phi \\ 0 \\ r \sin \phi \end{bmatrix} \quad (3-22)$$

Substituting ϕ in Eq. (3-22) with the two results from Eq. (3-21) reveals which result corresponds to the minimum of the circle and which corresponds to the maximum.

$$\frac{d^2 z(\phi_{min})}{d\phi^2} > 0 \quad \text{and} \quad \frac{d^2 z(\phi_{max})}{d\phi^2} < 0 \quad (3-23)$$

3-4-1 Determining the leg's state

A leg can be in one of three states depending on where the leg's minimum is. If the minimum is anywhere on the leg's arc, the leg is said to be in the *rolling-contact* state. If the minimum is at the toe the leg is said to be in the *toe-contact* state and finally, if the minimum is at the hip it is said to be in the *hip-contact* state. All of these possibilities are illustrated in Figure 3-7, using four different examples.

The angle ϕ_{min} has been found in the previous section and the question becomes: if the leg is in contact with the ground, is it in rolling-, toe- or hip-contact? Despite hip-contact being an ignored state for the most part, it is important to be able to differentiate between the three possibilities here. To summarise, the leg state can be determined by finding ϕ_{min} and checking the following conditions.

- If $\phi_{min,i} \geq 0$ then leg i is in rolling-contact
- If $-\frac{\pi}{2} \leq \phi_{min,i} < 0$ then leg i is in toe-contact
- If $-\pi \leq \phi_{min,i} < -\frac{\pi}{2}$ then leg i is in hip-contact

Depending on the result of these checks, $H_{C_i}^{H_i}$ is chosen from the two options in Eq. (3-14) and Eq. (3-16).

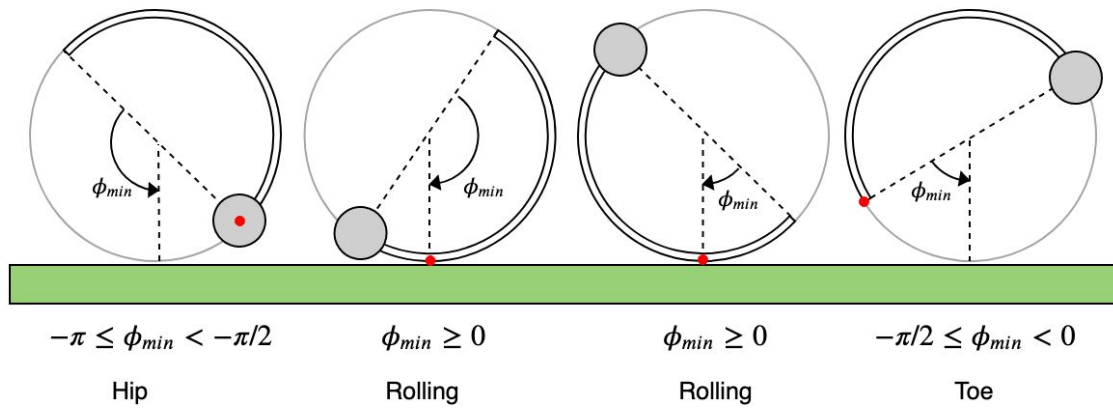


Figure 3-7: The leg is shown in four different positions, each of which can be defined as either hip-contact, rolling-contact or toe-contact by checking the value of ϕ_{min} . The red markers on each leg represent the lowest point on the leg, or the “potential contact”.

3-4-2 Calculating $\dot{\phi}_{min}$

The speed at which the angle ϕ_{min} is changing is important to calculate. This is the speed at which the leg appears to rotate when taking the body's orientation into account. For example, if the body is pitching at the exact same speed and opposite direction as the leg speed $\dot{\theta}_i$, then the leg will appear to not be rotating at all and $\dot{\phi}_{min} = 0$.

$$\dot{\phi}_{min,i} = \frac{\dot{\theta}_r \sin \theta_y (1 + \tan^2 \theta_r) - \dot{\theta}_p \cos \theta_y (1 + \tan^2 \theta_p) + \dot{\theta}_y (\tan \theta_r \cos \theta_y + \tan \theta_p \sin \theta_y)}{1 + (\tan \theta_r \sin \theta_y - \tan \theta_p \cos \theta_y)^2} - \dot{\theta}_i \quad (3-24)$$

3-5 Contact state and contact events

A contact event occurs whenever a leg touches down or lifts off from the ground. When a leg touches down, a contact force is present at its point of contact and when a leg lifts off the contact force becomes 0, hence the dynamic model is heavily dependent on which legs are touching the ground. This is referred to as the robot's *contact state*. Let a contact state be the set of leg numbers in contact with the ground and is given the symbol \mathcal{C} . Since the state changes with time it is a function of t .

$$\mathcal{C}(t) = \{L_1 \dots L_n\}, \quad (3-25)$$

where $L_1 \dots L_n \in \{1, 2, 3, 4, 5, 6\}$ and n is the number of legs touching the ground. For example, in the tripod gait, at some point the contact state may be $\mathcal{C} = \{1, 4, 5\}$, indicating that legs 1, 4 and 5 are touching the ground.

Let the flight set $\mathcal{F}(t)$ contain the leg indices of each leg which is in flight at time t .

$$\mathcal{F}(t) = \{1, 2, 3, 4, 5, 6\} \setminus \mathcal{C}(t) \quad (3-26)$$

3-5-1 Contact polygon

The contact polygon is the shape formed on the ground between all of the contact points. In three-legged contact this shape is a triangle. If the body's centre of mass falls within the contact polygon, the body is stable and balanced. However, as soon as the CoM is outside the contact polygon, a "toppling" effect can be observed where a leg lifts off of the ground and the body topples over the two legs closest to the centre of mass.

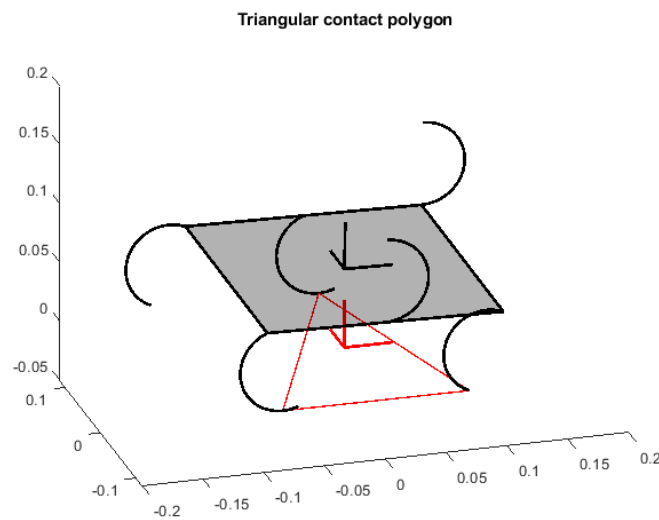


Figure 3-8: An example of a triangular contact polygon for $\mathcal{C} = \{2, 3, 6\}$

3-5-2 Detecting contact events

Assume first that the initial contact state $\mathcal{C}(0)$ is known. As the legs rotate, one of two things can happen:

1. A leg in \mathcal{C} lifts off of the ground.
2. A leg in \mathcal{F} touches down.

Usually a lift-off event occurs at the same time as a different leg touching down, so it would be simple to assume that each contact event occurs in a touchdown and lift-off pair. However, there are two exceptions to this assumption. First, the toppling scenario mentioned in Section 3-5-1 consists of just a lift-off event, with touchdown occurring once the Zebro has toppled. Second, in the tetrapod gait it is theoretically possible to have four legs touching the ground, meaning that a touchdown event occurred without a lift-off.

Detecting a touchdown

Detecting a touchdown event is fairly trivial, one must just check the inertial z -coordinate of the potential contact position of each leg in $\mathcal{F}(t)$. If this value is less than or equal to 0, then at some time between $t - dt$ and t the leg touched the ground. In this model the assumption is made that dt is small enough such that the touchdown time can be taken to be t . If touchdown is detected for leg i , then

$$\mathcal{C}(t) \leftarrow \mathcal{C}(t - dt) \cup i \quad (3-27)$$

$$\mathcal{F}(t) \leftarrow \mathcal{F}(t - dt) \setminus i \quad (3-28)$$

As mentioned in the previous section, a touchdown usually occurs alongside a lift-off, so the next step is to detect which leg (if any) will lift up from the ground.

Detecting a lift-off

Lift-off occurs for two reasons: first, when a leg touches down and effectively replaces a different leg in \mathcal{C} , and second, when the body's centre of mass exits the contact polygon and the body topples.

Finding which leg gets replaced when a leg i touches down is done in a few steps.

1. All new contact state possibilities in which i is included are calculated.
2. All legs are rotated to the angle at $t + dt$.
3. Each possible contact state found in step 1 is tested by orienting the robot such that each leg in the possible contact state is touching the ground. If any leg which is not in the possible contact state is below the ground, this possible state is discarded.

It was found that in some cases, two possible contact states were feasible. Choosing between the two is done by checking in which state the body's centre of mass falls within the new contact polygon as explained in the following section.

Determining if a point is in the contact polygon

Assume the contact polygon to be a triangle. This is a fair assumption since the only case of feasible four-legged contact in the tetrapod gait is guaranteed to contain the centre of mass. Therefore, the contact polygon will be a triangle in the inertial XY -plane, with vertices \mathbf{p}_i , \mathbf{p}_j and \mathbf{p}_k where $i, j, k \in \mathcal{C}(t)$. The centre of mass is projected onto the XY -plane by simply moving it vertically down to $z = 0$: $\mathbf{p}_0 = [x_b \ y_b \ 0]^T$.

Determining whether \mathbf{p}_0 is within the contact triangle is done following a technique outlined in a blog post by “blackpaw” [19]. Observe Figure 3-9, the cross product of $\mathbf{B} - \mathbf{A}$ and $\mathbf{p} - \mathbf{A}$ will produce a vector pointing in the opposite direction to the cross product of $\mathbf{B} - \mathbf{A}$ and $\mathbf{p}' - \mathbf{A}$, indicating that \mathbf{p} and \mathbf{p}' are on different sides of the line.

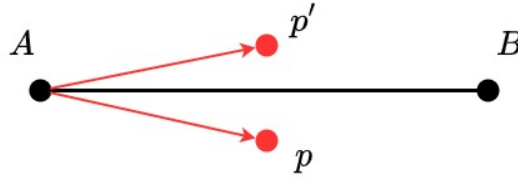


Figure 3-9: Two points p and p' on either side of a line AB .

It follows that, for the point p to be in a triangle ABC , it must be on the same side of AB as point C , see Figure 3-10. Or in other words, $(\mathbf{B} - \mathbf{A}) \times (\mathbf{p} - \mathbf{A})$ must point in the same direction as $(\mathbf{B} - \mathbf{A}) \times (\mathbf{C} - \mathbf{A})$. One can confirm that these two vectors point in the same direction if their dot product is greater than 0.

$$((\mathbf{B} - \mathbf{A}) \times (\mathbf{p} - \mathbf{A})) \cdot ((\mathbf{B} - \mathbf{A}) \times (\mathbf{C} - \mathbf{A})) > 0 \quad (3-29)$$

This must be checked for each vertex of the triangle, since p must be on the “correct” side of each of the triangle’s edges to be considered inside.

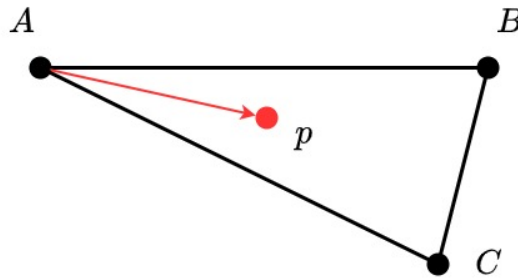


Figure 3-10: A point p inside the triangle ABC

In terms of the previously established coordinates of the points of contact and the body’s centre of mass, the conditions required for the centre of mass to be in the contact triangle are the following.

$$({}^I\mathbf{p}_{C_i} - {}^I\mathbf{p}_{C_j}) \times \begin{bmatrix} x_b \\ y_b \\ 0 \end{bmatrix} - {}^I\mathbf{p}_{C_j} \cdot ({}^I\mathbf{p}_{C_i} - {}^I\mathbf{p}_{C_j}) \times ({}^I\mathbf{p}_{C_k} - {}^I\mathbf{p}_{C_j}) \geq 0 \quad (3-30)$$

$$({}^I\mathbf{p}_{C_j} - {}^I\mathbf{p}_{C_k}) \times \begin{bmatrix} x_b \\ y_b \\ 0 \end{bmatrix} - {}^I\mathbf{p}_{C_k} \cdot ({}^I\mathbf{p}_{C_j} - {}^I\mathbf{p}_{C_k}) \times ({}^I\mathbf{p}_{C_i} - {}^I\mathbf{p}_{C_k}) \geq 0 \quad (3-31)$$

$$({}^I\mathbf{p}_{C_k} - {}^I\mathbf{p}_{C_i}) \times \begin{bmatrix} x_b \\ y_b \\ 0 \end{bmatrix} - {}^I\mathbf{p}_{C_i} \cdot ({}^I\mathbf{p}_{C_k} - {}^I\mathbf{p}_{C_i}) \times ({}^I\mathbf{p}_{C_j} - {}^I\mathbf{p}_{C_i}) \geq 0 \quad (3-32)$$

If all three of these conditions are met, it can be concluded that the body's CoM falls within the contact triangle produced by legs i, j and k . If the condition in Eq. (3-29) is not met, then it can be concluded that the leg represented by C will lift off of the ground. This translates to leg k for Eq. (3-30), i for Eq. (3-31) and j for Eq. (3-32).

3-5-3 Contact transitions in a tetrapod gait

In Suriana's kinematic model of the tetrapod gait, it was always assumed that the Zebro's contact state followed the theoretical tetrapod gait perfectly in the pattern shown in Figure 3-11. In the figure the colours red, green and blue are used to indicate a pair of legs which are rotating together and which *should* be in contact with the ground. The white legs are the legs which are in the air.

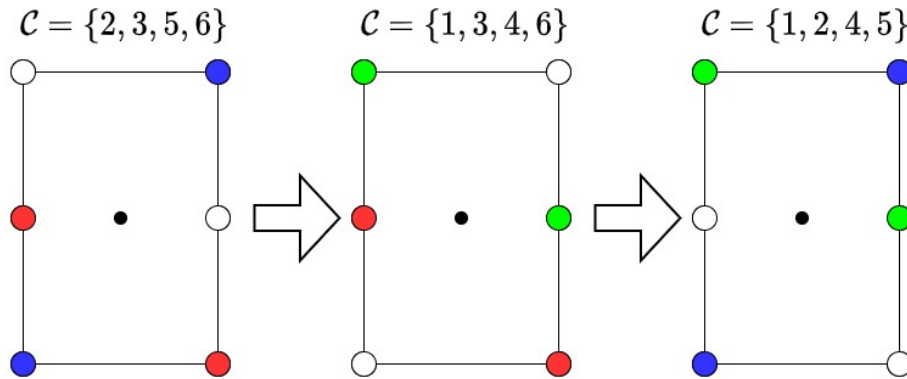


Figure 3-11: The theoretical gait progression in a tetrapod gait as used by Suriana. The separate leg groups which move synchronously are indicated with colours.

The problem with Suriana's assumption is that four-legged contact does not always occur. Rather, the Zebro pitches and rolls due to differences in hip heights for the legs in C , causing one leg to be in the air, which according to Suriana should be in contact. This is shown in Figure 3-12, in which the legs are coloured as in the convention in Figure 3-11.

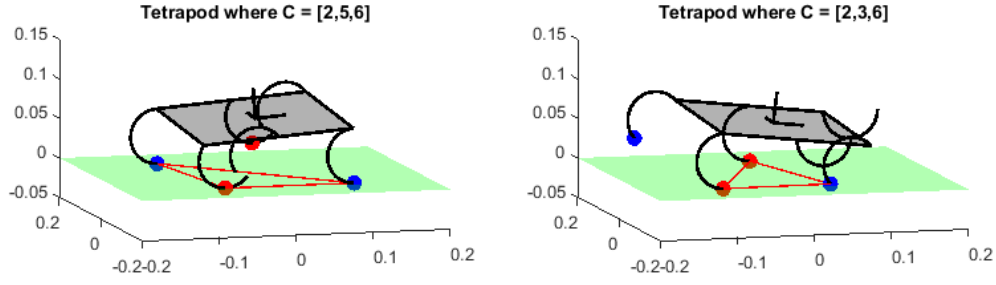


Figure 3-12: Two possible contact states within one “step” of the tetrapod gait. Left: $\mathcal{C} = \{2, 5, 6\}$, right: $\mathcal{C} = \{2, 3, 6\}$.

Instead of the progression shown in Figure 3-11, through simulation with the previously mentioned contact detection algorithm the following contact state progression was found.

$$\{2, 5, 6\} \rightarrow \{2, 3, 6\} \rightarrow \{1, 3, 4, 6\} \rightarrow \{1, 4, 5\} \rightarrow \{2, 4, 5\} \rightarrow \{2, 5, 6\} \quad (3-33)$$

3-5-4 Contact transitions in a tripod gait

In a tripod gait there are two leg pairs which move synchronously: $\{1, 4, 5\}$ and $\{2, 3, 6\}$. If it is assumed that there is no pitching of the body, which is a reasonable assumption for the tripod gait, then these leg pairs will always undergo a contact event together.

Generally in robot dynamics, contact is modelled as a kinematic constraint [20]. If \mathbf{p}_c is the position of contact, it is not allowed to move anymore:

$$\mathbf{p}_c = \text{constant} \quad (3-34)$$

$$\dot{\mathbf{p}}_c = \mathbf{J}_c \dot{\mathbf{q}} = \mathbf{0} \quad (3-35)$$

$$\ddot{\mathbf{p}}_c = \mathbf{J}_c \ddot{\mathbf{q}} + \dot{\mathbf{J}}_c \dot{\mathbf{q}}, \quad (3-36)$$

where \mathbf{J}_c is the geometric Jacobian of the contact point. The problem with this contact model in the case of the Zebro is that it does not allow for any slippage, since slip is by definition movement of the contact point.

A different method of incorporating contact into the dynamic model of the Zebro was used, namely a simple hybrid automaton in which two contact sets \mathcal{C}^i and \mathcal{C}^o can change discretely. \mathcal{C}^i is the set of legs in contact with the ground on the *inner* side of the Zebro and \mathcal{C}^o is the set of legs in contact with the ground on the *outer* side of the Zebro. A left turn is used in the theory in this thesis, so the legs on the inner side are 1, 3, 5 and the legs on the outer side are 2, 4, 6. The set of all legs in contact with the ground is $\mathcal{C} = \mathcal{C}^i \cup \mathcal{C}^o$.

Let \dot{z}^i be the vertical velocity of the inner side of the Zebro and \dot{z}^o the same for the outer side. These velocities are shown in Figure 3-14. The parameter $z_{h,i}$ is the vertical distance between a leg’s potential contact point and its hip, shown in Figure 3-13. The following table describes the discrete transition in the contact state and at which condition this occurs. Essentially, contact is detected separately per side by looking at the hip heights of the legs in

flight and comparing them to the legs in contact. These hip heights are found kinematically in the following section.

Transition	Condition	Reset
$\mathcal{C}^i = \{1, 5\} \rightarrow \mathcal{C}^i = \{3\}$	$\frac{1}{2}(z_{h,1} + z_{h,5}) > z_{h,3}$	$\dot{z}^i = \dot{z}_{h,3}$
$\mathcal{C}^i = \{3\} \rightarrow \mathcal{C}^i = \{1, 5\}$	$z_{h,3} > \frac{1}{2}(z_{h,1} + z_{h,5})$	$\dot{z}^i = \frac{1}{2}(\dot{z}_{h,1} + \dot{z}_{h,5})$
$\mathcal{C}^o = \{2, 6\} \rightarrow \mathcal{C}^o = \{4\}$	$\frac{1}{2}(z_{h,2} + z_{h,6}) > z_{h,4}$	$\dot{z}^o = \dot{z}_{h,4}$
$\mathcal{C}^o = \{4\} \rightarrow \mathcal{C}^i = \{2, 6\}$	$z_{h,4} > \frac{1}{2}(z_{h,2} + z_{h,6})$	$\dot{z}^o = \frac{1}{2}(\dot{z}_{h,2} + \dot{z}_{h,6})$

Table 3-1: The table shows the condition which causes a hybrid state transition to occur. Each transition also incurs a reset of the velocity of the side of the Zebro undergoing a transition. This is essentially the vertical velocity of the body changing discretely as the Zebro “bounces” up and down.

3-5-5 Leg kinematics: hip height and its vertical velocity

Calculating the height of hip i is done using some trigonometry. Figure 3-13 shows the hip height in the leg’s sagittal plane, so a multiplication with $\cos \theta_r$ is required to account for the rolling angle of the body.

$$\begin{aligned} \text{Rolling-contact: } z_{h,i} &= (r + r \cos \phi_{min,i}) \cos \theta_r \\ \text{Toe-contact: } z_{h,i} &= 2r \cos \phi_{min,i} \cos \theta_r \end{aligned} \quad (3-37)$$

Its velocity is simply the time derivative of Eq. (3-37). Note that the assumption $\dot{\theta}_r \approx 0$ is made, because in a tripod gait the rolling velocity should be minimal.

$$\begin{aligned} \text{Rolling-contact: } \dot{z}_{h,i} &\approx -r \dot{\phi}_{min,i} \sin \phi_{min,i} \cos \theta_r \\ \text{Toe-contact: } \dot{z}_{h,i} &\approx -2r \dot{\phi}_{min,i} \sin \phi_{min,i} \cos \theta_r \end{aligned} \quad (3-38)$$

The acceleration is the time derivative of Eq. (3-38). Note that it is assumed that $\ddot{\phi}_{min,i} \approx 0$ since $\ddot{\theta}_i = 0$.

$$\begin{aligned} \text{Rolling-contact: } \ddot{z}_{h,i} &\approx -r \dot{\phi}_{min,i}^2 \cos \phi_{min,i} \cos \theta_r \\ \text{Toe-contact: } \ddot{z}_{h,i} &\approx -2r \dot{\phi}_{min,i}^2 \cos \phi_{min,i} \cos \theta_r \end{aligned} \quad (3-39)$$

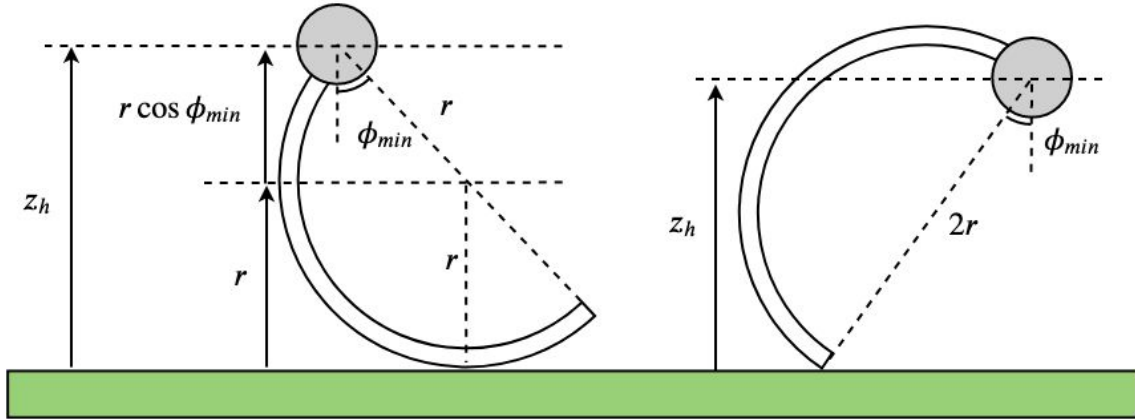


Figure 3-13: The vertical distance between the hip and the lowest point on a leg is shown for both the rolling and toe-contact cases.

3-5-6 Body kinematics: vertical velocity of the body and its roll angle

Using kinematics for the vertical displacement and velocity of the body was deemed acceptable, because the main problem with Suriana's kinematic model was the effect of slip, which only has an effect on the body's horizontal velocity parallel to the ground and not the vertical component. It is simply impossible for slip to occur vertically.

The body's vertical position, velocity and acceleration are shown in the following equation. The expressions for $z^i, z^o, \dot{z}^i, \dot{z}^o, \ddot{z}^i$ and \ddot{z}^o can be found in the Equations (3-37) to (3-39). Since the body's CoM is in the middle between the inner and outer side, the vertical position, velocity and acceleration of the CoM is simply the average between the two sides.

$$z_b = \frac{z^o + z^i}{2}, \quad \dot{z}_b = \frac{\dot{z}^o + \dot{z}^i}{2}, \quad \ddot{z}_b = \frac{\ddot{z}^o + \ddot{z}^i}{2} \quad (3-40)$$

The angle θ_r can also be found kinematically.

$$w \sin \theta_r = \dot{z}^i - \dot{z}^o, \quad \dot{\theta}_r w \cos \theta_r = \ddot{z}^i - \ddot{z}^o \quad (3-41)$$

The values for \ddot{z}_b and $\dot{\theta}_r$ will be important in Section 4-4-2.

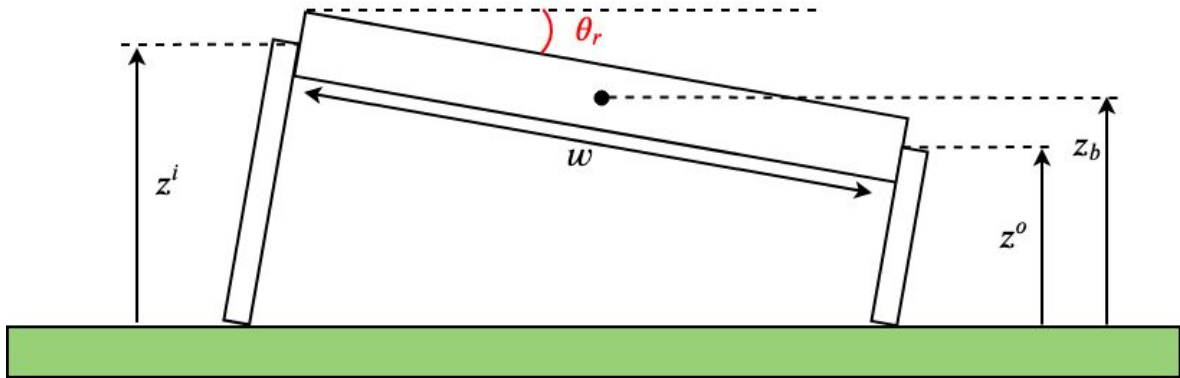


Figure 3-14: The roll angle θ_r is shown in red. This angle arises due to a difference in height between the inner and outer side.

3-6 Summary

Four coordinate frames have been defined: inertial, body, hip and contact. Each of which has a homogeneous transformation to translate a point in one frame to the next. This way, the point $[0, 0, 0]^T$ in the C -frame (the position of contact) can be expressed in inertial coordinates. Two leg states have been used: Rolling-contact and Toe-contact, each with a different transformation between the Hip frame and the Contact frame. The state which a leg is currently in is a function of ϕ_{min} where

$$\tan(\phi_{min,i}) = \frac{1 + \tan \theta_i (\tan \theta_r \sin \theta_y - \tan \theta_p \cos \theta_y)}{\tan \theta_i - (\tan \theta_r \sin \theta_y - \tan \theta_p \cos \theta_y)}. \quad (3-42)$$

If $\phi_{min,i} \geq 0$ then leg i is in rolling-contact and if $-\frac{\pi}{2} \leq \phi_{min,i} < 0$ then leg i is in toe-contact. Hip-contact is also a possible state but is ignored throughout this thesis since it is assumed the body will never touch the ground.

A set of conditions for touchdown and lift-off detection were found by checking if the body's CoM is within the contact polygon, resulting in an explanation for the wobbling in the tetrapod gait and the prediction that the contact state progression in the tetrapod gait is as follows.

$$\{2, 5, 6\} \rightarrow \{2, 3, 6\} \rightarrow \{1, 3, 4, 6\} \rightarrow \{1, 4, 5\} \rightarrow \{2, 4, 5\} \rightarrow \{2, 5, 6\} \quad (3-43)$$

Finally, a simpler contact transition algorithm was described for the tripod gait. Since minimal wobbling is expected for the tripod, the contact state transitions are simpler to predict and can be modelled using a hybrid automaton where a transition occurs as soon as the hip height of the leg(s) in the air on one side exceeds the hip height of the leg(s) on the ground on that same side. This is followed up with some of the kinematics used to find these hip heights, as well as the kinematically determined vertical acceleration of the body.

Chapter 4

Dynamics of the Body

4-1 Introduction

A dynamic model is a set of equations of motion that govern how a system moves. To model the Zebro, one must derive an equation that calculates the accelerations of the body as a result of the forces acting upon the body. This is also known as “forward dynamics”, as opposed to inverse dynamics which is used to calculate the forces using known accelerations. A dynamic model differs from a kinematic model since kinematics concerns itself with the geometry and velocities of a body, rather than forces and torques. In a dynamic model one can incorporate aspects which have been previously ignored in kinematic models such as slip, gravity and contact forces.

Contact forces are introduced in Section 4-2, which are subsequently used in the Newton-Euler framework in Section 4-3. Next, a model for the weight distribution of the Zebro is looked at, since this gives a good indication of the forces acting on a leg.

Next, in Section 4-5 a model for traction forces is given, namely “Pacejka’s magic formula”. Generally, this model is used for car tyres, so it is altered slightly to be applicable to the Zebro’s half-circular legs.

Finally, in Section 4-6 the experimental setup for identifying the slip model mentioned previously is explained. This is an experiment which was designed and performed in conjunction with a separate research group at the TU Delft who wrote the following report on their results: [21]. Data gathered by myself casts some doubt on the accuracy of the results in [21], but improving their model is left for future work since the experimental setup was no longer available for more sophisticated experiments.

4-2 Contact forces

Where there is contact, there is a contact force present. Let the contact force vector exerted by the ground on leg i be $\mathbf{F}_{c,i}$ which is equal to $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ if $i \notin \mathcal{C}$.

$${}^I \mathbf{F}_{c,i} = \begin{bmatrix} {}^I f_{x,i} \\ {}^I f_{y,i} \\ {}^I f_{z,i} \end{bmatrix} \quad \text{for } i \in \mathcal{C} \quad (4-1)$$

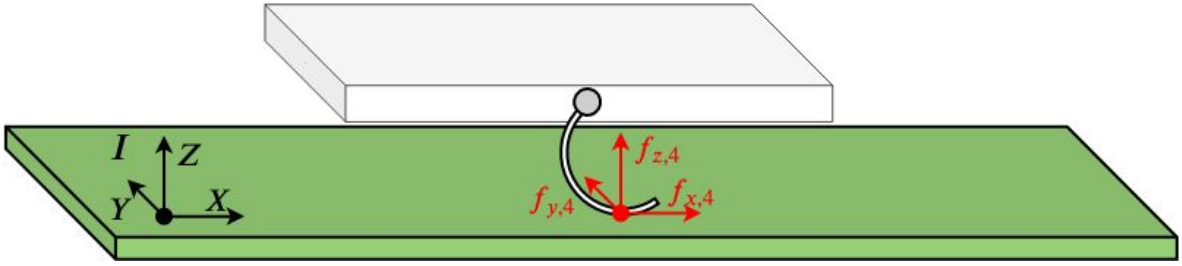


Figure 4-1: The general contact force in the inertial frame is shown in red.

In the contact coordinate frame \mathcal{C} these forces can be given a more physical context, namely: the normal force and the traction force. The normal force is the force perpendicular to the ground and the traction force is the force produced by rotating or sliding over the ground. The traction force consists of two components, the longitudinal traction which is in the ${}^C X$ -direction and the lateral traction which is in the ${}^C Y$ -direction.

$${}^C \mathbf{F}_{c,i} = \begin{bmatrix} f_{lon,i} \\ f_{lat,i} \\ f_{n,i} \end{bmatrix} \quad (4-2)$$

Both $f_{lon,i}$ and $f_{lat,i}$ are functions of $f_{n,i}$, which is a topic that will be explored into further detail in Section 4-5-2. For now, suffice to say

$$f_{lon,i} = P_{lon}(f_{n,i}) \quad \text{and} \quad f_{lat,i} = P_{lat}(f_{n,i}) \quad (4-3)$$

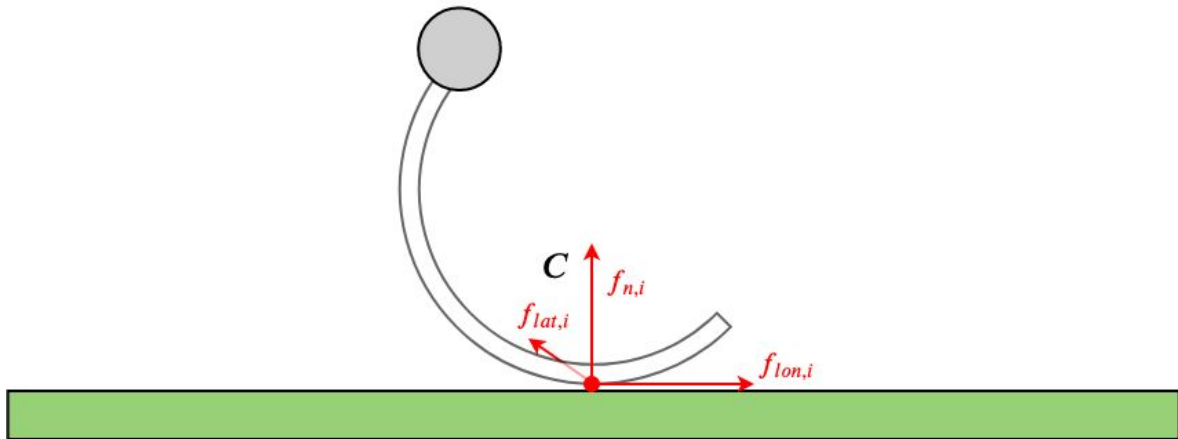


Figure 4-2: In the C_i frame, the contact force can be split up into a normal force, a lateral traction force and a longitudinal traction force.

4-3 Newton-Euler framework

The Newton-Euler framework is a method of obtaining the equations of motion by separately evaluating the linear and rotational dynamics. Since the Zebro is assumed to be a cuboid with mass m_b and massless legs, no complicated multibody dynamics is required. Instead, the Zebro is analysed as a single rigid body with several forces applied at the contact points.

4-3-1 Newton's equation

The linear dynamics of a body is simply the forces acting on it equated with the accelerations of the body multiplied by the mass.

$$\mathbf{F} = m\mathbf{a} \quad (4-4)$$

The vector \mathbf{F} is the sum of all the forces acting on the body, which is essentially the sum of all the contact forces in the inertial frame minus the weight of the body. The vector \mathbf{a} is the acceleration of the body's Centre of Mass in the inertial x -, y - and z -direction.

$$\sum_{i \in \mathcal{C}} ({}^I \mathbf{F}_{c,i}) - \begin{bmatrix} 0 \\ 0 \\ m_b g \end{bmatrix} = \begin{bmatrix} m_b \ddot{x}_b \\ m_b \ddot{y}_b \\ m_b \ddot{z}_b \end{bmatrix} \quad (4-5)$$

4-3-2 Euler's equation

Euler's equation deals with the rotational dynamics of a rigid body. Let \mathbf{M} be the moments acting on the body, I_b be the body's inertia tensor and $\boldsymbol{\omega}$ be the rotational velocity of the body.

$$\mathbf{M} = I_b \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (I_b \boldsymbol{\omega}) \quad (4-6)$$

The moment acting on the body is simply the summation of the cross products of the distance from the body's CoM to the contact point and the force acting on the body at said contact point.

$$\mathbf{M} = \sum_{i \in \mathcal{C}} \left(({}^I \mathbf{p}_{c,i} - {}^I \mathbf{p}_b) \times {}^I \mathbf{F}_{c,i} \right) \quad (4-7)$$

Substituting Eq. (4-7) and some of the generalised coordinates into Eq. (4-6) leads to the following equation for the Zebro's rotational motion [22].

$$\sum_{i \in \mathcal{C}} \left(({}^I \mathbf{p}_{c,i} - {}^I \mathbf{p}_b) \times {}^I \mathbf{F}_{c,i} \right) = I_b \begin{bmatrix} \ddot{\theta}_r \\ \ddot{\theta}_p \\ \ddot{\theta}_y \end{bmatrix} + \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_p \\ \dot{\theta}_y \end{bmatrix} \times \left(I_b \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_p \\ \dot{\theta}_y \end{bmatrix} \right) \quad (4-8)$$

The inertia tensor of a cuboid is

$$I_b = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (4-9)$$

where $I_{xx} = m_b(w^2 + h^2)/12$, $I_{yy} = m_b(l^2 + h^2)/12$ and $I_{zz} = m_b(w^2 + l^2)/12$, with w , l and h the dimensions of the body as shown in Figure 1-2.

4-3-3 General form equations of motion

The general form for equations of motion of a legged robot, as given in [20], is

$$M(\mathbf{q})\ddot{\mathbf{q}} + b(\mathbf{q}, \dot{\mathbf{q}}) + g(\mathbf{q}) = \boldsymbol{\tau} \quad (4-10)$$

where $\boldsymbol{\tau}$ is a vector containing the forces and moments on the system, $M(\mathbf{q})$ is a generalised orthogonal mass matrix, $b(\mathbf{q}, \dot{\mathbf{q}})$ contains the Coriolis terms and $g(\mathbf{q})$ is the gravitational term.

To arrive at this form, combine Eq. (4-5) and Eq. (4-8) as shown in Eq. (4-11).

$$\underbrace{\begin{bmatrix} m_b I_3 & \mathbf{0}^T & 0 & \dots & 0 \\ \mathbf{0}^T & I_b & 0 & \dots & 0 \\ \mathbf{0}^T & \mathbf{0}^T & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}^T & \mathbf{0}^T & 0 & \dots & 0 \end{bmatrix}}_M + \underbrace{\begin{bmatrix} \ddot{x}_b \\ \ddot{y}_b \\ \ddot{z}_b \\ \ddot{\theta}_r \\ \ddot{\theta}_p \\ \ddot{\theta}_y \\ \ddot{\theta}_1 \\ \vdots \\ \ddot{\theta}_6 \end{bmatrix}}_{\ddot{\mathbf{q}}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \dot{\theta}_p \dot{\theta}_y (I_{zz} - I_{yy}) \\ \dot{\theta}_r \dot{\theta}_y (I_{xx} - I_{zz}) \\ \dot{\theta}_p \dot{\theta}_r (I_{yy} - I_{xx}) \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_b + \underbrace{\begin{bmatrix} 0 \\ 0 \\ g \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_g = \underbrace{\begin{bmatrix} \sum_{i \in \mathcal{C}} ({}^I \mathbf{F}_{c,i}) \\ \sum_{i \in \mathcal{C}} \left(({}^I \mathbf{p}_{c,i} - {}^I \mathbf{p}_b) \times {}^I \mathbf{F}_{c,i} \right) \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{\boldsymbol{\tau}} \quad (4-11)$$

Solving this equation to find the body's accelerations is a simple case of rearranging.

$$\ddot{\mathbf{q}} = M^{-1}(\mathbf{q}) (\boldsymbol{\tau} - b(\mathbf{q}, \dot{\mathbf{q}}) - g(\mathbf{q})) \quad (4-12)$$

There are a lot of redundant zeroes in M in Eq. (4-11) which can be ignored to simplify the expression by temporarily using \mathbf{q}^* instead of \mathbf{q} .

$$\mathbf{q}^* = [x_b \quad y_b \quad z_b \quad \theta_r \quad \theta_p \quad \theta_y]^T \quad (4-13)$$

$$\underbrace{\begin{bmatrix} m_b I_3 & \mathbf{0} \\ \mathbf{0} & I_b \end{bmatrix}}_M \underbrace{\begin{bmatrix} \ddot{x}_b \\ \ddot{y}_b \\ \ddot{z}_b \\ \ddot{\theta}_r \\ \ddot{\theta}_p \\ \ddot{\theta}_y \end{bmatrix}}_{\mathbf{\ddot{q}}^*} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \dot{\theta}_p \dot{\theta}_y (I_{zz} - I_{yy}) \\ \dot{\theta}_r \dot{\theta}_y (I_{xx} - I_{zz}) \\ \dot{\theta}_p \dot{\theta}_r (I_{yy} - I_{xx}) \end{bmatrix}}_b + \underbrace{\begin{bmatrix} 0 \\ 0 \\ g \\ 0 \\ 0 \\ 0 \end{bmatrix}}_g = \underbrace{\begin{bmatrix} \sum_{i \in \mathcal{C}} ({}^I \mathbf{F}_{c,i}) \\ \sum_{i \in \mathcal{C}} (({}^I \mathbf{p}_{c,i} - {}^I \mathbf{p}_b) \times {}^I \mathbf{F}_{c,i}) \end{bmatrix}}_{\boldsymbol{\tau}} \quad (4-14)$$

The only information missing which prevents the body's accelerations from being calculated are the contact forces $\mathbf{F}_{c,i}$ for $i \in \mathcal{C}$. How these forces are estimated is handled in the following section.

4-4 Weight distribution estimation

To estimate the normal force on a leg, one can start by finding the weight distribution over the legs. If the contact point of a leg j is closer to the body's CoM than the contact point of leg k , then it makes sense that leg j is carrying more of the robot's weight than leg k . Exactly how much more can be estimated using geometry.

Let the line \mathbf{r}_{ij} be the vector from contact point i to contact point j . Let d_k be the perpendicular distance from contact point k to the line \mathbf{r}_{ij} , where $i \neq j \neq k$. Finally, d_{Bk} is the distance from the body's centre of mass to the line \mathbf{r}_{ij} . Figure 4-3 shows an example in which $i = 1$, $j = 4$ and $k = 5$.

In static equilibrium, the moment exerted by the normal force on leg 5 ($f_{n,5}$) around \mathbf{r}_{ij} should be equal to the moment exerted by the Zebro's weight. Of course, when walking the Zebro is certainly not in static equilibrium, but this gives a reasonable estimate from which a minimisation function can be used to get a more accurate solution. This is covered in Section 4-4-2.

$$\hat{f}_{n,k} d_k = m g d_{Bk} \quad (4-15)$$

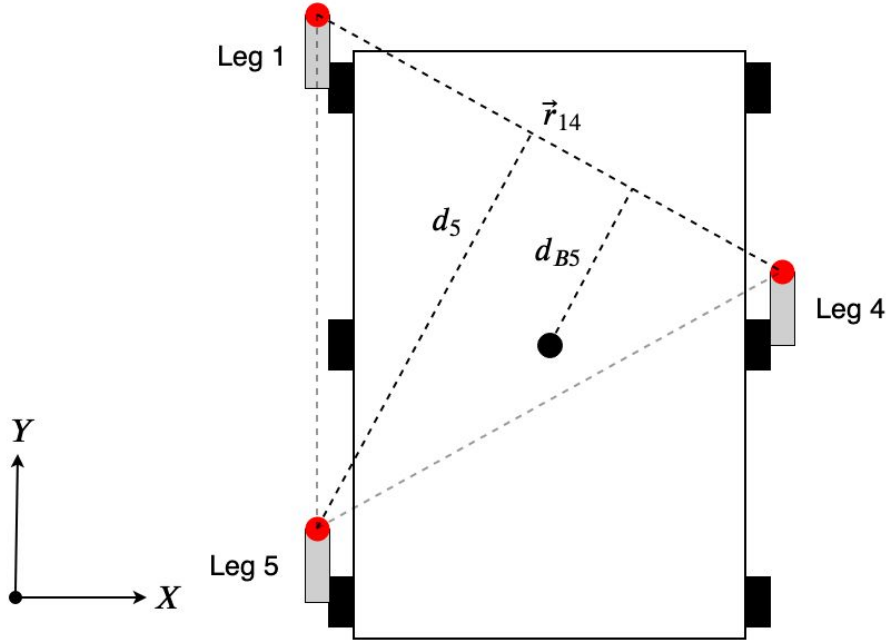


Figure 4-3: Bird's eye view of the Zebro showing the points of contact in red and the CoM as a black marker in the middle.

4-4-1 Distance between a point and a line defined by two points

Let the inertial x -coordinate of contact point i be x_i and the inertial y -coordinate of contact point i be y_i . The distances d_k and d_{Bk} can be found using the following equations [23].

$$d_k = \frac{|(y_j - y_i)x_k - (x_j - x_i)y_k + x_j y_i - y_j x_i|}{\sqrt{(y_j - y_i)^2 + (x_j - x_i)^2}} \quad (4-16)$$

$$d_{Bk} = \frac{|(y_j - y_i)x_b - (x_j - x_i)y_b + x_j y_i - y_j x_i|}{\sqrt{(y_j - y_i)^2 + (x_j - x_i)^2}} \quad (4-17)$$

Rearranging Eq. (4-15) and substituting d_k and d_{Bk} allows for finding an initial estimate for the normal force on leg k , as long as legs i, j and k are in contact with the ground.

$$\hat{f}_{n,k} = \frac{mg d_{Bk}}{d_k} \quad (4-18)$$

4-4-2 Using an optimisation function to improve the estimate

It has already been established that the traction force on a leg is some function of the normal force, $f_{lon,i} = P_{lon}(f_{n,i})$. This function will be discussed later in Section 4-5-2. Using the initial estimates for $f_{n,i}$ as found using Eq. (4-18) is not sufficient since these values are calculated under the premise of static equilibrium. As shown in Figure 4-4 the traction forces

on the legs will cause a pitching moment on the body, similar to how a motorcycle performs a wheelie.

This incorrect pitching moment \hat{M}_p can be calculated as

$$\hat{M}_p = \sum_{i \in \mathcal{C}} (\hat{f}_{n,i} x_i + \hat{f}_{lon,i} z_b) \quad (4-19)$$

where x_i is the x -coordinate of the contact point in the B -frame. Figure 4-4 shows the case in which $\mathcal{C} = \{1, 4, 5\}$. The rolling moment can be calculated similarly,

$$\hat{M}_r = \sum_{i \in \mathcal{C}} (\hat{f}_{n,i} y_i + \hat{f}_{lat,i} z_b) \quad (4-20)$$

where y_i is the y -coordinate of the contact point in the B -frame. This can be taken a step further by using the current estimates for $\hat{f}_{n,i}$ in the Newton-Euler equations of motion in Eq. (4-14) resulting in an acceleration vector $\hat{\ddot{\mathbf{q}}}$. Any unwanted behaviour can now be filtered out by optimising $\hat{f}_{n,i}$ such that $\hat{\mathbf{q}}$ conforms to some constraints unique to the tripod gait.

The tripod gait constraints are

- Pitching speed $\dot{\theta}_p = 0$ because the front and back legs of each side move in unison
- Rolling speed $\dot{\theta}_r$ is kinematically determined in Section 3-5-6
- The sum of the normal forces is equal to $m_b \ddot{z}_b + m_b g$, where \ddot{z}_b is kinematically determined in Section 3-5-6

Note that these constraints are actually on the generalised velocity $\dot{\mathbf{q}}$, whereas $\ddot{\mathbf{q}}$ is calculated in the equations of motion. However, since $\dot{\mathbf{q}}(t - dt)$ is known, $\dot{\mathbf{q}}(t)$ can be found.

$$\dot{\mathbf{q}}(t) = \dot{\mathbf{q}}(t - dt) + \ddot{\mathbf{q}} dt \quad (4-21)$$

To optimise the estimates $\hat{f}_{n,i}$, the following function must be minimised. the values of $\hat{f}_{n,i}$ for which the equation Eq. (4-22) is 0, are the values which can be used.

$$\begin{aligned} & \arg \min_{\hat{f}_{n,i} \text{ for } i \in \mathcal{C}} \left((\hat{\theta}_p - \dot{\theta}_p)^2 + (\hat{\theta}_r - \dot{\theta}_r)^2 \right) \\ & \text{s.t. } \sum_{i \in \mathcal{C}} \hat{f}_{n,i} = m_b \ddot{z}_b + m_b g \end{aligned} \quad (4-22)$$

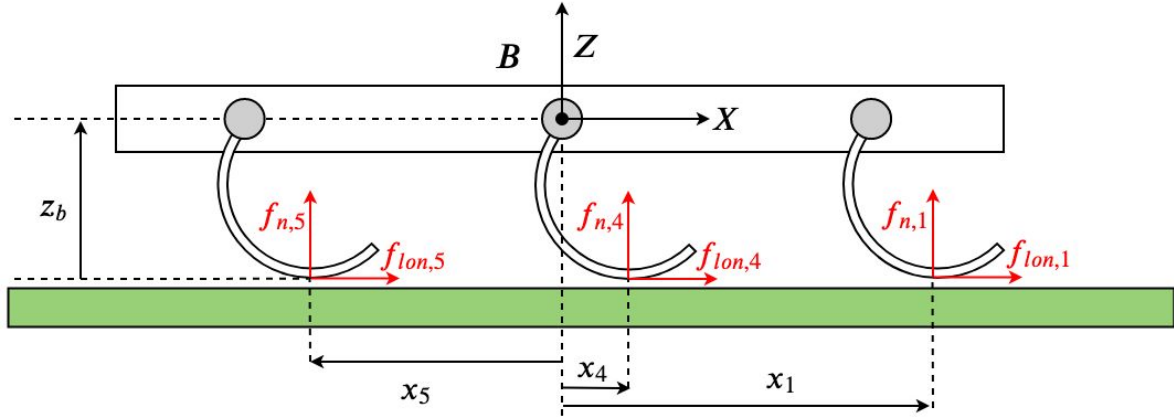


Figure 4-4: The contact forces are shown in red, along with the x coordinates in the B frame of the contact points. The forces clearly produce a pitching moment.

4-5 Slip model

Slip is essentially a velocity difference between two surfaces, resulting in a traction force from the friction. The slip between a Zebro's leg and the ground plays a huge role in determining the dynamics of the Zebro, and for this reason a separate Bachelor's thesis was tasked with performing an experiment which would help identify a model for the Zebro's slip. Their results are pivotal to this thesis. It has been mentioned in the previous chapter how $f_{lon} = P_{lon}(f_n)$ and $f_{lat} = P_{lat}(f_n)$. The aim of this chapter is to determine those functions P_{lon} and P_{lat} . First, the standard slip model used in most engineering applications, Pacejka's magic formula, is introduced in Section 4-5-1. This model is specifically for tyres, so in Section 4-5-2 an analogous version of Pacejka's parameter κ is found for the Zebro's half-circular leg. In the following sections the experiment that was conducted is introduced and the results and limitations of it are discussed.

4-5-1 Pacejka's magic formula and quantifying the slip of a tire

Two types of slip are described by Pacejka [24]: *longitudinal* and *lateral* slip. Longitudinal slip acts in the sagittal plane and can be imagined more colloquially as "wheel spin", whereas lateral slip is perpendicular to the sagittal plane and can be seen when a car is drifting around a corner. When slip occurs a traction force arises because of the friction between the wheel and the ground. Pacejka's magic formula is often used to estimate this traction force.

Magic formula

In many engineering cases, Pacejka's magic formula suffices for an estimation of the longitudinal and lateral traction forces present due to slip. For the longitudinal force a slip ratio κ is required which can be found in Section 4-5-1.

$$f_{lon} = D \sin C \arctan(B\kappa - E(B\kappa - \arctan(B\kappa))) \quad (4-23)$$

The formula for the lateral force is very similar, except a slip angle α is required this time. The angle is calculated in Section 4-5-1.

$$f_{lat} = D \sin C \arctan(B\alpha - E(B\alpha - \arctan(B\alpha))) \quad (4-24)$$

The parameters B, C, D and E can all be inferred from a slip curve which looks like the left image in Figure 4-5, using a step-by-step process [21]. On the right image it is shown how the parameters D, x_m and y_a are found in the curve. As for the rest of the parameters:

$$C = \pm \left(1 - \frac{2}{\pi} \arcsin \frac{y_a}{D} \right) \quad (4-25)$$

next, B can be found using C and the slope H at $\kappa = 0$.

$$B = \frac{H}{CD} \quad (4-26)$$

Lastly, E is found using B, C and x_m .

$$E = \frac{Bx_m - \tan(\frac{\pi}{2C})}{Bx_m - \arctan(Bx_m)} \quad (4-27)$$

After completing these steps, Eq. (4-23) and Eq. (4-24) can be used as a model for traction forces.

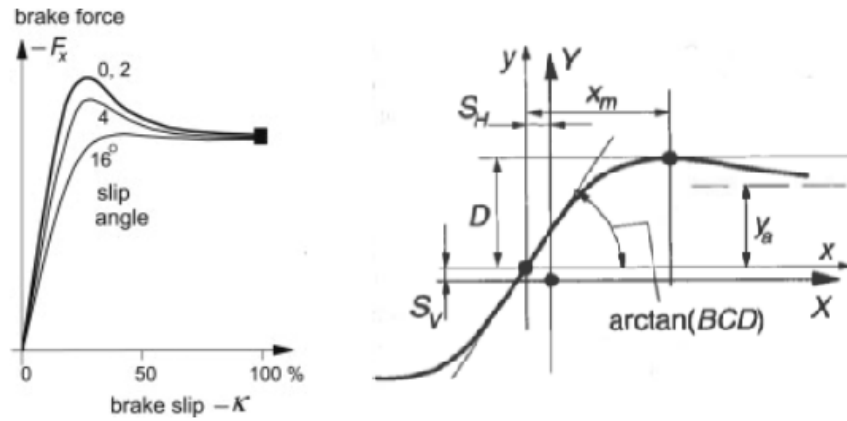


Figure 4-5: An example of a general slip curve for a tire. From [25].

Longitudinal slip

Figure 4-6 is used in [24] to define longitudinal slip. It shows a wheel of radius r and angular velocity Ω rolling over the ground. The wheel has a forward velocity horizontal to the ground of V_x . Point S is the point at which contact occurs and has a horizontal velocity V_{sx} shown in Eq. (4-28).

$$V_{sx} = V_x - r\Omega. \quad (4-28)$$

The longitudinal slip is quantified by the slip ratio, κ , and is defined as the ratio between the negative velocity of S and the forward velocity of the wheel.

$$\kappa = -\frac{V_{sx}}{V_x} = \frac{r\Omega - V_x}{V_x} \quad (4-29)$$

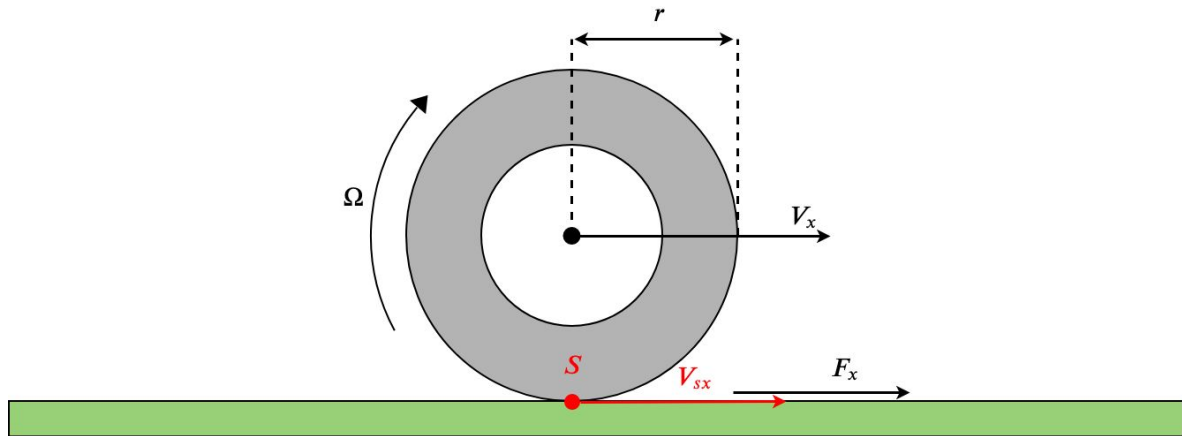


Figure 4-6: Side view diagram of a tire slipping longitudinally on a flat surface. Adapted from [24].

Lateral slip

Lateral slip is defined as the ratio of the lateral velocity of the contact point and its longitudinal (forward) speed. An image similar to Figure 4-7 was used in [24] to illustrate lateral slip. A rolling tire can be seen from above. If it was rolling without lateral slip, it would be moving in the x direction, however, the velocity of the wheel's centre, V_c , is clearly not on the x axis. This implies there is some sliding in the negative y direction and, as a result of friction, a traction force F_y .

The angle used to quantify lateral slip is α and can be calculated as in Eq. (4-29). Note that V_x is the same horizontal velocity shown in Figure 4-6.

$$\tan \alpha = -\frac{V_{sy}}{V_x} \quad (4-30)$$

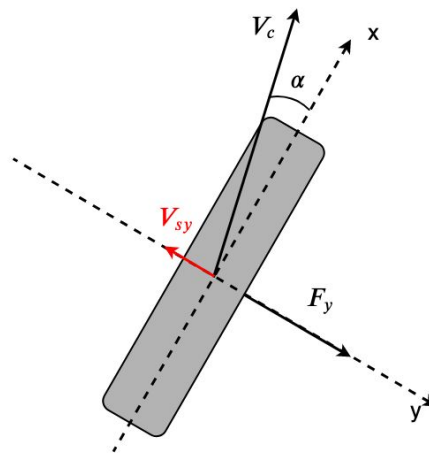


Figure 4-7: Bird's eye view diagram of a tire slipping laterally on a flat surface. Adapted from [24].

4-5-2 Quantifying the slip of a half-circular leg

Having shown how to calculate κ and α for a tire, the same must now be done for the Zebro's half-circular legs. Doing this results in two complications.

1. The velocity of the contact position is not as trivial to calculate as with a tire due to the leg's shape
2. Contrary to a tire, a leg in toe-stance has 0% slip if the contact position is stationary

In the following two sections both the rolling-contact and toe-contact states are considered.

Slip ratio in rolling stance

Two scenarios are shown in Figure 4-8. On the left, where $\kappa \approx 0$, a case in which almost no slip occurs is shown. The leg can be seen rolling over the ground normally. In the image on the right the case where $\kappa \rightarrow \infty$ is shown. Here the leg appears to remain almost stationary, despite the fact that it is rotating. One can deduce from these two scenarios that the quantifiable characteristic of slippage is the difference between the distance covered over the ground and the length of the arc covered over the leg by rolling. If the leg arc covered is equal to the distance covered over the ground then the leg is rolling perfectly without slip. If the leg arc covered is non-zero whereas the distance over the ground is 0, the leg is spinning over the ground without any traction.

To summarise: if $\kappa \approx 0$ then $x'_c - x_c \approx l_{arc}$ and if $\kappa \rightarrow \infty$ then $x'_c - x_c \rightarrow 0$, where the points x_c and x'_c are shown in Figure 4-8.

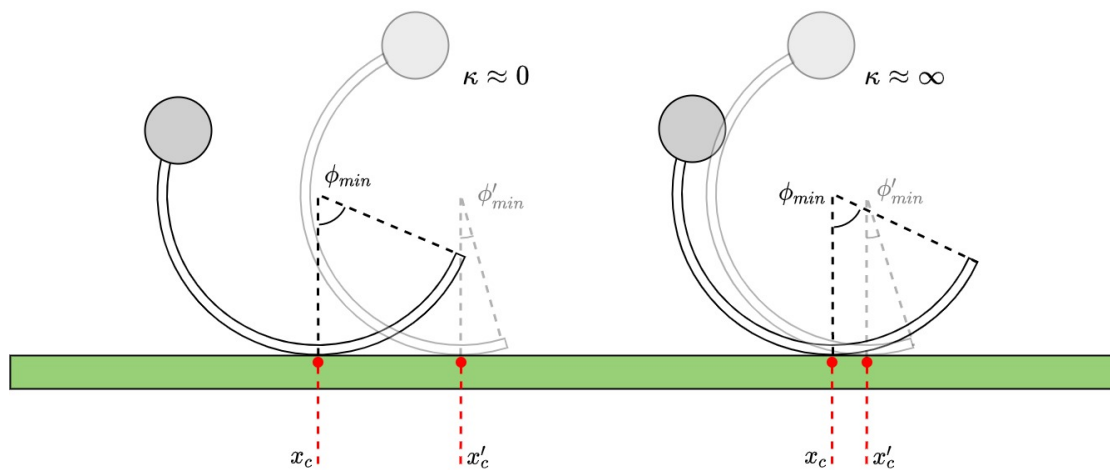


Figure 4-8: Scenarios $\kappa \approx 0$ and $\kappa \approx \infty$ are shown for a leg in rolling stance. The contact positions are marked in red and referred to as x_c .

Let the horizontal speed of the contact point be v_R and the speed at which the contact point moves along the ground be v_C . Respectively, these speeds are analogous to $r\Omega$ and V_x in

Eq. (4-29), resulting in the following slip ratio formula.

$$\kappa_R = \frac{v_R - v_C}{v_C} \quad (4-31)$$

To confirm, the two extreme cases can be checked: if $v_R = v_C$ then $\kappa = 0$ and if $v_C \rightarrow 0$ then $\kappa \rightarrow \infty$, aligning with the expectations from Figure 4-8.

Calculating the horizontal velocity of the leg's minimum requires some geometry to show that $\angle BCA = \phi_{min}/2$ and $\angle CAB = \phi_{min}/2$ in the left diagram in Figure 4-9. The distance from the hip to the contact is therefore $l = 2r \cos(\phi_{min}/2)$. With the leg rotating clockwise with velocity $-\dot{\phi}_{min}$, the horizontal speed of the leg's minimum (point C) is the following.

$$v_R = -\dot{\phi}_{min} l \cos(\phi_{min}/2) = -2\dot{\phi}_{min} r \cos^2(\phi_{min}/2) \quad (4-32)$$

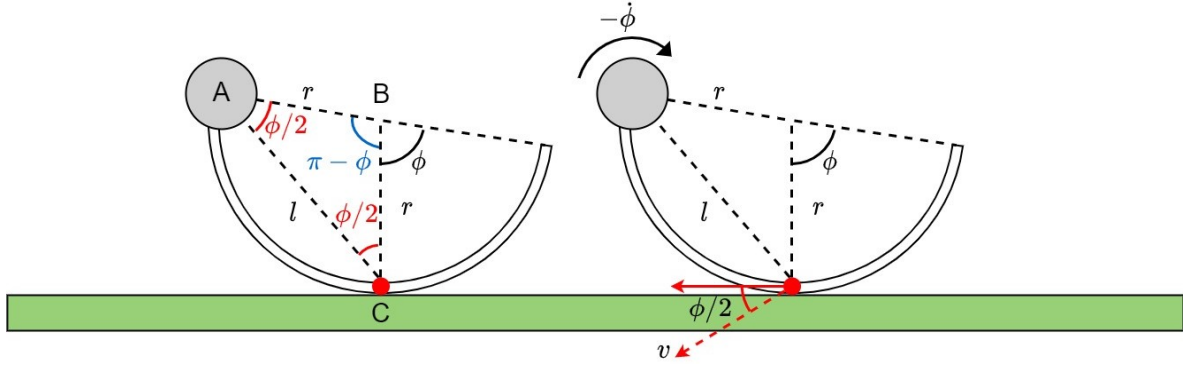


Figure 4-9: Triangle ABC is isosceles since two sides are of length r , therefore implying that $\angle BCA = \angle CAB$. Since $\angle ABC = \pi - \phi_{min}$ the other angles in the triangle must be $\phi_{min}/2$.

Recall that $\dot{\phi}_{min}$ is negative for forward walking, hence the negative expression in Eq. (4-32). The expression for $\dot{\phi}_{min}$ is given in Section 3-4-2

The value of v_C is found using the current linear and rotational velocity of the body as well as the position of the contact point in the inertial frame, ${}^I\mathbf{p}_c$. This position is calculated with the transformations found in Section 3-3.

$$\begin{bmatrix} {}^I\mathbf{p}_c \\ 1 \end{bmatrix} = H_B^I H_H^B H_C^H \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (4-33)$$

The velocity of this point is found in the following equations.

$${}^I\dot{\mathbf{p}}_c = {}^I\dot{\mathbf{p}}_b + \dot{\boldsymbol{\theta}}_b \times ({}^I\mathbf{p}_c - {}^I\mathbf{p}_b) \quad (4-34)$$

Since v_R is expressed in the C -frame, the same should be true for v_C . This is done by rotating with a matrix R_I^C .

$$\begin{bmatrix} {}^C\dot{x}_c \\ {}^C\dot{y}_c \\ {}^C\dot{z}_c \end{bmatrix} = R_I^{CI} \dot{\mathbf{p}}_c \quad (4-35)$$

Where $v_C = {}^C\dot{x}_c$ and $R_I^C = (R_B^I R_H^B R_C^H)^{-1}$ as found in Section 3-3.

Slip ratio in toe stance

The toe stance case will now be considered. See Figure 4-10 for a diagram of the two important scenarios. If there is almost no slip ($\kappa \approx 0$) one would expect the toe to remain almost stationary as the hip rotates around it. On the other hand, if there is a lot of slip, one would expect the toe to be sliding backwards and the hip to be moving almost vertically downwards. This poses a slight problem since this scenario is not comparable to a slipping tire in which the exact opposite is true: a stationary contact point usually means $\kappa \rightarrow \infty$.

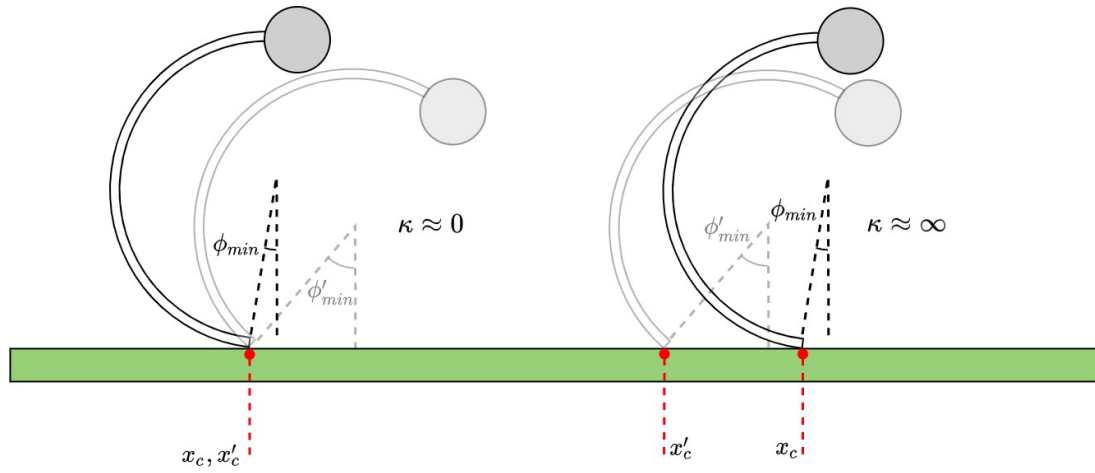


Figure 4-10: Scenarios $\kappa \approx 0$ and $\kappa \approx \infty$ are shown for a leg in toe stance. The contact positions are marked in red and referred to as x_c .

Two conditions can be used to deduce a valid heuristic for κ . The speed v_H is the horizontal speed of the leg's hip, whereas v_T is the horizontal speed of the toe due to the leg's rotation. See Figure 4-11 for an illustration.

1. As $v_H \rightarrow 0$, $\kappa \rightarrow \infty$
2. As $(v_T - v_H) \rightarrow 0$, $\kappa \rightarrow 0$

An expression for κ which satisfies both conditions is given in Eq. (4-36).

$$\kappa_T = \frac{v_T - v_H}{v_H} \quad (4-36)$$

Consider the leg floating in space rather than touching the ground. It is rotating about the hip with speed $-\dot{\phi}_{min}$, which results in a tangential velocity magnitude of $-2r\dot{\phi}_{min}$. In the horizontal direction this gives the following result for v_T .

$$v_T = -2r\dot{\phi}_{min} \cos(\phi_{min}) \quad (4-37)$$

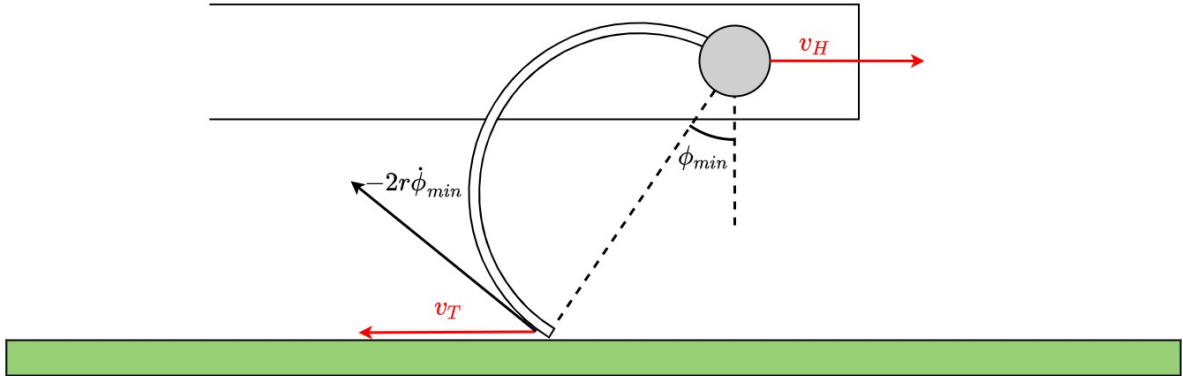


Figure 4-11: Horizontal contact velocity and body velocity shown for the toe stance. For 0% slip, the two should cancel each other out such that the toe is stationary.

The hip's horizontal velocity v_H is found using the transformations from Section 3-3. Since the Zebro's hips are attached in a rigid body, their velocities are simply the summation of body's velocity itself and a rotational component.

$${}^I\dot{\mathbf{p}}_h = {}^I\dot{\mathbf{p}}_b + \dot{\boldsymbol{\theta}}_b \times ({}^I\mathbf{p}_h - {}^I\mathbf{p}_b) \quad (4-38)$$

Or, written out as matrices:

$$\begin{bmatrix} {}^I\dot{x}_h \\ {}^I\dot{y}_h \\ {}^I\dot{z}_h \end{bmatrix} = \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{z}_b \end{bmatrix} + \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_p \\ \dot{\theta}_y \end{bmatrix} \times \left(\begin{bmatrix} {}^Ix_h \\ {}^Iy_h \\ {}^Iz_h \end{bmatrix} - \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \right). \quad (4-39)$$

Finally, notice that v_T is purely in the x direction of the C -frame, so v_H should also be expressed in this same frame.

$$\begin{bmatrix} {}^C\dot{x}_h \\ {}^C\dot{y}_h \\ {}^C\dot{z}_h \end{bmatrix} = R_I^C \begin{bmatrix} {}^I\dot{x}_h \\ {}^I\dot{y}_h \\ {}^I\dot{z}_h \end{bmatrix} \quad (4-40)$$

Where $v_H = {}^C\dot{x}_h$ and $R_I^C = (R_B^I R_H^B R_C^H)^{-1}$ as found in Section 3-3.

4-5-3 Model for the lateral traction force

The slip angle α is calculated exactly the same as in Pacejka's model, so

$$\tan \alpha = \frac{{}^C\dot{y}_c}{{}^C\dot{x}_c} \quad (4-41)$$

with ${}^C\dot{x}_c$ and ${}^C\dot{y}_c$ found as in Eq. (4-35). However, during the model identifying stage, no experiments were performed to determine the lateral slip model. To solve this problem it was decided to use the general magic formula for longitudinal slip, but replace κ with a function of α to give a realistic result.

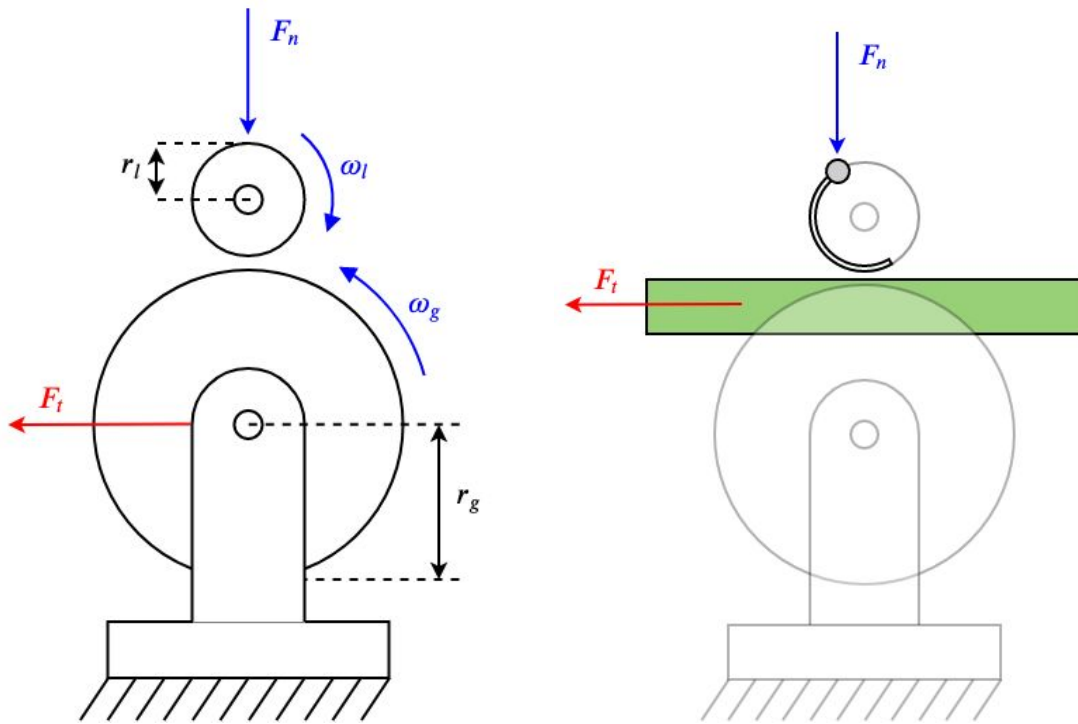
A heuristic which was deemed valid is to compare the lateral slip to longitudinal slip by assuming a slip angle $\alpha = \pi/2$ is equivalent to a slip ratio of $\kappa = 1$ and a slip angle of $\alpha = 0$ is equivalent to a slip ratio of $\kappa = 0$, resulting in the following formula for the lateral slip ratio κ_α .

$$\kappa_\alpha = \frac{\alpha}{\pi/2} \quad (4-42)$$

This value is then used in the general magic formula from Eq. (4-23) to estimate the lateral traction force.

4-6 Experimental setup for determining the longitudinal slip curve

The experimental setup used was designed in conjunction with a separate research group [21]. The experiment aimed to model a leg rolling over the ground, whilst measuring the traction force produced by this action. Two rotating discs of differing diameters and materials were used; the smaller disc represents a leg and is made of the same material as the Zebro's leg, and the larger disc represents the ground and has an outer layer of the same material as the ground. The ground-disc is given a radius significantly larger than the leg-disc to reduce the curvature at the contact point and simulate a flat surface [21]. Figure 4-12 shows a simple diagram of the experiment. On the left the two discs are shown, whereas on the right their physical representations are depicted.



(a) A vertical depiction of the test setup showing the leg disc on top and the ground disc on the bottom. (b) A physical representation of the setup shown in Figure 4-12a.

Figure 4-12: A side-by-side comparison of the test setup and the physical scenario it is supposed to represent.

The actual way in which the experiment was built and conducted can be seen in Figure 4-13. The two discs are mounted to a base and a threaded shaft is used to push the “leg” into the “ground”, simulating the weight of the Zebro. Results are obtained for a range of different weights.

The goal of the experiment is to plot the traction force against the slip ratio, producing the well-known slip curve. In order to achieve this, the slip ratio is varied between 0 and 1 by controlling the angular velocities of the discs whilst the forces on the “ground” are being measured using a Schunk FTS-Delta SI-330-30 load cell. The motors are kept at a constant speed with a simple PD controller.

Let the angular velocity of the smaller disc be ω_l and the angular velocity of the larger disc be ω_g . They have radius r_l and r_g respectively. The linear velocities of the two discs at the contact point are

$$v_l = \omega_l r_l, \quad v_g = \omega_g r_g \quad (4-43)$$

making the slip ratio at the contact point

$$\kappa = \frac{\omega_l r_l - \omega_g r_g}{\omega_l r_l}. \quad (4-44)$$

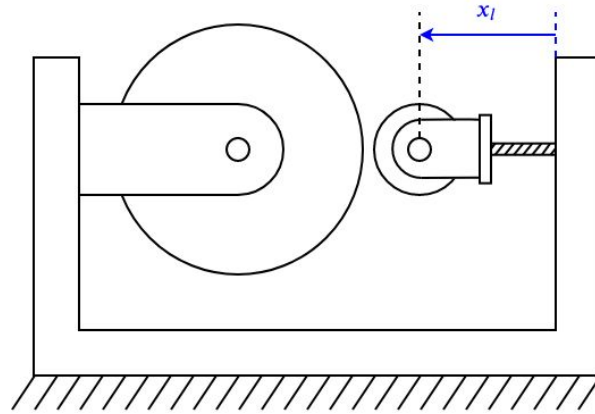


Figure 4-13: A schematic diagram of the experiment setup in the configuration in which it was manufactured.

4-6-1 Results of the experiment

Four datasets have been plotted in Figure 4-14, each with a different normal force F_n . The top-left plot is for $F_n = 2.13$ N, top-right is for $F_n = 6.63$ N, bottom-left is with $F_n = 8.13$ N and finally, the bottom-right plot is for $F_n = 12.75$ N. Using these results the following values for B and C were calculated using the method outlined in Section 4-5-1.

$$B = 34.34 \text{ and } C = 1.564 \quad (4-45)$$

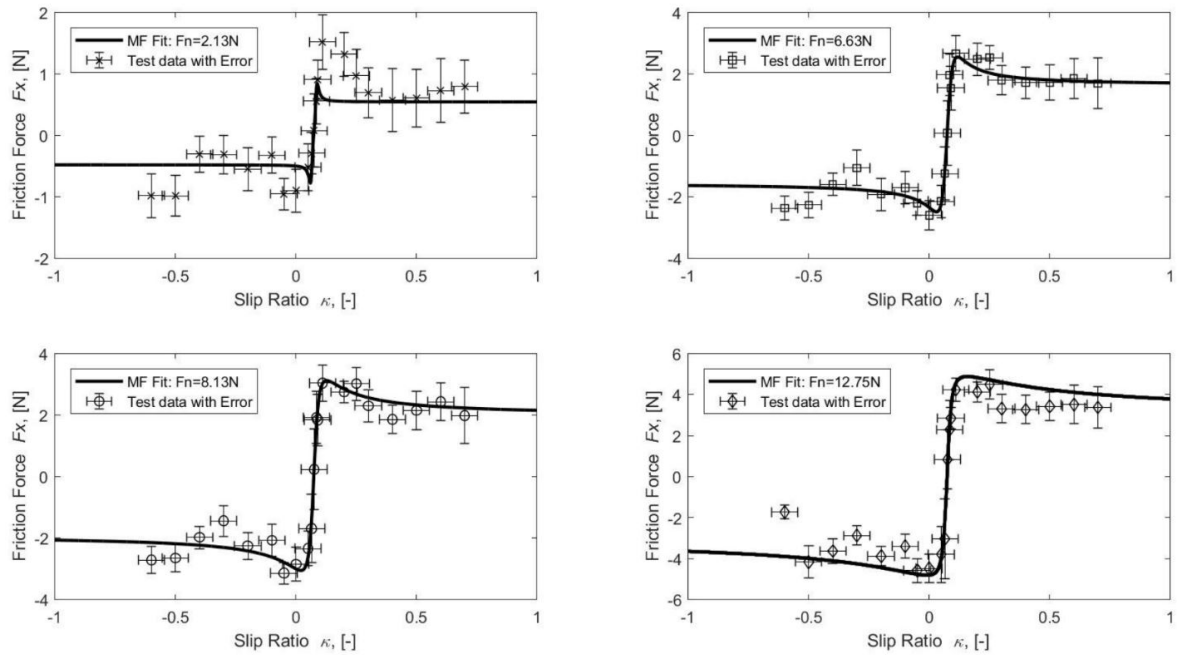


Figure 4-14: Four different experiments performed. Top-left: $F_n = 2.13$ N, top-right: $F_n = 6.63$ N, bottom-left: $F_n = 8.13$ N and bottom-right: $F_n = 12.75$ N.

4-6-2 Limitations of the experiment

A noticeable problem with the experimental setup was the imperfect circular shape of the discs, most notably the ground-disc, which had a visible bump despite having to simulate a flat ground. This bump caused significant fluctuations in F_n which in turn caused fluctuations in F_t , making the results unreliable.

To make use of this problem and turn it into an advantage, an experiment was conducted in which κ was kept as constant as possible, and F_n and F_t were recorded and plotted against time. This meant that the fluctuations in F_n and F_t did not affect the result, since they are both being recorded. Doing this for several different slip ratios yielded a lot of data which was filtered and plotted in order to create a three-dimensional slip curve.

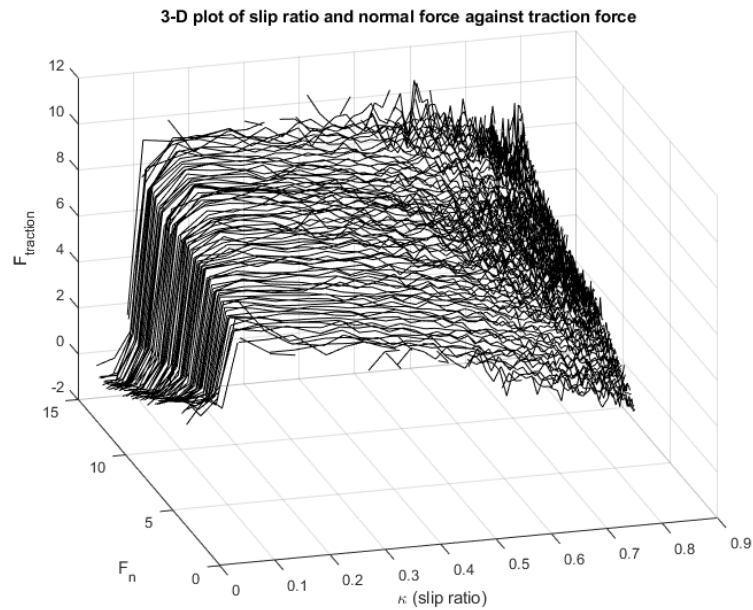


Figure 4-15: Three dimensional plot of the normal force against traction force for 62 separate values for κ .

It turns out that when plotting the curve produced by Pacejka's magic formula provided in [21], the results in Figure 4-15 do not match well with the model, casting significant doubt on the identified model.

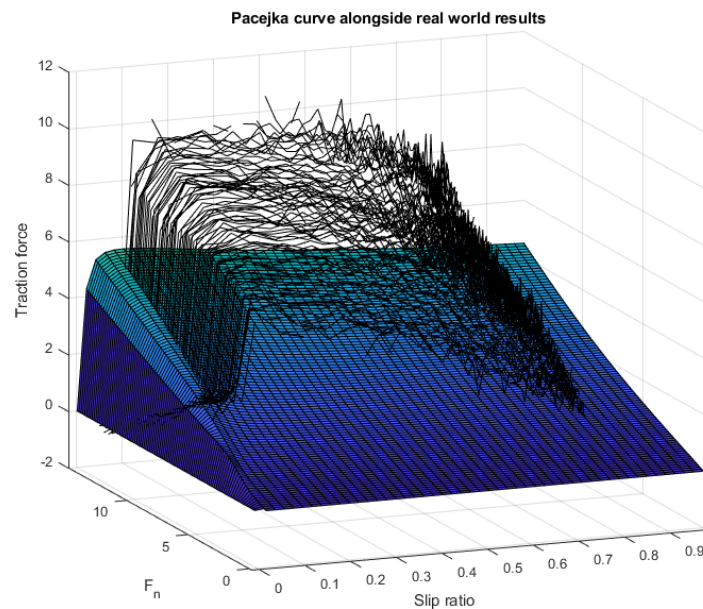


Figure 4-16: The independently found data compared with the previously identified magic formula model.

An important feature which was missing from the experiment was the possibility to record the velocities of the discs. Due to the fluctuations from the bump, the velocities would vary a lot, making the calculated value of the slip ratio unreliable. If the velocities were recorded, one could calculate the slip ratio instantaneously and plot this against the traction force, giving much more accurate results.

4-7 Summary

The following block diagram should give a concise overview of how the components outlined within this chapter should interact with each other to form a complete model of the system. The blocks *Contact Detection* and *Kinematics* were discussed in Chapter 3, but the others are mentioned in this chapter.

To summarise, kinematic equations of the body are applied to find the velocities of certain points on the legs in \mathcal{C} which are used to find the longitudinal and lateral slip parameters κ and κ_α . Two different formulas for the leg's longitudinal slip ratio are applicable. Which one is chosen depends on the leg's state of either rolling-contact or toe-contact. In rolling-contact, the formula for κ_R , as found in Eq. (4-31), is used, whereas in toe-contact, the formula for κ_T is used.

The vertical acceleration of the body is also found kinematically and inverse dynamics are used to estimate the vertical contact forces at the contact points which would result in this acceleration (F_c from the *Inverse Dynamics* block). These contact forces were initially estimated using a weight distribution model in Section 4-3 and then optimised such that the forces would produce a vertical motion equal to the kinematically determined vertical acceleration.

This contact force estimation, along with the slip parameters, results in the traction force at the contact point (F_t from the *Traction Model* block) using Pacejka's magic formula in Section 4-5-1 whose parameters were found experimentally using the setup in Section 4-6. Both the vertical contact force and the traction force are used in the *Forward Dynamics* block to produce the generalised accelerations of the body, $\ddot{\mathbf{q}}$, which is the final result of the model.

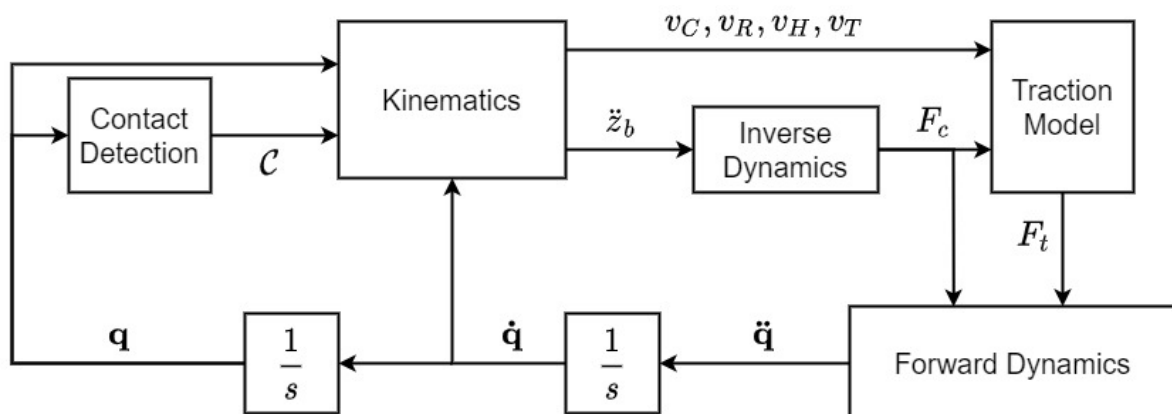


Figure 4-17: Block diagram containing each component of the model.

Chapter 5

Simulations

5-1 Introduction

In this chapter, Section 5-2 shows the approach used to apply Suriana’s max-plus gait to the dynamic model. The new turning algorithm from Section 2-6 is also modelled using the same dynamic model, showing a perfectly circular path in Section 5-3. It should be mentioned that due to current events at the time of writing, regarding COVID-19, any experiments to validate the model are unfortunately not possible. This will be left for further research. Instead, a more qualitative approach to experimentation is taken to give a rough idea of the quality of the results in Section 5-4, using photographs taken of the Zebro in motion.

To conclude this chapter, in Section 5-5 a first look is taken at modelling the more complicated tetrapod gait, which has been difficult to model due to the complicated “wobbling” motion observed when walking. This motion is modelled using kinematics, paving the way for a future dynamic model of walking in the tetrapod gait.

5-2 Simulating the max-plus algorithm

Suriana’s switching max-plus algorithm was not available, so a different approach to model his gait was required. Following the theory in his thesis, a similar gait was designed, without the use of max-plus.

5-2-1 Modelling the leg speeds

The three parameters used in the max-plus algorithm were τ_g , τ_f and τ_p . The parameter τ_p is used to control how much the Zebro steers. The legs on the inner side spend $\tau_g + \tau_p$ seconds on the ground and τ_f seconds in the air, whereas the legs on the outside spend τ_g seconds on the ground and $\tau_f + \tau_p$ seconds in the air.

Using the touchdown and lift-off angles that Suriana chose, the following formulas for the inner and outer leg velocities can be used to roughly emulate his max-plus algorithm.

$$\dot{\theta}_g^i = \frac{\theta_{LO} - \theta_{TD}}{\tau_g + \tau_p} \quad (5-1)$$

$$\dot{\theta}_g^o = \frac{\theta_{LO} - \theta_{TD}}{\tau_g} \quad (5-2)$$

$$\dot{\theta}_f^i = \frac{2\pi - (\theta_{LO} - \theta_{TD})}{\tau_f} \quad (5-3)$$

$$\dot{\theta}_f^o = \frac{2\pi - (\theta_{LO} - \theta_{TD})}{\tau_f + \tau_p} \quad (5-4)$$

These leg speeds were implemented in the model, with legs switching from *flight* to *ground* speed when contact is detected, and vice versa.

5-2-2 Simulating straightforward walking

Five seconds of straightforward walking, with $\tau_p = 0$ were simulated. The results in Figure 5-1 show the expected straight line for the path and a total distance covered of roughly one metre. Unfortunately, there is no way to validate this, but watching the Zebro walk for five seconds gives the impression that this could be realistic.

Taking a look at Figure 5-2, some behaviours which are to be expected can be confirmed. For example, the velocity of the body in the x direction shows a sort of asymmetrical parabola with maximums at the point where the legs are at angle $\pi/2$, and the largest x velocity is to be expected. Again, this cannot be confirmed experimentally, but analytically this makes sense.

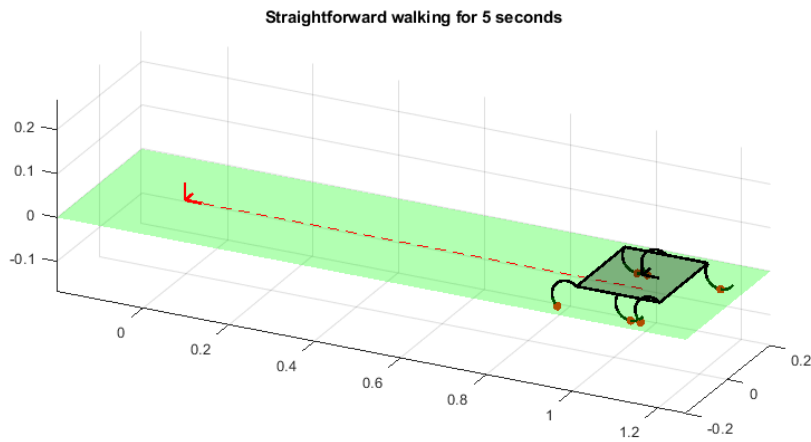


Figure 5-1: Five seconds of straightforward walking using $\tau_g = 0.4$ and $\tau_f = 0.3$

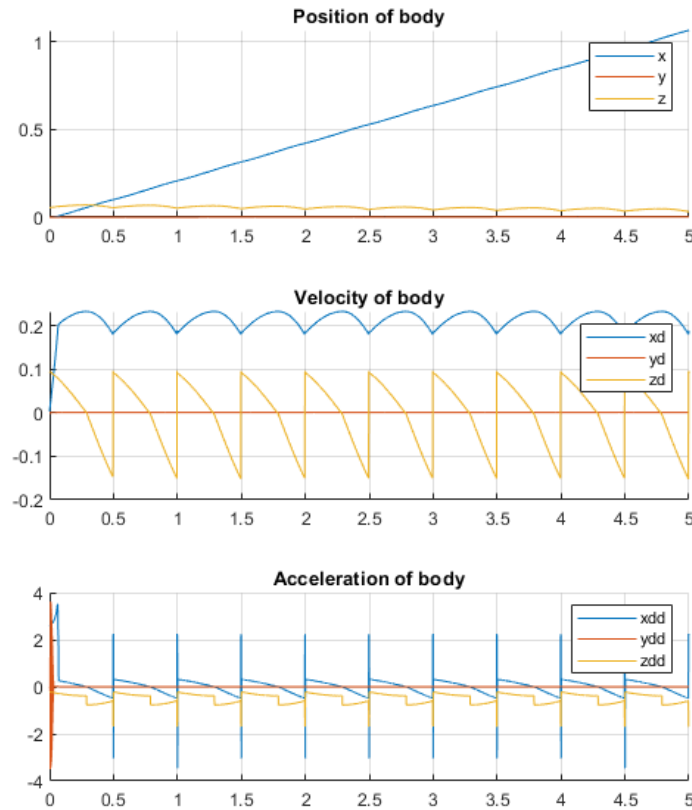


Figure 5-2: The position, velocity and acceleration of the body during 5 seconds of straightforward walking.

5-2-3 Comparison with real world experiment

In order to perform realistic experiments with the Zebro it was important to choose the right values for the the touchdown and lift-off angles. Recall that the lift-off angle is used as a free-to-choose input to the Zebro software, despite it being a physically determined parameter. Finding the ideal lift-off angle which results in touchdown and lift-off at the same time was an iterative process of measuring the hip height at certain angles, until the hip height at touchdown matches the hip height at lift-off.

The touchdown angle was kept the same for both the simulation and the actual robot at 50° . Eq. (2-17) results in a theoretical lift-off angle of 118° which served as an initial guess for the actual measured lift-off angle. This turned out to be 130° . The reason for the disparity between the theoretical value and the actual value is the fact that the Zebro's leg is slightly more than a half-circle, actually spanning an angle of roughly 200° , rather than the 180° which has been assumed for the model. This effect is displayed in Figure 5-9.

It was noted by Suriana that the middle left leg's angle was smaller by five degrees than expected [10], so an offset array was introduced in the software to ensure that the desired

position sent to the motor can be calibrated.

The Zebro was placed at a starting point and was made to walk in a straight line with $\tau_g = \tau_f = 1$. A set of photographs was merged together to give an indication of the robot's path during the experiment.

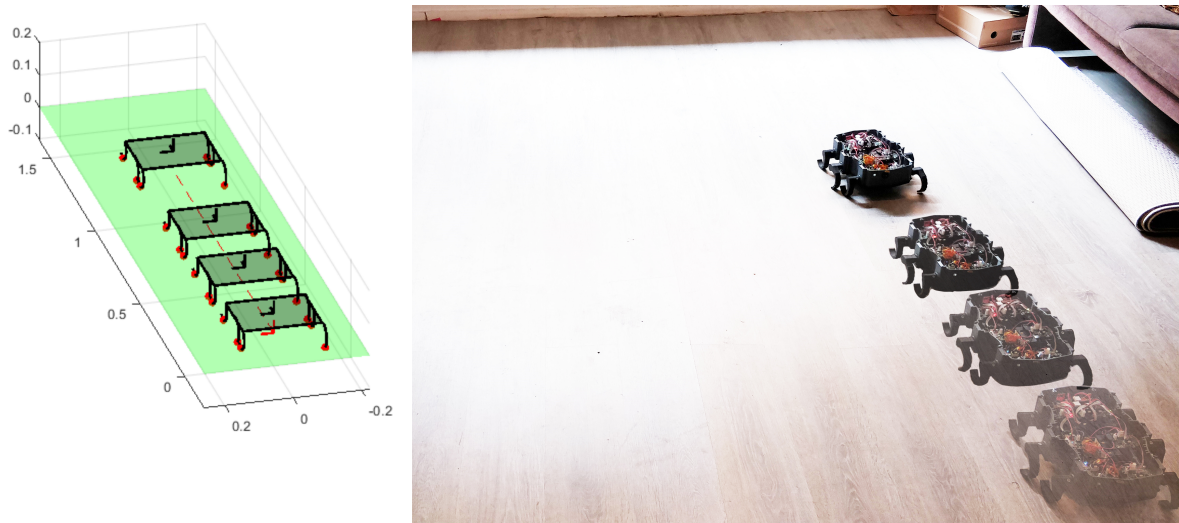


Figure 5-3: Simulated straightforward walking with $\tau_g = 1$ and $\tau_f = 1$ next to actual results. Snapshots were taken at $t = 0s, 4s, 8s, 14s$.

5-2-4 Simulating turning

Using the parameters $\tau_p = 0.1$, $\tau_g = 0.8$ and $\tau_f = 0.8$, as well as a touchdown angle $\theta_{TD} = 35^\circ$ and lift-off angle $\theta_{LO} = 130^\circ$, 4.8 seconds of turning was simulated and produced a clear turn to the left as expected. Unfortunately, due to problems in the Zebro's software, this test was not performed on the actual robot.

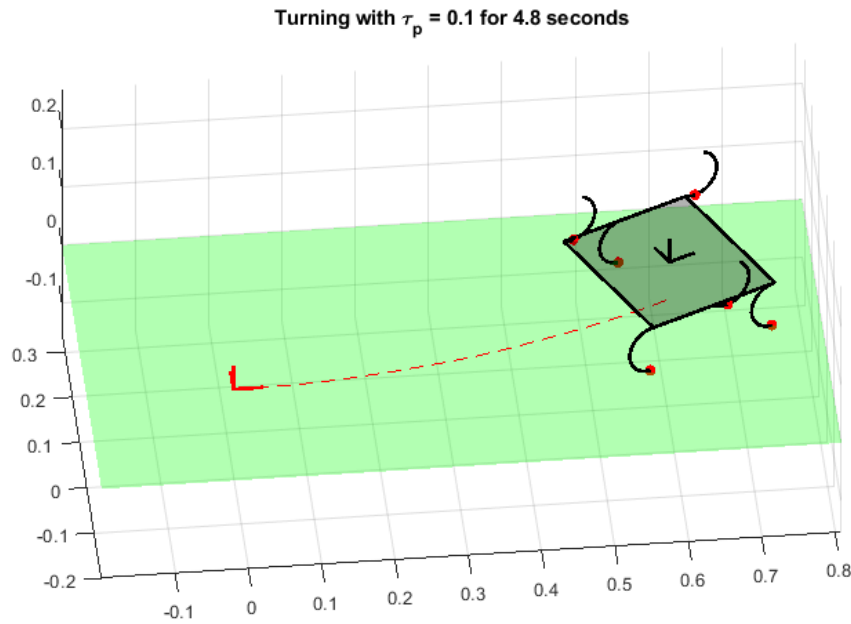


Figure 5-4: 4.8 seconds of turning using $\tau_p = 0.1$, $\tau_g = 0.8$ and $\tau_f = 0.8$

The generalised coordinates x , y , z , θ_r , θ_p and θ_y and their speeds were recorded and plotted against time, resulting in the graphs shown in Figure 5-5. Important to notice here is the strange behaviour when a touchdown or lift-off event occurs causing the robot to accelerate unexpectedly. This can be seen very clearly in the graph of the yaw angle. This result implies that due to the unsmooth contact transition in the max-plus gait, the model handles transition incorrectly. It is therefore important to test the new turning algorithm since it guarantees a smooth contact transition.

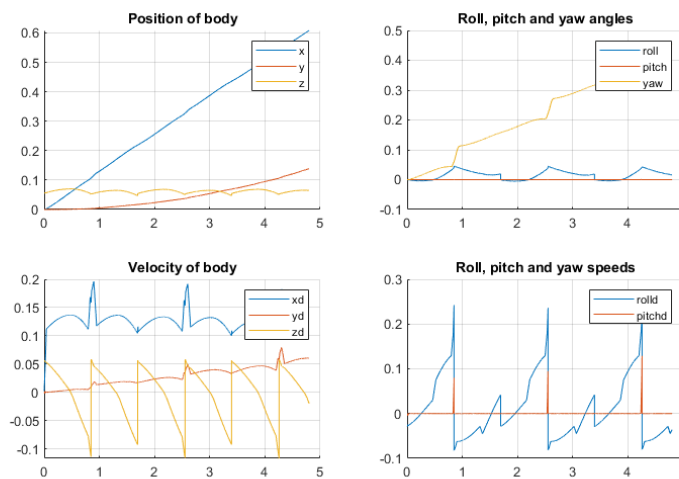


Figure 5-5: The position and velocity of the body, as well as the orientation plotted against time during 4.8 seconds of walking.

5-2-5 Key insights

A number of key insights were noted. Firstly, the contact state transitions occur as expected, with the touchdown of one leg group coinciding exactly with the lift-off of the other leg group, resulting in a smooth motion. That being said, in the Zebro's software the leg angle is only accurate to 3.6° , since the angle is defined by an integer from 0 to 100. This leaves room for improvement when prescribing the touchdown and lift-off angles.

A second point is the distance travelled along the ground. The simulated image and the photograph show a remarkably similar distances travelled between each snapshot. This is likely a lucky coincidence, since the slip model was not identified using the same floor material as in the photograph, but it does give an indication of promising results.

5-3 Simulating the novel turning algorithm

Using the kinematics outlined in Section 3-5-6, estimates for $\dot{\theta}_r$ and \ddot{z}_b can be found and used in the dynamic model of the system in order to model the contact forces. The results of these simulations show a very smooth walking behaviour, which makes sense since this turning algorithm is designed with the smooth contact transitions in mind. The path produced is a near perfect circle and, despite not being able to collect validation data, a simple test of the algorithm with a real Zebro showed very similar results.

5-3-1 Simulating 4 seconds of walking with steering factor 1.2

A smooth left turn is expected with $s = 1.2$. This simulation uses $\tau_c = 1$ and $\theta_{TD}^i = \pi/4$ rad resulting in the following characteristic parameters.

Parameter	Value
θ_{TD}^i	$\pi/4$ rad
θ_{LO}^i	2.1188 rad
θ_{TD}^o	0.6259 rad
θ_{LO}^o	2.2260 rad

Table 5-1: Characteristic leg angles for steering factor 1.2 and inner TD angle $\pi/4$.

Figure 5-6 shows the path and final orientation after 4 seconds. As expected, a left turn is performed. For a better understanding of the steps taken by the Zebro along this path, the CoM's position, velocity and acceleration during one whole cycle are plotted in the left column of Figure 5-7. Similarly, the roll, pitch and yaw angles are plotted in the right column of Figure 5-7.

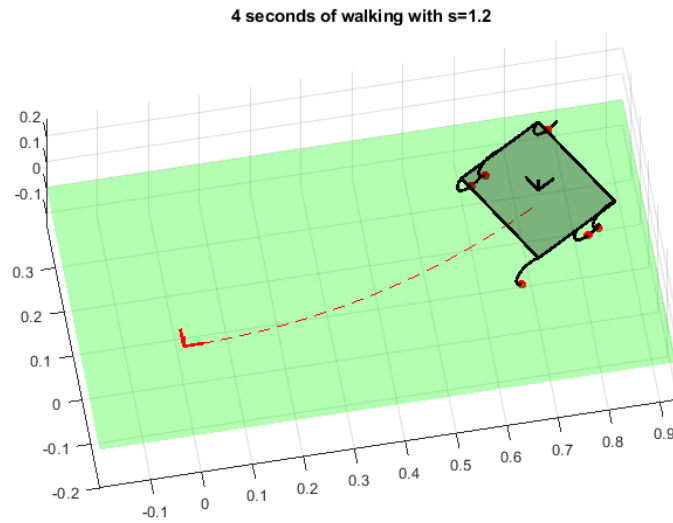


Figure 5-6: 4 seconds of walking with steering factor 1.2.

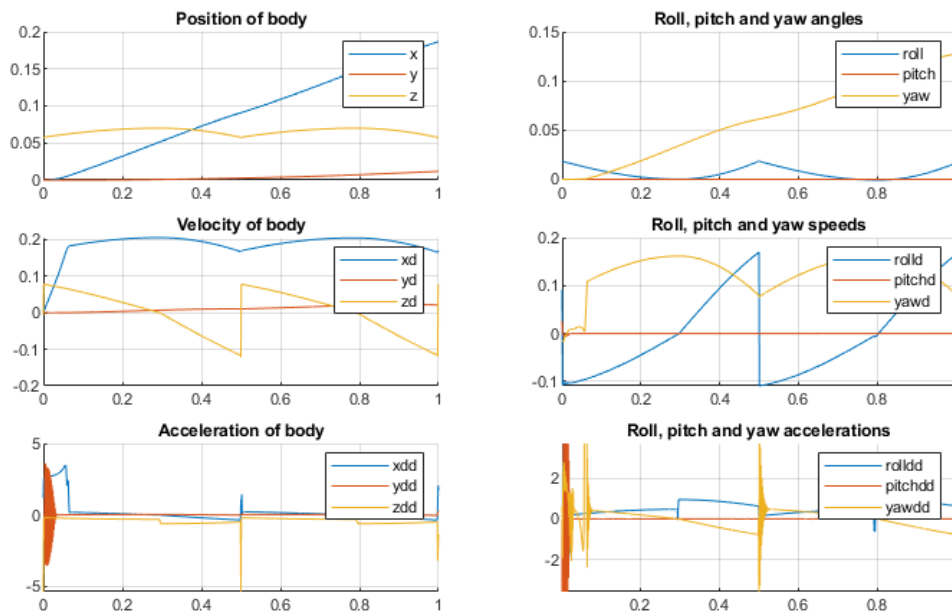


Figure 5-7: Left: Positions, velocities and accelerations of the body's centre of mass in each direction. Right: Roll, pitch and yaw angles, speeds and accelerations.

The results for position, velocity and acceleration look very similar to the results for straight-forward walking, with the main difference being the y position increasing due to the fact that the Zebro turns to the left. It is also interesting to look at the roll angle of the Zebro showing the expected result of being “slanted” at the time of touchdown and then gradually straightening out during the step. The oscillating result in the accelerations is caused by the

model's slip ratio oscillating and causing the Zebro to continuously accelerate and decelerate during some parts of a step. More info on this effect is given in Section 5-4-4.

5-3-2 Simulating 40 seconds of walking with steering factor 1.25

Another simulation which illustrates the working model is where the Zebro walks in a full circle and returns to roughly the same position as where it started. The chosen inner touchdown angle for this simulation was $\theta_{TD}^i = \pi/4$ rad, resulting in the following parameters.

Parameter	Value
θ_{TD}^i	$\pi/4$ rad
θ_{LO}^i	2.1188 rad
θ_{TD}^o	0.5858 rad
θ_{LO}^o	2.2526 rad

Table 5-2: Characteristic leg angles for steering factor 1.25 and inner TD angle $\pi/4$.

A noticeable problem in the model is that the Zebro gradually descends through the ground with every step. One can see in Figure 5-8 that at the end of the simulation the Zebro is fully “underground”. This is obviously not correct, and the problem is attributed to the contact state transitions occurring slightly too late so that the legs appear to be a few millimetres lower than they should be at the actual impact time. The effect is negligible for a few steps, but for a simulation lasting more than 10 seconds the Zebro can clearly be seen sinking into the ground. This problem can be worked around by resetting the body's z -coordinate at every contact detection.

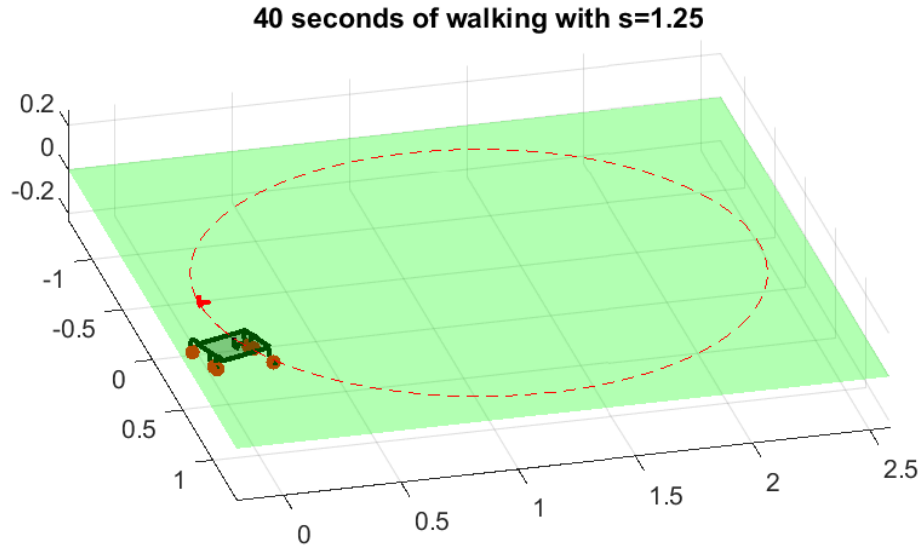


Figure 5-8: 40 seconds of walking with steering factor 1.25.

5-4 Experimental results of novel turning algorithm

To test whether the simulations gave realistic results, the Zebro's software was altered to allow for differing touchdown and lift-off angles on each side of the robot. These changes were made in the file `ZebroLeg.cpp`. A normal tripod gait was used for the timing since the rhythm is the same as a straightforward tripod gait.

The method for finding the touchdown and lift-off parameters of the Zebro was by first noting the theoretical parameters calculated in the simulation, and then iteratively rotating the Zebro's leg until the hip heights at lift-off and touchdown were equal. The angles found with the Zebro turned out to be significantly different to those calculated, as can be seen in Table 5-3. The reason for this, as mentioned in Section 5-2-3 is that the real Zebro leg has an arc angle of about 200° whereas the modelled Zebro has an arc angle of 180° .

5-4-1 Steering factor 1.1

The characteristic angles for a simulated steering factor of 1.1 are given in the table below. The illustration in Figure 5-9 shows how the difference between the simulated angles and the real angles comes about.

Parameter	Value (simulation)	Value (real)
θ_{TD}^i	50.0 deg	50.0 deg
θ_{LO}^i	118.0 deg	130.0 deg
θ_{TD}^o	46.0 deg	46.0 deg
θ_{LO}^o	120.7 deg	132.0 deg

Table 5-3: Characteristic angles for a steering factor of 1.1. The second column displays the angles calculated in the model whereas the third column shows the actual angles found.

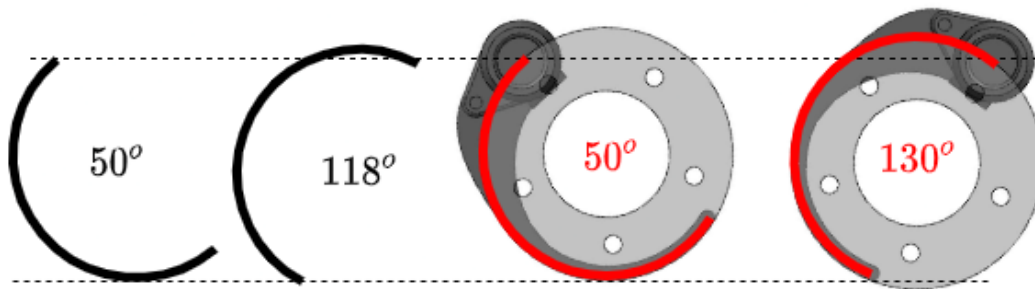


Figure 5-9: An illustration showing the angle differences between the actual Zebro and the modelled Zebro.

In Figure 5-10 the Zebro can be seen walking in a curve to the left. Comparing this to the next image in Figure 5-11, it can be seen that the model gives a very good estimation of the curve and distance travelled. A snapshot was taken every 5 seconds and each position corresponds well with the model.



Figure 5-10: A sequence of photographs taken while the Zebro was walking with steering factor 1.1 and merged together.

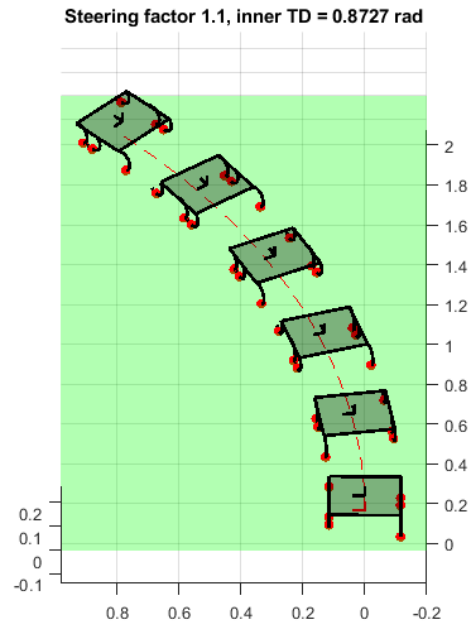


Figure 5-11: Simulation of steering factor 1.1 and $\theta_{TD}^i = 50^\circ$. Snapshots were taken at $t = 0s$, 5s, 10s, 15s, 20s and 25s.

5-4-2 Steering factor 1.3

Similarly to the previous experiment, the touchdown and lift-off angles for $s = 1.3$ were found and are given in Table 5-4.

Parameter	Value (simulation)	Value (real)
θ_{TD}^i	50.0 deg	50.0 deg
θ_{LO}^i	118.0 deg	130.0 deg
θ_{TD}^o	37.8 deg	37.8 deg
θ_{LO}^o	126.2 deg	138.0 deg

Table 5-4: Characteristic angles for a steering factor of 1.3. The second column displays the angles calculated in the model whereas the third column shows the actual angles found.

The Zebro was made to walk for 30 seconds with these parameters and the resulting path is shown in Figure 5-12. The same is done in the model and its resulting path is given in Figure 5-13.



Figure 5-12: A sequence of photographs taken while the Zebro was walking with steering factor 1.3 and merged together.

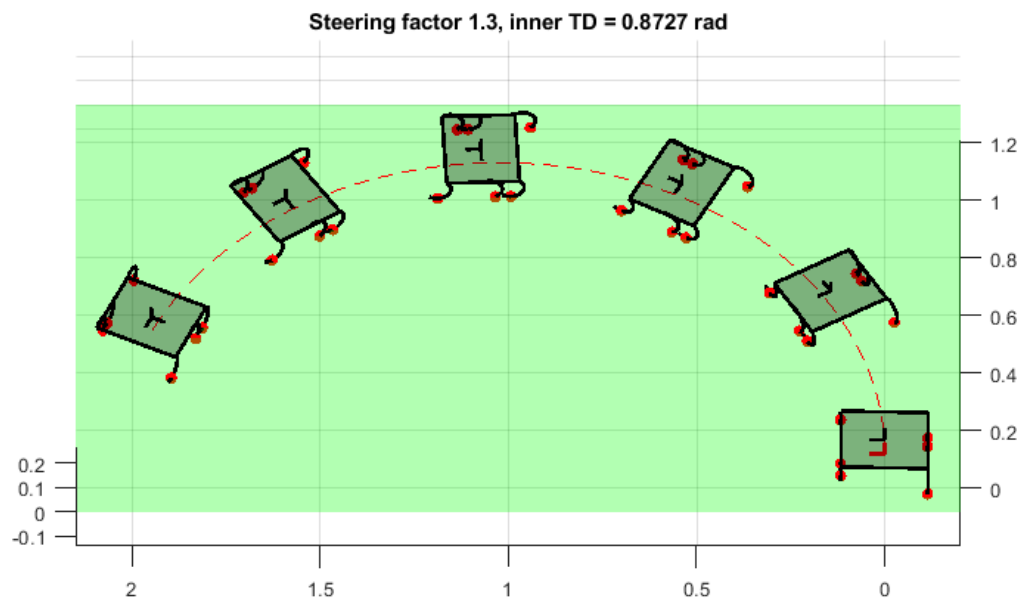


Figure 5-13: Simulation of steering factor 1.3 and $\theta_{TD}^i = 50^\circ$. Snapshots were taken at $t = 0s$, $6s$, $18s$, $24s$ and $30s$.

Unlike in the previous experiment, these results appear a lot less impressive. In the model a much tighter turning circle is expected which is completely unlike the actual results. A likely reason for this disparity is the fact that the Zebro's legs slip more on the laminated floor than on the modelled material. The reasoning for these bad results is explored further in Section 5-4-4.

5-4-3 Steering factor 1.5

The steering factor was increased to 1.5 and the experiment was repeated with a simulation time of 24 seconds. As expected, the Zebro makes a tighter curve, but again, just as with $s = 1.3$, the turning circle in the simulation is much smaller than in reality.

Parameter	Value (simulation)	Value (real)
θ_{TD}^i	50.0 deg	50.0 deg
θ_{LO}^i	118.0 deg	130.0 deg
θ_{TD}^o	29.7 deg	29.7 deg
θ_{LO}^o	131.6 deg	143.0 deg

Table 5-5: Characteristic angles for a steering factor of 1.5. The second column displays the angles calculated in the model whereas the third column shows the actual angles found.



Figure 5-14: A sequence of photographs taken while the Zebro was walking with steering factor 1.5 and merged together.

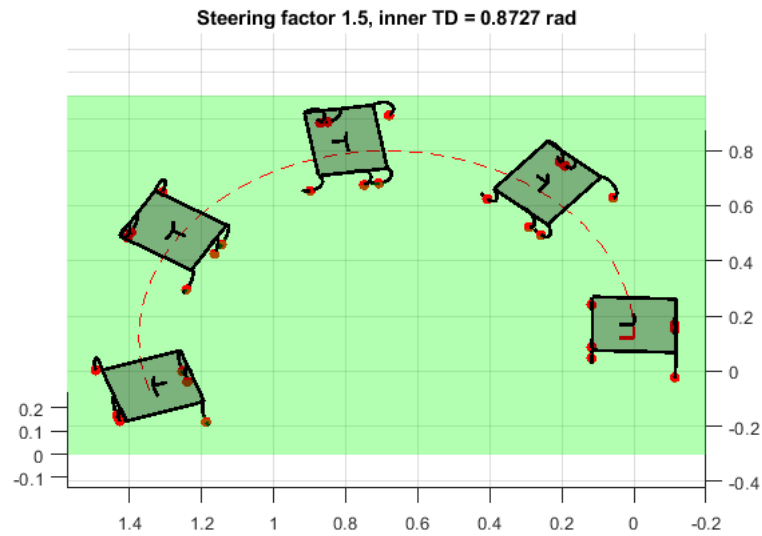


Figure 5-15: Simulation of steering factor 1.5 and $\theta_{TD}^i = 50^\circ$. Snapshots were taken at $t = 0s$, 6s, 18s and 24s.

5-4-4 Discussion of the results

The fact that the actual turning circle of the Zebro is so much larger than that in the simulation was an expected result. There are two reasons for this.

1. The actual steering factor of the Zebro became less than the simulated steering factor due to the shape of the legs not being half circular. Looking at Table 5-3, it can be seen that the simulated steering factor is $(120.7 - 46)/(118 - 50) = 1.1$ as expected, whereas the actual steering factor is $(132 - 46)/(130 - 50) = 1.08$, resulting in a slightly less sharp turn.
2. It is very likely that there is a problem with the slip model. For starters, the model was identified using a different material to the ground used in the experiments. Furthermore, plotting the slip ratio of each leg against time reveals that κ_i is unrealistically close to 0 during each step, implying that the leg always grips the ground perfectly which is clearly not the case.

Slip ratios The plotted slip ratios are shown in Figure 5-16. When the leg is in contact with the ground, the slip ratio appears to oscillate around 0, which is not the expected behaviour and indicates something wrong with the slip model.

This effect can be interpreted as the Zebro accelerating in one time step due to the traction force, causing the legs to be slipping in the next time step. In this next time step the slip ratio is therefore negative, causing the Zebro to decelerate enough for the slip ratio in the next time step to be positive again. This repeats throughout the entire step.

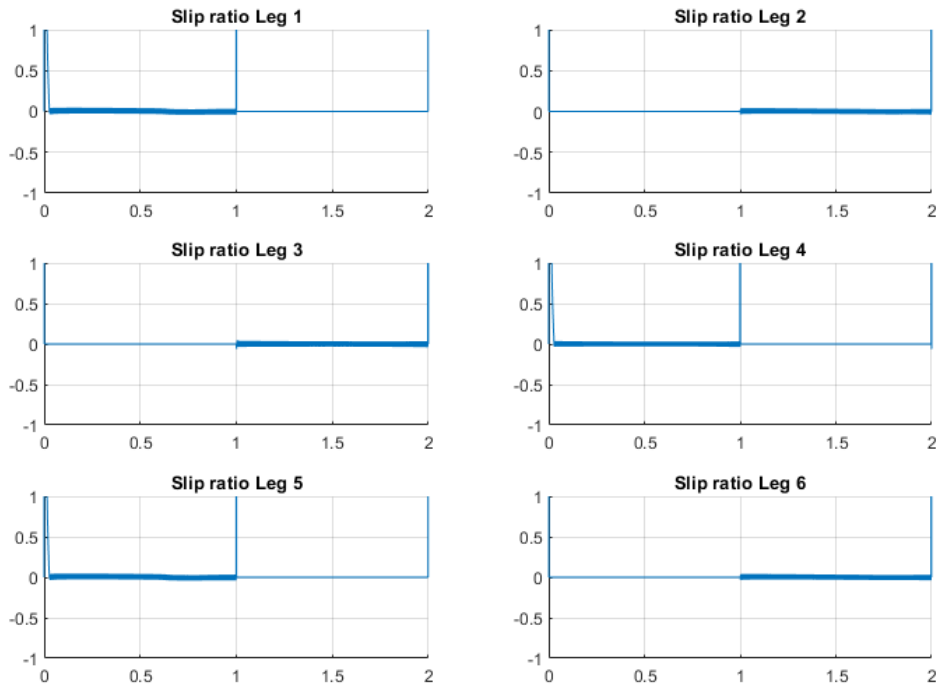


Figure 5-16: Slip ratios of legs 1 and 2 during one full step. For this simulation a time step of 0.001s was used.

Normal forces The normal forces plotted against time show a realistic result. It can be seen how the weight carried by each leg increases as its contact point nears the body's centre, which makes sense. The transition from rolling to toe contact can also be recognised clearly in legs 1,2,5 and 6, since the speed at which the leg's contact point is moving to or from the centre is greater in toe contact, corresponding to a steeper gradient in the normal force.

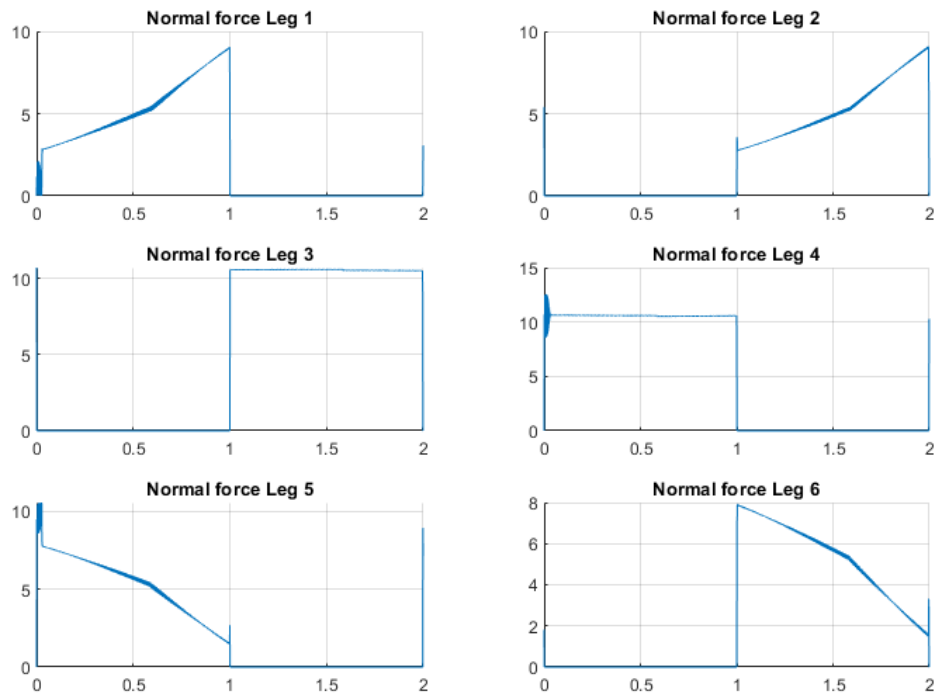


Figure 5-17: The normal force on each of the six legs during one full step.

Traction forces The results for the traction forces on each leg are more difficult to interpret. The reason for this is because the traction force is a function of the slip ratio, which has been shown to oscillate around 0, therefore causing the traction force to do the same.

To make sense of this data, a “net” force was found by taking the sum of each consecutive pair of positive and negative forces. This result is shown in Figure 5-19. Again, it becomes obvious that something is not right with the slip and traction model since the traction force on legs 1 and 2 should not be negative. Furthermore, the shapes of the curves are very unexpected and cannot be explained.

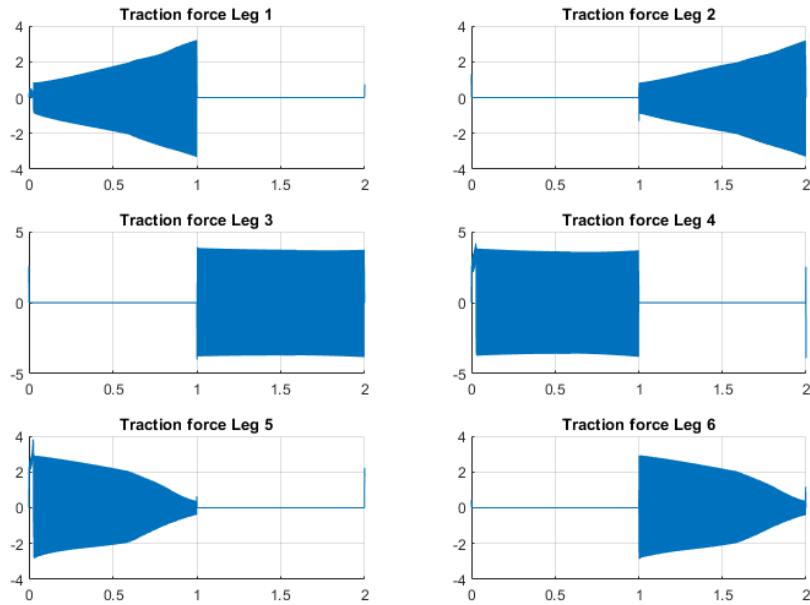


Figure 5-18: The traction force on each of the six legs during one full step.

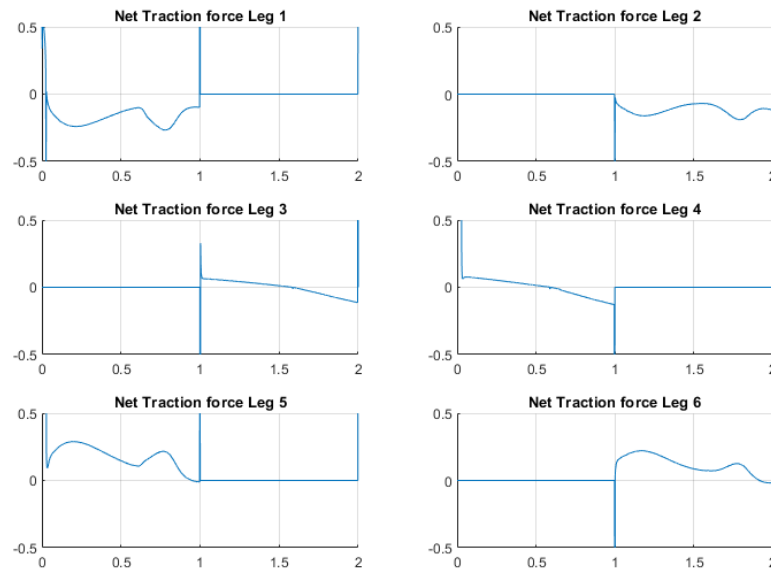


Figure 5-19: The traction force on each of the six legs during one full step.

5-5 Simulating tetrapod kinematics

In the dynamic model of the tripod the simple contact transition model in Table 3-1 was used. For a tetrapod this is significantly more complicated and the contact detection algorithm needs to be tested.

5-5-1 Initial state

For this model, both the touchdown and lift-off angles are assigned manually. This is because, unlike the tripod, there is less need for a perfect transition since the Zebro will appear to wobble no matter what. This means that a leg does not necessarily have to be at the touchdown angle when it touches down.

The initial leg angles are chosen so that each leg group reaches the touchdown angle at the same time as the subsequent leg group reaches the lift-off angle.

$$\theta_1(0) = \theta_4(0) = \theta_{LO} \quad (5-5)$$

$$\theta_2(0) = \theta_5(0) = \pi/2 \quad (5-6)$$

$$\theta_3(0) = \theta_6(0) = \theta_{TD} \quad (5-7)$$

Since two leg groups are on the ground whilst one group is in flight, the time spent on the ground should be twice the time spent in flight. Therefore, let $\tau_f = \tau_c/3$ and $\tau_g = 2\tau_c/3$, where τ_c is the cycle time again. The angular velocities of a leg on the ground is

$$\dot{\theta}_g = \frac{\theta_{LO} - \theta_{TD}}{\tau_g} \quad (5-8)$$

and the angular velocity of a leg in flight is

$$\dot{\theta}_f = \frac{2\pi - (\theta_{LO} - \theta_{TD})}{\tau_f} \quad (5-9)$$

so the initial state for the leg velocities is the following.

$$\dot{\theta}_1(0) = \dot{\theta}_4(0) = \dot{\theta}_f \quad (5-10)$$

$$\dot{\theta}_2(0) = \dot{\theta}_5(0) = \dot{\theta}_g \quad (5-11)$$

$$\dot{\theta}_3(0) = \dot{\theta}_6(0) = \dot{\theta}_g \quad (5-12)$$

Using these initial states and the contact detection algorithm outlined in Section 3-5-2 the kinematically determined motion of the tetrapod's "wobbling" effect can be modelled.

5-5-2 Simulation of the tetrapod's orientation during walking

The results shown are using $\theta_{TD} = \pi/4$, $\theta_{LO} = 3\pi/4$, $\tau_f = 1$ and $\tau_g = 2$. It is shown in Figures 5-20 to 5-22 how the contact state transitions in the following order:

1. $\mathcal{C} = \{2, 5, 6\} \rightarrow \mathcal{C} = \{2, 3, 6\}$, (see Figure 5-20)
2. $\mathcal{C} = \{2, 3, 6\} \rightarrow \mathcal{C} = \{1, 3, 4, 6\}$, (see Figure 5-21)
3. $\mathcal{C} = \{1, 3, 4, 6\} \rightarrow \mathcal{C} = \{1, 4, 5\}$, (see Figure 5-22a)
4. $\mathcal{C} = \{1, 4, 5\} \rightarrow \mathcal{C} = \{2, 4, 5\}$, (see Figure 5-22b)
5. Finally, from $\mathcal{C} = \{2, 4, 5\}$ back to the initial contact state $\mathcal{C} = \{1, 4, 5\}$.

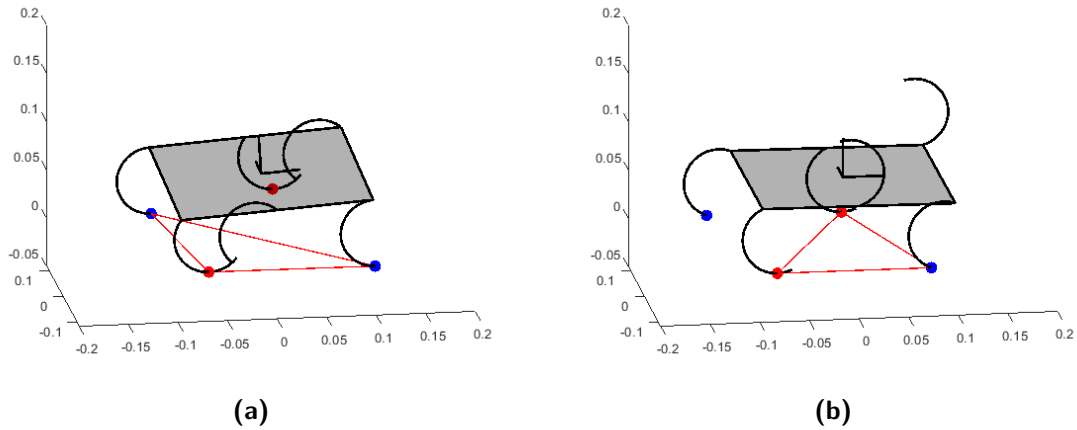


Figure 5-20: Starting on its hind legs, with $\mathcal{C} = \{2, 5, 6\}$, the Zebro transitions to $\mathcal{C} = \{2, 3, 6\}$ when leg 3 touches down.

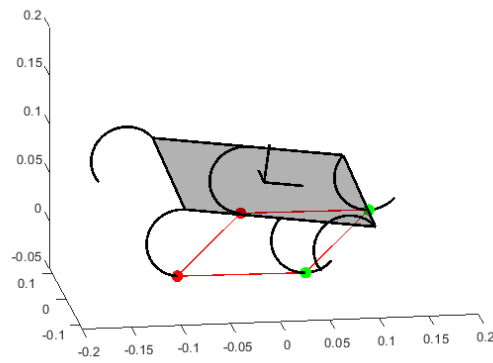


Figure 5-21: Legs 1 and 4 touchdown at the same time, making the contact state $\mathcal{C} = \{1, 3, 4, 6\}$

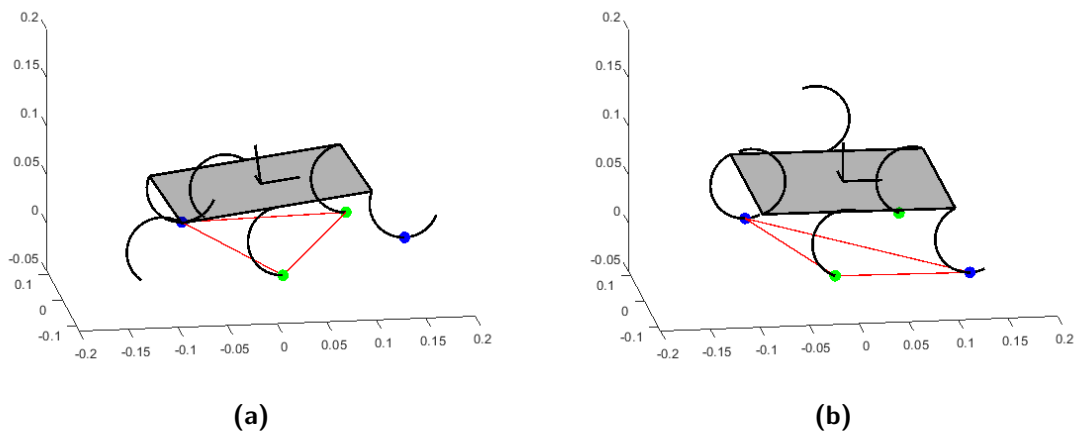


Figure 5-22: From $\mathcal{C} = \{1, 3, 4, 6\}$, both legs 3 and 6 lift up at the same time when leg 5 touches down making $\mathcal{C} = \{1, 4, 5\}$ (left). Next, leg 2 touches down making $\mathcal{C} = \{2, 4, 5\}$

In Figure 5-23 the body’s roll and pitch angles throughout one cycle of the tetrapod gait are plotted. Note that there are some discontinuous points in the plot, which can be attributed to contact being detected slightly too late, forcing the angle to “jump” to the correct value in the next time step. Decreasing this effect could be done by making the time step smaller, causing the error in transition time to be less.

These results could be validated by attaching two IMUs to the Zebro and tracking its orientation during the gait.

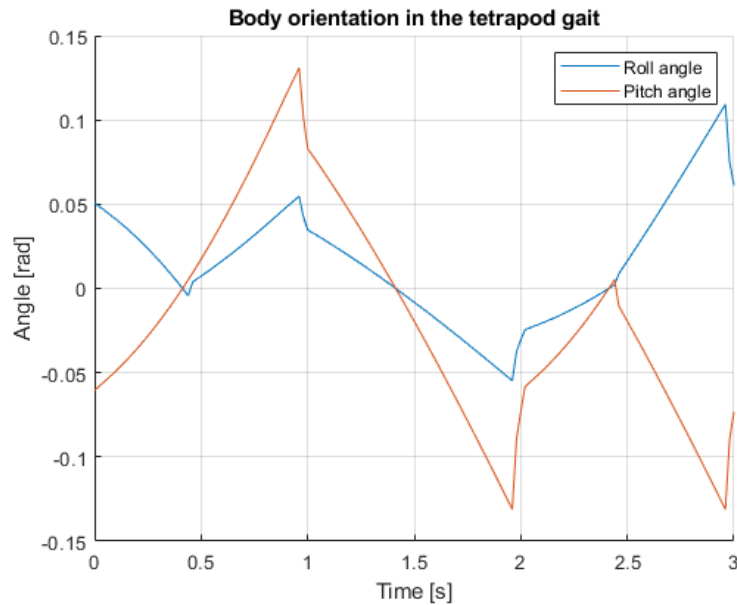


Figure 5-23: The roll and pitch angles of the body throughout one cycle of the tetrapod gait.

5-5-3 Comparison with a real Zebro

For this experiment, The Zebro’s touchdown and lift-off angles were chosen to be 38° and 117° respectively. This was an arbitrary choice since the goal of this experiment is simply to confirm whether the contact state transitions which were simulated correspond with real walking data. To achieve this, any combination of feasible touchdown and lift-off angles should be sufficient, but this choice resulted in a clear visual effect.

The following Figures 5-24 and 5-25 show a series of eight images of the walking robot. Each image is paired with a snapshot of the simulation at roughly the same time, to give an indication of the expected contact state.

The photographs match very well with the simulation, confirming that this kinematic method of finding the body’s orientation works as expected. The only differences between the photos and the simulation are occasional discrepancies between the position of the legs in flight. For example, in Figure 5-24, the first and fourth photo show the Zebro’s flight legs lag slightly behind the flight legs in the simulation. This is attributed to the fact that the Zebro sometimes pauses shortly before switching a leg to flight speed, whereas the simulation uses an instantaneous switch.

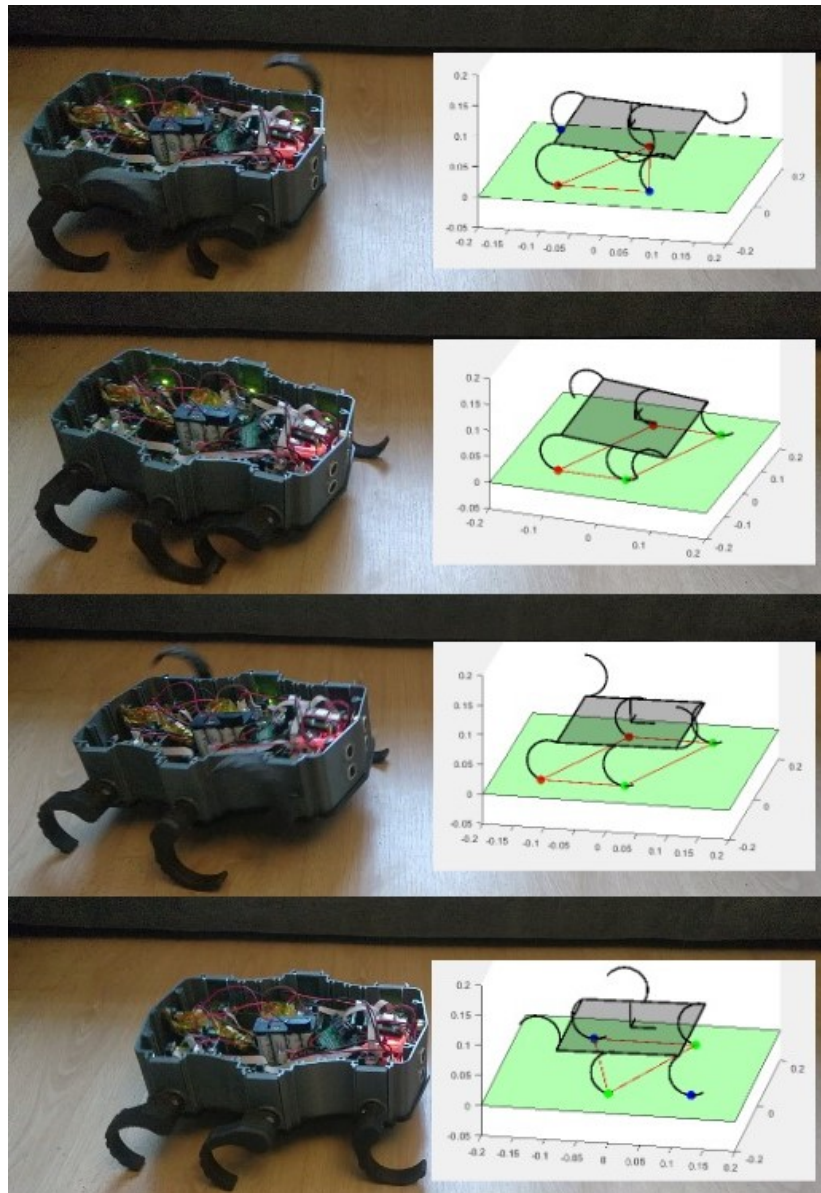


Figure 5-24: Snapshots of the simulated tetrapod gait beside photographs of an actual Zebro at the corresponding point in each step.

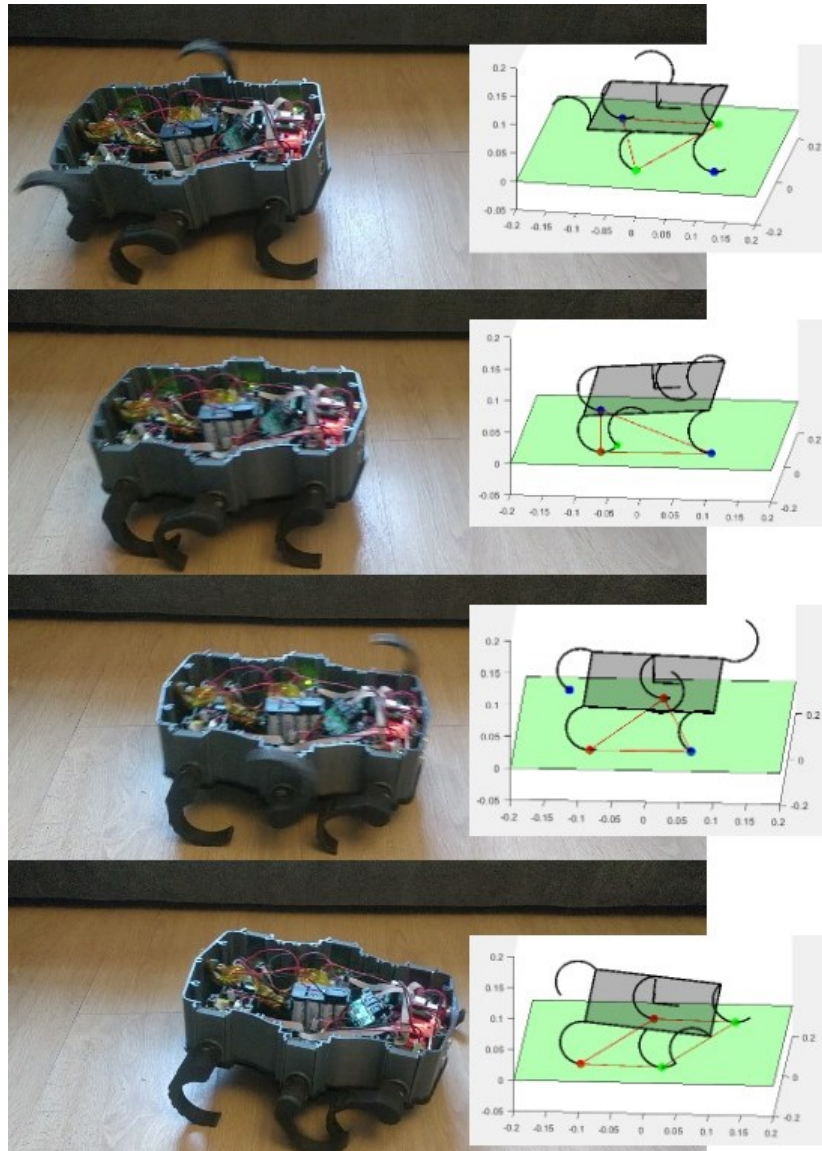


Figure 5-25: Snapshots of the simulated tetrapod gait beside photographs of an actual Zebro at the corresponding point in each step.

Conclusion and Future Work

In this research, the goal of modelling the dynamics of the Zebro has been achieved. Previous kinematic models have proven to be inaccurate, and it was expected that improvements could be made by incorporating traction forces into the model. To accomplish this, a slip model was identified experimentally, resulting in Pacejka's magic formula with the following constant characteristic parameters.

$$B = 34.34, \quad C = 1.564$$

However, the accuracy of this model is doubtful, since the curve produced does not appear to overlap with other data collected by myself. The data which was collected opens up the possibility for using a different curve fitting technique, which might yield a better model, but for the time being, the previously identified magic formula model is used.

With this model, traction forces between the leg and the ground can be estimated and used in the dynamic model. This model is built using the Newton-Euler framework for equations of motion. Newton-Euler was chosen over the Lagrange-Euler equations because of the heavy focus on traction forces, making the Lagrange-Euler equations less intuitive to use since these are found using the energy in a system, rather than forces.

The equations of motion are shown again in Eq. (6-1). By rearranging this equation the accelerations of the generalised coordinates of the body can be found.

$$\begin{bmatrix} m_b I_3 & \mathbf{0} \\ \mathbf{0} & I_b \end{bmatrix} \begin{bmatrix} \ddot{x}_b \\ \ddot{y}_b \\ \ddot{z}_b \\ \ddot{\theta}_r \\ \ddot{\theta}_p \\ \ddot{\theta}_y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dot{\theta}_p \dot{\theta}_y (I_{zz} - I_{yy}) \\ \dot{\theta}_r \dot{\theta}_y (I_{xx} - I_{zz}) \\ \dot{\theta}_p \dot{\theta}_r (I_{yy} - I_{xx}) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \sum_{i \in \mathcal{C}} ({}^I \mathbf{F}_{c,i}) \\ \sum_{i \in \mathcal{C}} (({}^I \mathbf{p}_{c,i} - {}^I \mathbf{p}_b) \times {}^I \mathbf{F}_{c,i}) \end{bmatrix} \quad (6-1)$$

The contact state \mathcal{C} which appears in Eq. (6-1) is the set of leg indices which are in contact with the ground. This state can change discretely when contact is detected using the contact detection algorithm in Chapter 3. The efficacy of this algorithm is shown in Chapter 5 where the previously unmodelled contact transitions in the tetrapod gait are clearly shown.

To display the results of each simulation a GUI was written in Matlab showing a heavily simplified Zebro walking on the ground. In these results we see the Zebro walk in both a straight line and a curve using Suriana's max-plus algorithm.

We then go on to significantly improve and simplify the Zebro's turning algorithm for a tripod gait. By differing the touchdown and lift-off angles of each side of the Zebro, rather than the touchdown and lift-off *times*, we can preserve a perfect tripod rhythm in which a smooth transition from one contact state to the next is experienced. The advantages of this are twofold:

1. Modelling the system becomes much simpler, because contact transitions are guaranteed to be $\mathcal{C} = \{1, 4, 5\} \leftrightarrow \mathcal{C} = \{2, 3, 6\}$, so contact detection can be simplified.
2. The real-life Zebro experiences much less wobbling while turning because of these smoother transitions.

This algorithm has been shown to produce a perfectly circular path in simulation as well as in reality. However, the actual Zebro's turning circle is significantly larger than predicted. This has been attributed to the fact that the Zebro slips a lot more than is simulated, therefore the traction required to make a sharp turn is not present.

However, despite simulations which appear to show realistic behaviour, the strange result for the slip ratio and the traction forces remains an unexplained problem. It is possible that the method of applying Pacejka's slip formula to a half-circular shape is incorrect, in which case further research is required.

6-1 Future Work

As was previously mentioned, further research on how to apply a slip model to a half-circular leg could be useful, since the approach used in this report did not appear to yield correct results.

Furthermore, three types of contact have been mentioned in this report: rolling, toe and hip contact, where hip contact could be applied to the heavily simplified model of the Zebro, but in actuality is almost impossible to occur due to the size of the Zebro's body casing. Its body will almost always touch the ground before the hip. Adding a state for body contact should be implemented for a proper model of the system. Expanding on this, a fifth contact state should also be included, namely the *aerial* state in which all the legs have left the ground. The reason this cannot be implemented in this model is because the contact state is currently calculated under the assumption that the lowest leg is in contact with the ground and therefore always has a contact force acting on it. In this model the contact forces are calculated from the contact state and the body orientation, meaning the contact state needs to be estimated *before* finding the forces. An aerial phase is the result of large contact forces pushing the Zebro up from the ground, so to implement this possibility in the model, the forces will need to be calculated before the contact state - when the force on a leg reaches 0, it is considered in the air.

A vision for the future model consists of three parts: one model for the motion whilst in the air; one dynamic model for when the Zebro is moving fast; and one kinematic model for if the Zebro moves slowly, such that an aerial phase and slippage is impossible.

Another future improvement to be made on this algorithm is to allow for non-flat surfaces. If the Zebro is ever to walk on the moon, then rocky and uneven ground must be accounted for. With full knowledge of the Zebro's surroundings, it is certainly possible to find the leg angle which will result in touchdown on an uneven surface by using the potential contact points calculated in Chapter 3. Without full knowledge of the surroundings, however, a closed-loop system would be required in which motor torque is measured and contact with the ground can be detected by a spike in the torque required for rotation.

Finally, the completion of the tetrapod model could be achieved in future work. This is certainly possible using the kinematically modelled orientation of the Zebro during a step in the tetrapod gait in Section 5-5-2. This can be used to estimate $\dot{\theta}_p$ and $\dot{\theta}_r$ which are subsequently used to find the contact forces responsible for these rotational velocities. With these contact forces the dynamic model will be complete.

Appendix A

MATLAB Code

A-1 Parameters

Listing A.1: Important parameters of the Zebro

```
1  % BODY PARAMETERS %
2  b1 = 0.1156;
3  l = 0.097;
4  L = 2*b1;
5  r = 0.035; % radius of leg
6
7  Bp_H = [1, 1, 0, 0, -1, -1; % positions of legs on body
8          b1, -b1, b1, -b1, b1, -b1;
9          0, 0, 0, 0, 0, 0];
10 m = 2.17;
11 g = 9.81;
12
13 h = 0.069; % height
14 w = 0.2512; % width
15 d = 0.194; % depth
16
17 J_x = m*(w^2 + h^2)/12;
18 J_y = m*(d^2 + h^2)/12;
19 J_z = m*(w^2 + d^2)/12;
20 Jb = diag([J_x, J_y, J_z]);
```

A-2 Plot zebro

Listing A.2: Code to plot the Zebro model

```
1 function plot_zebro(q, p_hips, r)
```

```

2     p_body = q(1:3);
3     theta_r = q(4);
4     theta_p = q(5);
5     theta_y = q(6);
6     theta = q(7:12);
7
8     H_IB = H_Inertia_to_Body(theta_r, theta_p, theta_y, p_body);
9     body = H_IB*[p_hips; 1,1,1,1,1,1];
10    plot3(body(1,[1,3,5,6,4,2,1]), body(2,[1,3,5,6,4,2,1]), body
        (3,[1,3,5,6,4,2,1]), 'k', 'linewidth', 2)
11
12    phi = linspace(0, pi, 20);
13    leg = [r*cos(phi)+r; zeros(1,20); -r*sin(phi); ones(1,20)];
14
15    for i = 1:6
16        % Plot the legs %
17        H_BH = H_Body_to_Hip(theta(i), p_hips(:,i));
18        leg_i = H_IB*H_BH*leg;
19        plot3(leg_i(1,:), leg_i(2,:), leg_i(3,:), 'k', 'linewidth', 2)
20
21        % Plot the minima %
22        alph = find_alpha(theta_r, theta_p, theta_y, theta(i));
23        H_HL = H_Hip_to_Leg(alph(1), r);
24        if alph(2) == 0 % Toe contact
25            H_LC = H_Leg_to_Toe(alph(1), r, q);
26        elseif alph(2) == pi % Hip contact
27            H_LC = H_Leg_to_Hip(alph(1), r);
28        else % Minimum contact
29            H_LC = H_Leg_to_Contact(r,q);
30        end
31
32        coordinate_frame = 0.04*[[1;0;0],[0;0;0],[0;1;0],[0;0;0],[0;0;1]];
33
34        contact_i = H_IB*H_BH*H_HL*H_LC*[0;0;0;1];
35        inertial_coords = [coordinate_frame; [1,1,1,1,1]];
36        body_coords = H_IB*[coordinate_frame; [1,1,1,1,1]];
37        hip_coords = H_IB*H_BH*[coordinate_frame; [1,1,1,1,1]];
38        leg_coords = H_IB*H_BH*H_HL*[coordinate_frame; [1,1,1,1,1]];
39        min_coords = H_IB*H_BH*H_HL*H_LC*[coordinate_frame; [1,1,1,1,1]];
40
41        plot3(contact_i(1,:), contact_i(2,:), contact_i(3,:), 'r.', 'markersize
            ', 20)
42        plot3(inertial_coords(1,:), inertial_coords(2,:), inertial_coords(3,:),
            'r-', 'linewidth', 2)
43        plot3(body_coords(1,:), body_coords(2,:), body_coords(3,:), 'k-', '
            linewidth', 2)
44

```

```

45     ground = fill3(10*[-1, -1, 1, 1, -1], 10*[-1, 1, 1, -1, -1],
        [0,0,0,0,0], 'g');
46     body_fill = fill3(body(1,[1,3,5,6,4,2,1]), body(2,[1,3,5,6,4,2,1]),
        body(3,[1,3,5,6,4,2,1]), 'k');
47
48     alpha(ground, 0.3)
49     alpha(body_fill, 0.3)
50 %     plot3(hip_coords(1,:), hip_coords(2,:), hip_coords(3,:), 'r-', '
        linewidth', 1)
51 %     plot3(leg_coords(1,:), leg_coords(2,:), leg_coords(3,:), 'g-', '
        linewidth', 1)
52 %     plot3(min_coords(1,:), min_coords(2,:), min_coords(3,:), 'b-', '
        linewidth', 1)
53     end
54 end

```

A-3 Animate zebro

Listing A.3: Code to animate the Zebro model

```

1  function animate_zebro(Q, dt, speed, p_hips, r, view_angle)
2      for i = 1:length(Q)/speed
3          clf; hold on; grid on; axis equal;
4          view(view_angle(1), view_angle(2));
5
6          xlim([min(Q(1,:))-0.2, max(Q(1,:))+0.2]);
7          ylim([min(Q(2,:))-0.2, max(Q(2,:))+0.2]);
8          zlim([min(Q(3,:))-0.2, max(Q(3,:))+0.2]);
9
10         plot_zebro(Q(:,i*(speed)-2), p_hips, r);
11         plot3(Q(1,1:i*(speed)-2), Q(2,1:i*(speed)-2), zeros(1,i*(speed)-2), 'r
            —')
12         pause(dt);
13     end
14 end

```

A-4 Main

Listing A.4: Main code to run

```

1  clear; %close all;
2  clc;
3
4  parameters
5
6  steering_factor = 1.1;
7  direction = left;

```

```

8 tau_f = 0.8;
9 tau_p = 0.1;
10 tau_g = 0.8;
11
12 theta_LO_suri = (90 + 40) * (pi/180);
13 theta_TD_suri = 35 * (pi/180);
14
15 % UNCOMMENT FOR SURIANA TURNING INSTEAD OF TRIPOD TURNING
16 cycle_time = tau_f+tau_g;
17 [theta_i, theta_o, thetad_i, thetad_o] = init_suriana_tripod(tau_f, tau_p,
    tau_g, theta_LO_suri, theta_TD_suri);
18 Ip_B = [0;0;r*(sin(theta_TD_suri)+1)];
19
20 % UNCOMMENT FOR TRIPOD TURNING INSTEAD OF SURIANA TURNING
21 % theta_i.TD = deg2rad(125-75);
22 % cycle_time = 2;
23 % [theta_i, theta_o, thetad_i, thetad_o] = init_tripod(theta_i.TD,
    steering_factor, cycle_time);
24
25 inner_leg_height = r*(sin(theta_i.TD)+1);
26 outer_leg_height = r*(sin(theta_o.TD)+1);
27 avg_body_height = 0.5*(inner_leg_height + outer_leg_height);
28 Ip_B = [0;0;avg_body_height];
29 roll0 = atan(r*(sin(theta_i.TD) - sin(theta_o.TD))/L);
30
31 if direction == left
32     thetad = [thetad_i.g;
33             thetad_o.f;
34             thetad_i.f;
35             thetad_o.g;
36             thetad_i.g;
37             thetad_o.f];
38     theta = [theta_i.TD;
39            theta_o.LO;
40            theta_i.LO;
41            theta_o.TD;
42            theta_i.TD;
43            theta_o.LO];
44     C = [1,4,5]; C_prev = [2,3,6];
45     C_i = [1,5];
46     C_o = [4];
47 else
48     thetad = [thetad_o.g;
49             thetad_i.f;
50             thetad_o.f;
51             thetad_i.g;
52             thetad_o.g];

```

```
53         thetad_i.f];
54     theta = [theta_o.TD;
55             theta_i.L0;
56             theta_o.L0;
57             theta_i.TD;
58             theta_o.TD;
59             theta_i.L0];
60     C = [1,4,5]; C_prev = [2,3,6];
61     C_i = [2,6];
62     C_o = [3];
63 end
64
65 q = [Ip_B; roll0;0;0; theta];
66 qd = [0;0;0;0;0;0; thetad];
67 qdd = [0;0;0;0;0;0;0;0;0;0;0];
68
69 eps = 0.000001;
70 results.q = [];
71 results.qd = [];
72 results.qdd = [];
73 results.f_n = [];
74 results.f_t = [];
75 results.f_g = [];
76 results.IF.x = [];
77 results.IF.y = [];
78 results.IF.z = [];
79 results.phi = [];
80 results.phi_ = [];
81 results.phid = [];
82 results.k = [];
83 results.C = [];
84 results.v_C = [];
85 results.v_R = [];
86 results.v_rot.x = [];
87 results.v_rot.y = [];
88 results.v_rot.z = [];
89 results.v_B = [];
90
91 % SIMULATION PARAMETERS
92 dt = 0.001;
93 tstop = 3*cycle_time;
94 T = 0:dt:tstop;
95 results.T = T;
96 fprintf('Simulating %0.1f seconds \n\n', tstop);
97 tic;
98
99 for t = T
```

```

100 % BODY ROTATION
101 R_IB = R_r(q(4)) * R_p(q(5)) * R_y(q(6));
102
103 % Update leg speeds
104 qd(7:12) = update_leg_speeds(q, theta_i, theta_o, thetad_i, thetad_o,
    direction);
105
106 phi = phi(q); % phi(1,:) is phi, phi(2,:) is phi*
107 phid = phi_d(q, qd);
108 Ip_C = contact_pos(q, phi, R_IB, Bp_H, r);
109 Ip_H = [Ip_B, Ip_B, Ip_B, Ip_B, Ip_B, Ip_B] + R_IB*Bp_H;
110 Ip_B = q(1:3);
111 H_IB = H_Inertia_to_Body(q(4), q(5), q(6), q(1:3));
112 H_BI = inv(H_IB);
113 BP_C = H_BI*[Ip_C; 1,1,1,1,1,1];
114 Bp_C = BP_C(1:3,:);
115
116 [C, C_i, C_o] = update_contact_state2(phi, C_i, direction, r);
117 % IF CONTACT STATE CHANGE
118 if ~isequal(C_prev, C)
119     if length(C) == 2
120         C = C_prev;
121     end
122     fprintf('Contact state change at: t = %0.3f. (Elapsed time is %0.3f
        seconds)\n', t, toc);
123     [z, z_i, z_o] = z_estimate(q, C_i, C_o, phi, r);
124     [zd, zd_i, zd_o] = zd_estimate(q, C_i, C_o, phi, phid, thetad_i,
        thetad_o, r);
125     theta_rd = theta_rd_estimate(q, zd_i, zd_o, L);
126
127 % Reset kinematically determined parameters
128 qd(3) = zd;
129 qd(4) = theta_rd;
130 q(4) = atan(r*(sin(theta_i.TD) - sin(theta_o.TD))/L);
131 q(4) = theta_r_estimate(q, z_i, z_o, L);
132 q(5) = 0;
133 q(3) = z;
134 end
135 C_prev = C;
136
137 % Weight distribution estimation
138 [f_g, C] = weight_distribution(q, Ip_C, Ip_H, C, m);
139 R_IC = cell(1,6);
140 for i = C
141     if phi(1,i) == 0 % Toe contact
142         R_IC{i} = R_y(q(6)) * R(q(6+i)) * R(phi(2,i) - pi/2);
143     else % Rolling contact

```

```

144         R_IC{i} = R_y(q(6)) * R(q(6+i)) * R(phi(1,i) - pi/2);
145     end
146 end
147
148 % Velocities of the contact points
149 Cv_C = contact_velocities(q,qd,R_IC,Ip_C,C);
150
151 % Slip ratio of the contact legs
152 [k, v_C, v_R, v_rot, v_B] = slip_ratio(q, qd, Ip_C, Ip_H, Bp_C, Bp_H, R_IC,
    phi, phid, C, r);
153
154 x0= f_g(C)';
155 opts = optimoptions(@fmincon,'Algorithm','interior-point','Display','off');
156
157 [z, z_i, z_o] = z_estimate(q, C_i, C_o, phi, r);
158 [zd, zd_i, zd_o] = zd_estimate(q, C_i, C_o, phi, phid, thetad_i, thetad_o,
    r);
159 theta_r = theta_r_estimate(q, z_i, z_o, L);
160 theta_rd = theta_rd_estimate(q, zd_i, zd_o, L);
161 zdd = zdd_estimate(C, phi, phid, r);
162 zdd = mean([mean(zdd(C_i)), mean(zdd(C_o))]);
163
164 Aeq = [1, 1, 1];
165 Aeq = ones(1,length(C));
166 Beq = m*g+m*zdd;
167 lb = zeros(1, length(C));
168 ub = m*g*ones(1,length(C));
169
170 [x, val] = fmincon(@(x) NE_opt(q, qd, C, R_IC, phi, k, x, Ip_C, Jb, m, dt,
    theta_r, 0, theta_rd, 0), x0, [], [], Aeq, Beq, lb,ub,[],opts);
171
172 f_n = [0,0,0,0,0,0];
173 f_n(C) = x;
174 f_t = pacejka(f_n, k);
175 f_t_lat = pacejka_lateral(f_n, qd, R_IC, C);
176
177 IF = zeros(3, 6);
178 for i = C
179     IF(:,i) = R_IC{i} * [f_t(i); f_t_lat(i); f_n(i)];
180 end
181
182
183 % NEWTON EULER EQUATIONS
184 [Ip_Bdd, Iw_Bd] = NE(q, qd, IF, Ip_C, Ip_H, Jb, m);
185
186 % ACCELERATION, VELOCITY and POSITION
187 qdd = [Ip_Bdd; Iw_Bd; 0;0;0;0;0;0];

```

```
188     qd = qd + qdd*dt;
189     q = q + qd*dt;
190
191     % Save the results
192     results = save_results(results, q, qd, qdd, f_n, f_t, f_g, IF, phi, phid, k
        , C, v_C, v_R, v_rot, v_B);
193 end
194
195 fprintf('\nFinished simulation\n')
196 toc
197
198 %%
199 figure(1); clf; hold on;
200 view_angle = [60,20];
201 % view_angle = [0,90];
202 % view_angle = [20,50];
203 % view_angle = [90,0];
204 view_angle = [40,10];
205 % speed = 1/dt;
206 speed = 0.1/dt;
207 tic;
208 fprintf('\nAnimating\n')
209 animate_zebro(results.q, dt, speed, Bp_H, r, view_angle);
210
211 toc
212
213 %%
214 f = 2;
215
216 figure(f); clf;
217 subplot(1,3,1)
218 hold on; grid on;
219 plot(results.T, results.IF.x);
220 title('Inertial X forces on each leg');
221
222 subplot(1,3,2)
223 hold on; grid on;
224 plot(results.T, results.IF.y);
225 title('Inertial Y forces on each leg');
226
227 subplot(1,3,3)
228 hold on; grid on;
229 plot(results.T, results.IF.z);
230 title('Inertial Z forces on each leg');
231 legend({'Leg 1', 'Leg 2', 'Leg 3', 'Leg 4', 'Leg 5', 'Leg 6'})
232
233 f = f+1;
```



```
234 figure(f); clf;
235 subplot(2,3,1)
236 hold on; grid on;
237 plot(results.T, results.IF.x(:,[1,4,5]));
238 title('Inertial X forces on each leg');
239 subplot(2,3,4)
240 hold on; grid on;
241 plot(results.T, results.IF.x(:,[2,3,6]));
242 title('Inertial X forces on each leg');
243
244 subplot(2,3,2)
245 hold on; grid on;
246 plot(results.T, results.IF.y(:,[1,4,5]));
247 title('Inertial Y forces on each leg');
248 subplot(2,3,5)
249 hold on; grid on;
250 plot(results.T, results.IF.y(:,[2,3,6]));
251 title('Inertial Y forces on each leg');
252
253 subplot(2,3,3)
254 hold on; grid on;
255 plot(results.T, results.IF.z(:,[1,4,5]));
256 title('Inertial Z forces on each leg');
257 legend({'Leg 1', 'Leg 4', 'Leg 5'})
258 subplot(2,3,6)
259 hold on; grid on;
260 plot(results.T, results.IF.z(:,[2,3,6]));
261 title('Inertial Z forces on each leg');
262 legend({'Leg 2', 'Leg 3', 'Leg 6'})
263
264
265 f = f+1;
266 figure(f); clf;
267 hold on; grid on;
268 f_n_plot = plot(results.T, results.f_n);
269 f_g_plot = plot(results.T, results.f_g);
270
271 for i = 1:6
272     f_g_plot(i).Color = f_n_plot(i).Color;
273     f_g_plot(i).LineStyle = '—';
274 end
275
276 title('Normal force on the legs');
277 legend({'Leg 1', 'Leg 2', 'Leg 3', 'Leg 4', 'Leg 5', 'Leg 6'})
278
279
280 f=f+1;
```

```
281 figure(f); clf; hold on; grid on;
282 plot(results.T, results.k);
283 legend({'Leg 1', 'Leg 2', 'Leg 3', 'Leg 4', 'Leg 5', 'Leg 6'})
284 title('Slip ratios');
285
286 % figure(100); clf; hold on; grid on;
287 % plot(results.T, results.k(:,[2,3,6]));
288 % legend({'Leg 2', 'Leg 3', 'Leg 6'})
289 % title('Slip ratios');
290
291
292 f=f+1;
293 figure(f); clf;
294 subplot(3,1,1)
295 hold on; grid on;
296 plot(results.T, results.q(1:3, :));
297 legend({'x', 'y', 'z'})
298 title('Position of body');
299 subplot(3,1,2)
300 hold on; grid on;
301 plot(results.T, results.qd(1:3, :));
302 legend({'xd', 'yd', 'zd'})
303 title('Velocity of body');
304 subplot(3,1,3)
305 hold on; grid on;
306 plot(results.T, results.qdd(1:3, :));
307 legend({'xdd', 'ydd', 'zdd'})
308 title('Acceleration of body');
309
310 f=f+1;
311 figure(f); clf;
312 subplot(1,3,1)
313 hold on; grid on;
314 plot(results.T, results.q(4:6, :))
315 legend({'roll', 'pitch', 'yaw'})
316 title('Roll, pitch and yaw angles')
317 subplot(1,3,2)
318 hold on; grid on;
319 plot(results.T, results.qd(4:5, :))
320 legend({'rolld', 'pitchd'})
321 title('Roll, pitch and yaw speeds')
322 subplot(1,3,3)
323 hold on; grid on;
324 plot(results.T, results.qdd(4:5, :))
325 legend({'rolldd', 'pitchdd'})
326 title('Roll, pitch and yaw accelerations')
327
```

```
328 f = f+1;
329 figure(f); clf; hold on; grid on; axis equal;
330 plot(results.q(1,:), results.q(2,:))
331 title('Path');
332
333
334 f = f+1;
335 figure(f); clf;
336
337 for i = 1:6
338     subplot(3,2,i)
339     hold on; grid on;
340     plot(results.T, results.k(:,i));
341     ylim([-1,1])
342     title(sprintf('Slip ratio Leg %i', i))
343 end
344
345 f = f+1;
346 figure(f); clf;
347
348 odds = 3:2:length(results.T);
349 evens = 2:2:length(results.T)-1;
350 f_t_filter = results.f_t(evens,:) + results.f_t(odds,:);
351
352 for i = 1:6
353     subplot(3,2,i)
354     hold on; grid on;
355     plot(linspace(0,results.T(end), size(f_t_filter,1)), f_t_filter(:,i));
356     title(sprintf('Net Traction force Leg %i', i))
357 end
358
359 f = f+1;
360 figure(f); clf;
361
362 for i = 1:6
363     subplot(3,2,i)
364     hold on; grid on;
365     plot(results.T, results.f_n(:,i));
366     title(sprintf('Normal force Leg %i', i))
367 end
368
369 f = f+1;
370 figure(f); clf; grid on; hold on;
371 for i = 1:6
372     subplot(3,2,i)
373     hold on; grid on;
374     plot(results.T, results.v_C(:,i));
```

```

375     plot(results.T, results.v_R(:,i));
376     plot(results.T, results.v_rot.x(i,:));
377     title(sprintf('Contact velocity Leg %i', i))
378 end
379
380 f = f+1;
381 figure(f); clf; grid on; hold on;
382
383 subplot(2,2,1)
384 hold on; grid on;
385 plot(results.T, results.q(1:3, :));
386 legend({'x', 'y', 'z'})
387 title('Position of body');
388 subplot(2,2,3);
389 hold on; grid on;
390 plot(results.T, results.qd(1:3, :));
391 legend({'xd', 'yd', 'zd'})
392 title('Velocity of body');
393
394 subplot(2,2,2)
395 hold on; grid on;
396 plot(results.T, results.q(4:6, :))
397 legend({'roll', 'pitch', 'yaw'})
398 title('Roll, pitch and yaw angles')
399 subplot(2,2,4)
400 hold on; grid on;
401 plot(results.T, results.qd(4:5, :))
402 legend({'rolld', 'pitchd'})
403 title('Roll, pitch and yaw speeds')
404
405 %% Save results to ..\Results\WS_14\
406 % note = 'lateral_force_pacejka';
407 filename = sprintf('time_%0.2f_steer_%0.2f_iTD_%0.4f_tauc_%0.1f_v1.mat', tstop,
    steering_factor, theta_i.TD, cycle_time);
408 if exist('note', 'var')
409     filename = strcat(note, '___', filename);
410     clear note
411 end
412 filepath = strcat('C:\Users\Folkert\MATLAB Drive\Thesis\Results\Experiments\',
    filename);
413 version = 1;
414 while exist(filepath, 'file')
415     version = version+1;
416     filepath(end-4:end) = [];
417     filepath = strcat(filepath, num2str(version), '.mat');
418 end
419 save(filepath, 'results', 'Bp_H', 'r', 'dt')

```

A-5 Gait initiation

A-5-1 Init tripod

Listing A.5: Tripod initiation

```

1 function [theta_i, theta_o, thetad_i, thetad_o] = init_tripod(theta_i_TD, K, tc
   )
2
3
4 tg = tc/2; % tc is cycle time. Ground time and flight time must be equal, so
   tg = tf = tc/2
5 tf = tc/2;
6 theta_i.TD = theta_i_TD;
7 theta_i.L0 = pi - asin(0.5*sin(theta_i.TD)+0.5); % L0 angle where height is
   equal to height at TD angle.
8
9 syms x
10 theta_o.TD = vpasolve(pi - asin(0.5*sin(x)+0.5) - x == K*(theta_i.L0-theta_i.TD
   )); % Set distance covered on outer side = K * distance on inner side
11 theta_o.TD = double(theta_o.TD); % sym to double
12 theta_o.L0 = pi - asin(0.5*sin(theta_o.TD)+0.5);
13
14 thetad_i.g = (theta_i.L0 - theta_i.TD) / tg; % rad/s required to get from TD
   to L0 in tg seconds
15 thetad_o.g = (theta_o.L0 - theta_o.TD) / tg;
16 thetad_i.f = (2*pi - (theta_i.L0 - theta_i.TD)) / tf; % rad/s required from
   L0 to TD in tf seconds.
17 thetad_o.f = (2*pi - (theta_o.L0 - theta_o.TD)) / tf;
18
19 fprintf('Tripod initialised with steering factor %0.1f. \n\tInner legs: TD =
   %0.4f, L0 = %0.4f. \n\tOuter legs: TD = %0.4f, L0 = %0.4f.\n',...
20 K, theta_i.TD, theta_i.L0, theta_o.TD, theta_o.L0);
21
22 fprintf('Tripod initialised with steering factor %0.1f. \n\tInner legs: TD =
   %0.4f, L0 = %0.4f. \n\tOuter legs: TD = %0.4f, L0 = %0.4f.\n',...
23 K, rad2deg(theta_i.TD), rad2deg(theta_i.L0), rad2deg(theta_o.TD), rad2deg(
   theta_o.L0));
24 end

```

A-5-2 Init Suriana tripod

Listing A.6: Suriana tripod initiation

```

1 function [theta_i, theta_o, thetad_i, thetad_o] = init_suriana_tripod(tau_f,
   tau_p, tau_g, theta_L0, theta_TD)
2
3 theta_d_i_g = (theta_L0 - theta_TD) / (tau_g + tau_p);

```

```

4  theta_d_o_g = (theta_LO - theta_TD) / tau_g;
5  theta_d_i_f = (2*pi - (theta_LO - theta_TD)) / tau_f;
6  theta_d_o_f = (2*pi - (theta_LO - theta_TD)) / (tau_f + tau_p);
7
8
9  theta_d = [theta_d_i_g;
10           theta_d_o_f;
11           theta_d_i_f;
12           theta_d_o_g;
13           theta_d_i_g;
14           theta_d_o_f];
15
16  theta = [theta_TD;
17          theta_LO;
18          theta_LO;
19          theta_TD;
20          theta_TD;
21          theta_LO];
22
23  theta_i.TD = theta_TD;
24  theta_i.LO = theta_LO;
25  theta_o.TD = theta_TD;
26  theta_o.LO = theta_LO;
27  thetad_i.f = theta_d_i_f;
28  thetad_i.g = theta_d_i_g;
29  thetad_o.f = theta_d_o_f;
30  thetad_o.g = theta_d_o_g;
31
32
33  fprintf('Suriana-Tripod initialised with tau_f = %0.1f, tau_g = %0.1f and tau_p
         = %0.1f. \n\t Touchdown angle = %0.4f, Liftoff angle = %0.4f.\n',...
34         tau_f, tau_g, tau_p, theta_TD, theta_LO);
35
36  end

```

A-6 Rotation Matrices

Listing A.7: Leg rotation

```

1  function ret = R(theta)
2      ret = [cos(theta),  0, sin(theta);
3            0,           1, 0;
4            -sin(theta), 0, cos(theta)];
5  end

```

Listing A.8: Roll rotation

```

1 function ret = R_r(theta)
2     ret = [1, 0, 0;
3           0, cos(theta), -sin(theta);
4           0, sin(theta), cos(theta)];
5 end

```

Listing A.9: Pitch rotation

```

1 function ret = R_p(theta)
2     ret = [cos(theta), 0, sin(theta);
3           0, 1, 0;
4           -sin(theta), 0, cos(theta)];
5 end

```

Listing A.10: Yaw rotation

```

1 function ret = R_y(theta)
2     ret = [cos(theta), -sin(theta), 0;
3           sin(theta), cos(theta), 0;
4           0, 0, 1];
5 end

```

A-7 Homogeneous Transformations

Listing A.11: Homogeneous transformation inertial to body

```

1 function ret = H_Inertia_to_Body(theta_r, theta_p, theta_y, p)
2     ret = [R_r(theta_r)*R_p(theta_p)*R_y(theta_y), p; 0,0,0,1];
3 end

```

Listing A.12: Homogeneous transformation body to hip

```

1 function ret = H_Body_to_Hip(theta, p)
2     ret = [R(theta), p; 0,0,0,1];
3 end

```

Listing A.13: Homogeneous transformation hip to leg

```

1 function ret = H_Hip_to_Leg(alpha, r)
2     ret = [R(alpha-pi/2), [r;0;0]; 0,0,0,1];
3 end

```

Listing A.14: Homogeneous transformation leg to hip

```

1 function ret = H_Leg_to_Hip(alpha, r)
2     ret = [eye(3), [r*sin(pi+alpha);0;-r*cos(pi+alpha)]; 0,0,0,1];
3 end

```

Listing A.15: Homogeneous transformation leg to toe

```

1 function ret = H_Leg_to_Toe(alpha, r, q)
2     ret = [R_r(q(4))', [-r*sin(-alpha);0;-r*cos(-alpha)]; 0,0,0,1];
3 end

```

Listing A.16: Homogeneous transformation leg to contact

```

1 function ret = H_Leg_to_Contact(r,q)
2     ret = [R_r(q(4))', [0;0;-r]; 0,0,0,1];
3 end

```

A-8 Dynamics

A-8-1 Newton-Euler

Listing A.17: Newton-Euler equations

```

1 function [pb_dd, wb_d] = NE(q, qd, f, Ip_C, Ip_H, Jb, m)
2     g = [0; 0; -9.81];
3     pb_mtrx = kron(ones(1,6), q(1:3)); % [Ip_B, Ip_B, ..., Ip_B]
4     wb = qd(4:6);
5
6     pb_dd = sum(f,2)./m + g;
7     wb_d = Jb\((sum(cross(Ip_C - pb_mtrx, f), 2) - cross(wb, Jb*wb)));

```

A-8-2 Newton-Euler for optimisation

Listing A.18: Newton-Euler equations in optimisation function

```

1 function res = NE_opt(q, qd, C, R_IC, phi, k, f, Ip_C, Jb, m, dt, roll, pitch,
    rolld, pitchd)
2     f_n = [0,0,0,0,0,0];
3     f_n(C) = f;
4     f_t = pacejka(f_n, k);
5     f_t_lat = pacejka_lateral(f_n, qd, R_IC, C);
6
7     IF = zeros(3, 6);
8     for i = C
9         IF(:,i) = R_IC{i} * [f_t(i); f_t_lat(i); f_n(i)];
10    end
11
12    g = [0; 0; -9.81];
13    pb_mtrx = kron(ones(1,6), q(1:3)); % [Ip_B, Ip_B, ..., Ip_B]
14    wb = qd(4:6);
15
16    pb_dd = sum(IF,2)./m + g;
17    wb_d = Jb\((sum(cross(Ip_C - pb_mtrx, IF), 2) - cross(wb, Jb*wb)));
18

```



```

19 qdd = [pb_dd; wb_d; 0;0;0;0;0;0];
20 qd = qd + qdd*dt;
21 q = q + qd*dt;
22
23 res = (qd(4) - rolld)^2 + (qd(5) - pitchd)^2;

```

A-9 Slip

A-9-1 Slip ratio

Listing A.19: Slip ratio calculation

```

1 function [k, v_C_ret, v_R_ret, v_rot_ret, v_B_ret] = slip_ratio(q, qd, Ip_C,
   Ip_H, Bp_C, Bp_H, R_IC, phi, phid, C, r)
2   v_C_ret = [0,0,0,0,0,0];
3   v_R_ret = [0,0,0,0,0,0];
4   v_B_ret = [0,0,0,0,0,0];
5   v_rot_ret = zeros(3,6);
6
7   k = [0,0,0,0,0,0];
8   for i = C
9     v_yaw = cross([qd(4);qd(5);qd(6)], Bp_C(:,i));
10    v_R = -r*phid(i);
11
12    v_B = R_IC{i}\qd(1:3);
13    v_H = R_IC{i}\(cross(qd(4:6), Ip_H(:,i)-q(1:3)) + qd(1:3));
14    v_rot = R_IC{i}\cross(qd(4:6), Ip_C(:,i)-q(1:3));
15
16    if phi(1,i) == 0 % Toe contact
17      v_R = -2*r*phid(i)*cos(phi(2,i));
18      v_C = v_B(1) + v_rot(1);
19
20      k(i) = (v_R - v_H(1))/(sign(v_H(1))*v_H(1));
21
22      k(i) = min(100, k(i)); % Avoid Inf answer
23      k(i) = max(-100, k(i));
24    else % Rolling contact
25      v_R = -2*r*phid(i)*cos(phi(1,i)/2)^2;
26      v_C = v_B(1) + v_rot(1);
27      k(i) = sign(v_R)*(v_R - (sign(v_C))*v_C)/((sign(v_C))*v_C);
28      k(i) = min(100, k(i)); % Avoid Inf answer
29      k(i) = max(-100, k(i));
30    end
31    v_C_ret(i) = v_C;
32    v_R_ret(i) = v_R;
33    v_B_ret(i) = v_B(1);
34    v_rot_ret(:,i) = v_rot;

```

```

35
36     if v_R == 0 && v_C == 0
37         k(i) = 0;
38     end
39 end
40 end

```

A-9-2 Pacejka (longitudinal)

Listing A.20: Pacejka model for longitudinal slip

```

1 function ret = pacejka(f_n, k)
2     b = 34.34;
3     c = 1.564;
4
5     for i = 1:6
6         if f_n(i) == 0
7             ret(i) = 0;
8             continue
9         end
10        d = 0.3798*f_n(i);
11        e = (b*0.0066*f_n(i) - tan(pi/(2*c)))/(b*0.0066*f_n(i)-atan(b*0.0066*
            f_n(i)));
12
13        ret(i) = d*sin(c*atan(b*k(i) - e*(b*k(i)-atan(b*k(i))))));
14    end
15 end

```

A-9-3 Pacejka (lateral)

Listing A.21: Pacejka model for lateral slip

```

1 function ret = pacejka_lateral(f_n, qd, R_IC, C)
2     b = 34.34;
3     c = 1.564;
4
5     ret = [0,0,0,0,0,0];
6     for i = C
7         if f_n(i) == 0
8             continue
9         end
10        v_C = R_IC{i}\qd(1:3);
11        if v_C(2) == 0
12            alpha = 0;
13        else
14            alpha = atan(-v_C(2)/v_C(1));
15        end
16    end

```

```

17     alpha_corr = alpha/(pi/2);
18
19     d = 0.3798*f_n(i);
20     e = (b*0.0066*f_n(i) - tan(pi/(2*c)))/(b*0.0066*f_n(i)-atan(b*0.0066*
        f_n(i)));
21
22     ret(i) = d*sin(c*atan(b*alpha_corr - e*(b*alpha_corr-atan(b*alpha_corr)
        )));
23 end
24 end

```

A-10 Kinematics

A-10-1 Leg angles

Listing A.22: Finding the angle ϕ_{min}

```

1 function ret = phi_(q)
2     theta_r = q(4);
3     theta_p = q(5);
4     theta_y = q(6);
5     theta = q(7:12);
6
7     ret = zeros(2,6);
8
9     for i = 1:6
10         X1 = sin(theta_r)*sin(theta_y)*cos(theta(i)) - cos(theta_r)*sin(theta_p)
            *cos(theta_y)*cos(theta(i)) - cos(theta_r)*cos(theta_p)*sin(theta(
            i));
11         X2 = sin(theta_r)*sin(theta_y)*sin(theta(i)) - cos(theta_r)*sin(theta_p)
            *cos(theta_y)*sin(theta(i)) + cos(theta_r)*cos(theta_p)*cos(theta(
            i));
12
13         a1 = atan(-X2/X1);
14         a2 = a1 - sign(a1)*pi;
15
16         derivative = -cos(a1)*X1 + sin(a1)*X2;
17         if a1 > 0
18             if derivative > 0    % Rolling contact
19                 ret(:,i) = [a1; a1];
20             else
21                 ret(:,i) = [pi; a2]; % Hip contact
22             end
23         else
24             if derivative > 0
25                 ret(:,i) = [0; a1]; % Toe contact
26             else

```

```

27         ret(:,i) = [a2; a2]; % Rolling contact
28     end
29 end
30 end
31 end

```

Listing A.23: Finding $\dot{\phi}_{min}$

```

1 function ret = phi_d(q,qd)
2     theta_r = q(4);
3     theta_p = q(5);
4     theta_y = q(6);
5     theta_r_d = qd(4);
6     theta_p_d = qd(5);
7     theta_y_d = qd(6);
8     theta_d    = qd(7:12);
9
10    ret = [0,0,0,0,0,0];
11    for i = 1:6
12        X = tan(theta_r)*sin(theta_y) - tan(theta_p*cos(theta_y));
13        X_d = theta_r_d*sin(theta_y)*(1+tan(theta_r)^2) - theta_p_d*cos(theta_y)
            *(1+tan(theta_p)^2) + theta_y_d*(tan(theta_r)*cos(theta_y) + tan(
            theta_p)*sin(theta_y));
14
15        ret(i) = X_d/(1+X^2) - theta_d(i);
16    end
17 end

```

A-10-2 Body angles

Listing A.24: Finding the roll angle θ_r

```

1 function theta_r = theta_r_estimate(q, z_i, z_o, L)
2     theta_r = asin((z_i - z_o)/L);
3 end

```

Listing A.25: Finding $\dot{\theta}_r$

```

1 function theta_rd = theta_rd_estimate(q, zd_i, zd_o, L)
2     theta_rd = (mean(zd_i) - mean(zd_o))/(cos(q(4))*L);
3 end

```

A-10-3 Body height estimates

Listing A.26: Estimating the z coordinate of the body

```

1 function [z, z_i, z_o] = z_estimate(q, C_i, C_o, phi, r)
2     z = [0,0,0,0,0,0];
3     z_i = [];
4     for i = C_i

```

```

5         if phi(1, i) == 0
6             z_i = [z_i, 2*r*cos(phi(2,i))*cos(q(4))];
7         elseif phi(1,i) == pi
8             z_i = [z_i, 0];
9         else
10            z_i = [z_i, (r+r*cos(phi(1,i)))*cos(q(4))];
11        end
12    end
13    z_o = [];
14    for i = C_o
15        if phi(1, i) == 0
16            z_o = [z_o, 2*r*cos(phi(2,i))*cos(q(4))];
17        elseif phi(1,i) == pi
18            z_o = [z_o, 0];
19        else
20            z_o = [z_o, (r+r*cos(phi(1,i)))*cos(q(4))];
21        end
22    end
23    z = mean([mean(z_i), mean(z_o)]);

```

Listing A.27: Estimating \dot{z}

```

1 function [zd, zd_i, zd_o] = zd_estimate(q, C_i, C_o, phi, phid, thetad_i,
    thetad_o, r)
2     zd = [0,0,0,0,0,0];
3     zd_i = [];
4     for i = C_i
5         if phi(1, i) == 0
6             zd_i = [zd_i, 2*r*thetad_i.g*sin(phi(2,i))];
7         elseif phi(1,i) == pi
8             zd_i = [zd_i, 0];
9         else
10            zd_i = [zd_i, r*thetad_i.g*sin(phi(1,i))];
11        end
12    end
13    zd_o = [];
14    for i = C_o
15        if phi(1, i) == 0
16            zd_o = [zd_o, 2*r*thetad_o.g*sin(phi(2,i))];
17        elseif phi(1,i) == pi
18            zd_o = [zd_o, 0];
19        else
20            zd_o = [zd_o, r*thetad_o.g*sin(phi(1,i))];
21        end
22    end
23    zd = mean([mean(zd_i), mean(zd_o)]);
24 end

```

Listing A.28: Estimating \ddot{z}

```

1 function zdd = zdd_estimate(C, phi, phid, r)
2     zdd = [0,0,0,0,0,0];
3     for i = C
4         if phi(1, i) == 0
5             zdd(i) = -2*r*phid(i)^2*cos(phi(2,i));
6         else
7             zdd(i) = -r*phid(i)^2*cos(phi(1,i));
8         end
9     end
10 end

```

A-10-4 Weight distribution

Listing A.29: Estimating the weight on the legs

```

1 function [fg, C] = weight_distribution(q, Ip_C, Ip_H, C, m)
2 pb = q(1:2);
3 fg = [0,0,0,0,0,0];
4 g = 9.81;
5 p = {};
6 for i = 1:6
7     p{i} = Ip_C(1:2, i);
8 end
9
10 if length(C) == 1
11     fg(C) = m*g;
12 end
13
14 if length(C) == 2
15
16     d1 = distanceBetweenPointAndPerpLine(p{C(1)}, p{C(1)}, p{C(2)});
17     d2 = distanceBetweenPointAndPerpLine(p{C(2)}, p{C(1)}, p{C(2)});
18     db = distanceBetweenPointAndPerpLine(pb, p{C(1)}, p{C(2)});
19
20     % Distances between centre and contact, to determine which side
21     % of the line the centre is.
22     db1 = distanceBetweenTwoPoints(pb, p{C(1)});
23     db2 = distanceBetweenTwoPoints(pb, p{C(2)});
24
25     if db1 > db2 % then centre is on the side of p2
26         f1 = m*g*(d2-db)/(d1+d2);
27         f2 = m*g*(d1+db)/(d1+d2);
28     else
29         f1 = m*g*(d2+db)/(d1+d2);
30         f2 = m*g*(d1-db)/(d1+d2);
31     end

```

```

32     fg(C) = [f1, f2];
33 end
34
35 if length(C) == 3
36     d1 = distanceBetweenPointAndLine(p{C(1)}, p{C(2)}, p{C(3)});
37     d2 = distanceBetweenPointAndLine(p{C(2)}, p{C(1)}, p{C(3)});
38     d3 = distanceBetweenPointAndLine(p{C(3)}, p{C(1)}, p{C(2)});
39     db1 = distanceBetweenPointAndLine(pb, p{C(2)}, p{C(3)});
40     db2 = distanceBetweenPointAndLine(pb, p{C(1)}, p{C(3)});
41     db3 = distanceBetweenPointAndLine(pb, p{C(1)}, p{C(2)});
42
43     f1 = db1/d1 * m*g;
44     f2 = db2/d2 * m*g;
45     f3 = db3/d3 * m*g;
46
47     fg(C) = [f1, f2, f3];
48 end
49
50 if length(C) > 3
51     r = zeros(1,length(C)); % Distance between centre and contact
52     for i = C
53         r(i) = distanceBetweenTwoPoints(p{i}, pb);
54     end
55     [~, idx] = sort(r);
56     C = sort(idx(end-2:end));
57
58     [fg, C] = weight_distribution(q, Ip_C, Ip_H, C, m);
59 end
60 end
61
62 function d = distanceBetweenPointAndLine(p0, p1, p2)
63 d = abs((p2(2)-p1(2))*p0(1) - (p2(1)-p1(1))*p0(2) + p2(1)*p1(2) - p2(2)*p1(1))/
    ...
64     sqrt((p2(2)-p1(2))^2 + (p2(1)-p1(1))^2);
65 end
66
67 function d = distanceBetweenTwoPoints(p1,p2)
68 d = sqrt((p1(1)- p2(1))^2 + (p1(2) - p2(2))^2);
69 end
70
71 function d = distanceBetweenPointAndPerpLine(p0, p1, p2)
72     x1 = p1(1);
73     x2 = p2(1);
74     y1 = p1(2);
75     y2 = p2(2);
76     a = x2 - x1;
77     b = y2 - y1;

```

```

78     c = -(0.5*(y2-y1)^2 + y1*(y2-y1) + (x2-x1)*(0.5*(x2-x1)+x1));
79
80     d = abs(a*p0(1) + b*p0(2) + c)/(sqrt(a^2 + b^2));
81 end

```

A-11 Contact points

A-11-1 Contact position

Listing A.30: Finds the potential contact position of a leg

```

1 function Ip_C = contact_pos(q, phi, R_IB, Bp_H, r)
2     Ip_B = q(1:3);
3     Ip_C = zeros(3,6);
4     theta = q(7:12);
5
6     for i = 1:6
7         L = [r*(cos(phi(1,i)) + 1);
8             0;
9             -r*sin(phi(1,i))];
10        Ip_C(:,i) = Ip_B + R_IB*(R(theta(i))*L + Bp_H(:,i));
11    end
12 end

```

A-11-2 Contact velocities

Listing A.31: Finds the velocities of each contact point

```

1 function v_C = contact_velocities(q,qd,R_IC,Ip_C,C)
2     v_C = zeros(3,6);
3
4     for i = C
5         v_C_body = R_IC{i}\qd(1:3);
6         v_C_rot = R_IC{i}\(cross(qd(4:6), Ip_C(:,i)-q(1:3)));
7
8         v_C(:,i) = v_C_body + v_C_rot;
9         v_C(3,i) = 0; % Contact is not moving away from the ground.
10    end
11 end

```

A-11-3 Update contact state

Listing A.32: Finds the current contact state

```

1 function [C, C_i, C_o] = update_contact_state2(phi, C_i, direction, r)
2     if direction == left
3         inner_legs = [1,3,5];
4     elseif direction == right

```



```

5         inner_legs = [2,4,6];
6     end
7
8     C_i_poss = setdiff(inner_legs, C_i);
9     z_i = hip_height(C_i, phi, r);
10    z_i_poss = hip_height(C_i_poss, phi, r);
11
12    if mean(z_i_poss >= z_i)
13        C_i = C_i_poss;
14    end
15
16    if direction == left
17        if C_i == 3
18            C_o = [2,6];
19        else
20            C_o = 4;
21        end
22    else
23        if C_i == 4
24            C_o = [1,5];
25        else
26            C_o = 3;
27        end
28    end
29
30    C = sort([C_i, C_o]);
31 end
32
33 function z = hip_height(i, phi, r)
34     if phi(1,i) == pi
35         z = 0;
36     elseif phi(1,i) == 0
37         z = 2*r*cos(phi(2,i));
38     else
39         z = (r*cos(phi(1,i)) + r);
40     end
41 end

```

A-12 Update leg speeds

Listing A.33: Updates leg speeds when touchdown or lift-off angle are reached

```

1 function thetad = update_leg_speeds(q, theta_i, theta_o, theta_d_i, theta_d_o,
   dir)
2 theta = q(7:12);
3 thetad = [0,0,0,0,0,0];
4

```

```

5  if dir == left
6      inner_legs = [1,3,5];
7      outer_legs = [2,4,6];
8  elseif dir == right
9      inner_legs = [2,4,6];
10     outer_legs = [1,3,5];
11 end
12
13 for i = inner_legs
14     thet = wrapTo2Pi(theta(i));
15     if thet >= theta_i.TD && thet < theta_i.L0
16         thetad(i) = theta_d_i.g;
17     else
18         thetad(i) = theta_d_i.f;
19     end
20 end
21 for i = outer_legs
22     thet = wrapTo2Pi(theta(i));
23     if thet >= theta_o.TD && thet < theta_o.L0
24         thetad(i) = theta_d_o.g;
25     else
26         thetad(i) = theta_d_o.f;
27     end
28 end
29
30 end

```

A-13 Save results

Listing A.34: Saves the results in a useful structure

```

1  function results = save_results(results, q, qd, qdd, f_n, f_t, f_g, IF, phi,
    phid, k, C, v_C, v_R, v_rot, v_B)
2      results.q(:,end+1) = q;
3      results.qd(:,end+1) = qd;
4      results.qdd(:,end+1) = qdd;
5      results.f_n(end+1, :) = f_n;
6      results.f_t(end+1, :) = f_t;
7      results.f_g(end+1, :) = f_g;
8      results.IF.x(end+1, :) = IF(1,:);
9      results.IF.y(end+1, :) = IF(2,:);
10     results.IF.z(end+1, :) = IF(3,:);
11     results.phi(end+1, :) = phi(1,:);
12     results.phi_(end+1, :) = phi(2,:);
13     results.phid(end+1, :) = phid;
14     results.k(end+1, :) = k;
15

```

```
16     C_ = [0,0,0,0,0,0];
17     C_(C) = 1;
18     results.C(end+1, :) = C_;
19
20     results.v_C(end+1, :) = v_C;
21     results.v_R(end+1, :) = v_R;
22     results.v_rot.x(:, end+1) = v_rot(1,:);
23     results.v_rot.y(:, end+1) = v_rot(2,:);
24     results.v_rot.z(:, end+1) = v_rot(3,:);
25     results.v_B(end+1, :) = v_B;
26 end
```

Bibliography

- [1] K. J. Waldron, V. J. Vohnout, A. Pery, and R. B. McGhee, "Configuration design of the adaptive suspension vehicle," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 37–48, 1984. DOI: [10.1177/027836498400300204](https://doi.org/10.1177/027836498400300204). [Online]. Available: <https://doi.org/10.1177/027836498400300204>.
- [2] U. Saranlı, "Dynamic locomotion with a hexapod robot," PhD thesis, University of Michigan, Ann Arbor, MI, USA, 2002, ISBN: 0-493-88647-8.
- [3] Robotblog, *Zebro (zes benen robot), rescue robot tu delft*. [Online]. Available: <https://www.robotblog.nl/zebro-rescue-robot/> (visited on 06/02/2019).
- [4] TU Delft Robotics Institute, *Zebro*. [Online]. Available: <https://tudelftroboticsinstitute.nl/robots/zebro> (visited on 06/02/2019).
- [5] K. C. Galloway, G. C. Haynes, B. D. Ilhan, A. M. Johnson, R. F. Knopf, G. Lynch, B. Plotnick, M. White, and D. E. Koditschek, "X-rhex : A highly mobile hexapedal robot for sensorimotor tasks," 2010.
- [6] A. M. Johnson, M. T. Hale, G. C. Haynes, and D. E. Koditschek, "Autonomous legged hill and stairwell ascent," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, Nov. 2011, pp. 134–142. DOI: [10.1109/SSRR.2011.6106785](https://doi.org/10.1109/SSRR.2011.6106785).
- [7] N. Neville, M. Buehler, and I. Sharf, "A bipedal running robot with one actuator per leg," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 848–853. DOI: [10.1109/ROBOT.2006.1641815](https://doi.org/10.1109/ROBOT.2006.1641815).
- [8] International Festival of Technology, *International festival of technology - lunar zebro*. [Online]. Available: <https://ifot-delft.com/en/events/lunar-zebro> (visited on 03/01/2019).
- [9] L. Hoes, *Decizebro*. [Online]. Available: <https://www.lottehoes.nl/lisanne-bouwde-mee-aan-de-decizebro-de-zesbenige-oliedomme-insectrobot/decizebro-7/> (visited on 07/11/2019).

- [10] W. Suriana, “Turning of a legged robot via a switching max-plus linear system: Simulation and implementation,” Master’s thesis, Delft University of Technology, Delft, the Netherlands, 2017. [Online]. Available: <http://resolver.tudelft.nl/uuid:4bca33e3-7c4b-4c46-8908-96481d6d7acc>.
- [11] T. Edridge, S. Krab, L. de Vries, and T. van Wijk, “A kinematic model of the zebro robot,” Bachelor’s thesis, Delft University of Technology, Delft, the Netherlands, 2018.
- [12] M. Hildebrand, “Symmetrical gaits of horses,” *Science*, vol. 150, no. 3697, pp. 701–708, 1965, ISSN: 0036-8075. DOI: [10.1126/science.150.3697.701](https://doi.org/10.1126/science.150.3697.701). [Online]. Available: <http://science.sciencemag.org/content/150/3697/701>.
- [13] G. Lopes, T. van den Boom, B. D. Schutter, and R. Babuska, “Modeling and control of legged locomotion via switching max-plus systems,” *IFAC Proceedings Volumes*, vol. 43, no. 12, pp. 382–387, 2010, 10th IFAC Workshop on Discrete Event Systems, ISSN: 1474-6670. DOI: <https://doi.org/10.3182/20100830-3-DE-4013.00063>.
- [14] U. Saranli, M. Buehler, and D. E. Koditschek, “Rhex: A simple and highly mobile hexapod robot,” *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001. DOI: [10.1177/02783640122067570](https://doi.org/10.1177/02783640122067570). [Online]. Available: <https://doi.org/10.1177/02783640122067570>.
- [15] M. Aghbelagh, “A template-based control architecture for dynamic legged locomotion,” Ph.D. thesis, Delft University of Technology, Delft, the Netherlands, 2016. [Online]. Available: <https://doi.org/10.4233/uuid:baeca8ab-8534-4006-bb4a-d7c36921150a>.
- [16] W. Yu, J. O. Y. Chuy, J. E. G. Collins, and P. Hollis, “Analysis and experimental verification for dynamic modeling of a skid-steered wheeled vehicle,” *IEEE Transactions on Robotics*, vol. 26, no. 2, pp. 340–353, Apr. 2010, ISSN: 1552-3098. DOI: [10.1109/TR0.2010.2042540](https://doi.org/10.1109/TR0.2010.2042540).
- [17] J. Ginsberg, *Engineering Dynamics*. Cambridge University Press, 2007. DOI: [10.1017/CB09780511805899](https://doi.org/10.1017/CB09780511805899).
- [18] K. M. Lynch and F. C. Park, *Modern robotics: mechanics, planning, and control*. Cambridge University Press, 2018.
- [19] “@blackpawn”, *Point in triangle test*. [Online]. Available: <https://blackpawn.com/texts/pointinpoly/> (visited on 04/27/2020).
- [20] M. Hutter, R. Siegwart, and T. Stastny, *Robot dynamics lecture notes*, 2017. [Online]. Available: https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2017/RD_HS2017script.pdf.
- [21] L. Collenteur, G. van Egmond, K. Hennissen, T. Janssen, and B. Verweij, “Development of a slip model for the leg-ground interaction of a zebro mobile robot,” Bachelor’s thesis, Delft University of Technology, Delft, the Netherlands, 2019.
- [22] S. Caron, *Newton-euler equations*. [Online]. Available: <https://scaron.info/teaching/newton-euler-equations.html> (visited on 04/01/2019).
- [23] D. Sunday, *Lines and distance of a point to a line*. [Online]. Available: http://geomalgorithms.com/a02-_lines.html (visited on 03/22/2020).

-
- [24] H. B. Pacejka, "Chapter 2 - basic tire modeling considerations," in *Tire and Vehicle Dynamics (Third Edition)*, H. B. Pacejka, Ed., Third Edition, Oxford: Butterworth-Heinemann, 2012, pp. 59–85, ISBN: 978-0-08-097016-5. DOI: <https://doi.org/10.1016/B978-0-08-097016-5.00002-4>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/B9780080970165000024>.
- [25] H. Pacejka, *Tire and Vehicle Dynamics*, 3rd ed. Elsevier Ltd., 2012.