Modeling, identification and stability of humanoid robots

with a case study on humanoid robot TUlip

P.W.M. van Zutven

DCT 2009.100

Masters's thesis

Coach: dr. D. Kostic MSc.[1]

Supervisor: prof. dr. H. Nijmeijer[1]

Committee: dr. D. Kostic MSc.[1]
prof. dr. H. Nijmeijer [1]
dr. ir. A. de Kraker[1]
prof. dr. ir. P.P. Jonker[2]

[1]Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics and Control Group

[2]Delft University of Technology
Faculty of Mechanical, Maritime and Materials Engineering (3ME)
Biomechanical Engineering Department

Eindhoven, October 2009

TUlip

What we know is a drop, what we don't know is an ocean.
Sir Isaac Newton (4 January 1643 - 31 March 1727)

Abstract

With the primary goal to give the research field of humanoid robotics a new impulse, the three technical universities in the Netherlands (Delft, Eindhoven and Twente) together with Philips have developed the humanoid robot TUlip. This robot is intended as key experimental platform for research on walking, dynamical analysis, control and artificial intelligence of humanoid robots. Using TUlip as a test bed, the work presented in this thesis focuses on the modeling and identification of humanoid robots, as well as on analysis of stability of bipedal walking.

For modeling, different methods are critically reviewed: Newton-Euler, Lagrange-Euler and TMT. By making use of Denavit Hartenberg convention for modeling robot kinematics, we contribute an automatic algorithm for derivation of Lagrange-Euler equations of motions for a general humanoid robot. This algorithm is instantiated on TUlip. Furthermore, conditions for ground contact and impact expressions are derived, which are implemented in a numerical simulation together with the equations of motion. For numerical integration, the event detection and time-stepping methods are considered. Arguments in favor of the event detection method are provided and this method is used in the implementation of the model of TUlip.

The model parameters of the robot TUlip are identified in dynamic experiments. To facilitate the dynamic identification a regressor form is derived using an automated algorithm from the equations of motion together with the actuator dynamics, friction and dynamics of the drive train. Persistently exciting trajectories are optimized using this regressor and these trajectories are used in experiments on the robot TUlip. These experiments provide estimates of the model parameters. The estimated parameters are validated in experiments.

Finally, the most commonly used stability criteria have been critically argued. Among several candidates, two the most conventional ones, namely the zero moment point and Poincaré map have been applied for design and stability analysis of walking gaits for a model of a planar humanoid robot. The results of this analysis show that these methods are neither sufficient nor necessary to guarantee stability of bipedal walking. Consequently, further developments are needed in the research on formal design and stability analysis of walking gaits.

Samenvatting

Met als primair doel het onderzoek op het gebied van humanoïde robots een nieuwe impuls te geven, hebben de drie technische universiteiten van Nederland (Eindhoven, Delft en Twente) samen met Philips de humanoïde robot TUlip ontworpen. Deze robot is bedoeld als experimenteel platform voor onderzoek naar lopen, dynamische analyses, regeling en kunstmatige intelligentie van humanoïde robots. Met TUlip als test opstelling, richt deze thesis zich op het modelleren en identificeren van humanoïde robots en het in kaart brengen van de huidige stabiliteitscriteria van loopbewegingen.

Er zijn in dit rapport verschillende modeleermethoden kritisch bekeken: Newton-Euler, Lagrange-Euler en TMT. Met behulp van de Denavit Hartenberg conventie om de kinematica van robots te modelleren is een automatisch algoritme ontworpen om de Lagrange-Euler bewegingsvergelijkingen van een willekeurige humanoïde robot af te leiden. Dit algoritme is gebruikt voor een model van TUlip. Verder zijn er condities voor grond-contact en impactberekeningen afgeleid, die met de bewegingsvergelijkingen zijn geïmplementeerd in een numerieke simulatie. Voor de numerieke integratie zijn de event detectie en time-stepping methode beschouwd en argumenten voor de event detectie methode zijn gegeven voor de numerieke implementatie van het model van TUlip.

De modelparameters van de robot TUlip zijn in dynamisch experimenten geïdentificeerd. Voor de dynamische experimenten is van de bewegingsvergelijkingen een regressor vorm afgeleid op een geautomatiseerde manier, waarbij actuator dynamica, wrijving en de dynamica van de aandrijvingen zijn meegenomen. Op deze regressor zijn voldoende exciterende trajectories geoptimaliseerd die gebruikt zijn in experimenten op de robot TUlip. De experimenten resulteren in geschatte parameters, die verder zijn gevalideerd met experimenten.

Tot slot zijn de meest gangbare stabiliteitscriteria kritisch beargumenteerd en onder de verschillende kandidaten zijn de twee meest voorkomende, namelijk zero moment point en Poincaré, toegepast bij het ontwerp en de stabiliteitsanalyse van loopbewegingen van een twee dimensionaal model van een humanoïde robot. De resultaten hiervan laten zien dat deze criteria niet voldoende, maar ook niet noodzakelijk zijn om de stabiliteit van een loopbeweging te analyseren. Dit geeft aan dat verder onderzoek nodig is op het gebied van ontwerp en de stabiliteit van humanoïde robots.

Preface

This thesis is the outcome of a master's project carried out within the Dynamics and Control Technology group at the Mechanical Engineering faculty of the Eindhoven University of Technology, Eindhoven, the Netherlands.

First of all I would like to thank my supervisor dr. Dragan Kostic MSc. for his supervisory support, help and guidance during the project. I would like to thank him for his enthusiasm, especially in Graz during the RoboCup, as well as during the weekly Humanoid Robotics meetings. Furthermore I would like to thank my colleagues from the Humanoid Robotics lab for the great working ambiance: Maarten Dekker, Atilla Filiz, Richard Kooijman and everyone else who walked in regularly. Also thanks to all members of the Humanoid Robotics group for their ideas and pleasant times during meetings and in Graz. Special thanks to Stijn Boere from this team, for the fruitful discussions during our lunch breaks. Finally, I would like to thank prof. dr. Henk Nijmeijer for supervising this project and the high quality feedback during our monthly meetings.

I have learned a lot during this project, I gained some experience about doing a project on my own. I enjoyed doing this project, it is fascinating to see the complexity of the human body and the ease with which we are able to walk. Modeling this behavior is not straightforward and at this moment still at its infancy. However, great progress has been made the last few years and I expect the first humanoid robots to appear in society in the coming decade.

Pieter van Zutven, October 2009

Contents

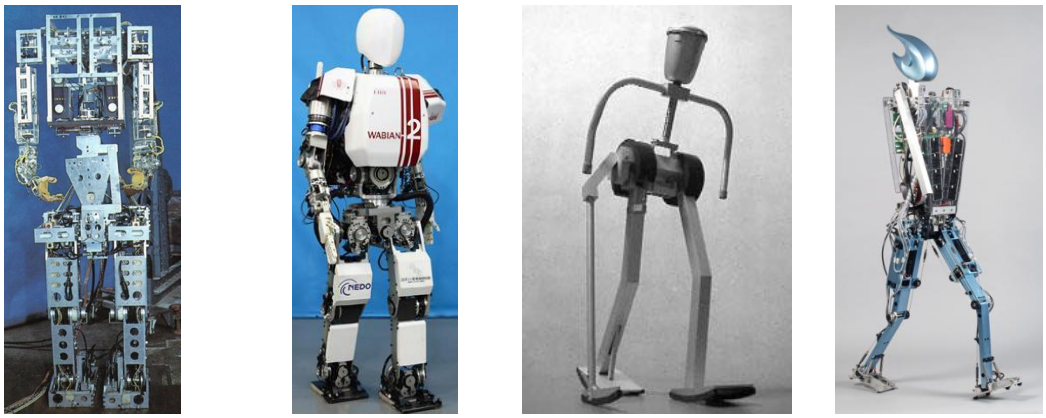
Chapter 1

Introduction

Ever since the Czech Karel Čapek introduced the word robot in his play Rossum's Universal Robots people are imagining how the world would look like when robots were an element of our daily life. This can be concluded from the numerous, mostly science fiction, movies that are made, like Star Wars, A.I., RoboCop, the Terminator or the Transformers or from popular books from Isaac Asimov for example. Together with the increasing interest of people on the field of robotics, the development of robots started. A robot is usually seen as an electro mechanical machine that can move around, operate a mechanical limb, sense and manipulate its environment, or exhibit intelligent behavior, especially behavior which mimics humans or other animals. Originally, the behavior was not of high intelligence, but nowadays robots have evolved to machines that are stronger than humans and can do tasks quicker and with higher accuracy. That's what makes the current field of robotics so incredibly fascinating. There are different kinds of robots in the field, from welding robotic arms in the automotive industry to tiny nanorobots. One specific group of robotic machines is classified as humanoid robots.

In this work, humanoid robots are considered. Humanoid robotics is an emerging technology that will become part of our daily life in the coming decades. Researchers and engineers are already developing humanoid robots that feature various human-like characteristics, since these robots should substitute people in a variety of tasks in industry, household, services, care, etc. These robots really do look like humans and they move in a human like way. It is generally accepted that walking on two legs should be the underlying principle of humanoid locomotion. Humanoid robots that walk on two legs are called bipeds.

The development of humanoid robots started in the seventies when Miomir Vukobratović proposed his theorem of the zero moment point to explain biped locomotion [33]. This theorem mathematically explained for the first time how bipeds are able to walk stably and it is still used in the majority of humanoid robots nowadays. Almost parallel to Vukobratović research, the Waseda University from Tokyo, showed their first humanoid robot Wabot-1, also using the concept of zero moment point to perform a stable gait. This robot is shown in figure 1.1(a). In the eighties another type of walking was introduced by Tad McGeer: passive dynamic walking [9]. He showed that a very simple biped could walk down a slope without any actuation. The energy from the gravity completely balances the loss of energy due to friction. The main reason why a group of researchers gained interest in dynamic walking



(a) Wabot-1, the first zero moment point based humanoid robot

(b) Wabian-2R, currently the most advanced zero moment point based humanoid robot

(c) Museon walker, a passive dynamic walking humanoid robot

(d) Flame, currently the most advanced dynamic walking humanoid robot

Figure 1.1: Examples of humanoid robots

is that it is energy efficient and the gaits are more human like. Namely, in the meantime, a lot of zero moment point humanoid robots were introduced and the main disadvantages were discovered. Although the concept of zero moment point is quiet easy to use, the gaits are slow, highly energy demanding and not human like. Currently, there are two different approaches to design walking gaits humanoid robots, one with the concept of zero moment point and another with the concept of dynamic walking in mind. The most advanced zero moment point based robot at this moment is most likely Wabian from the Waseda University, which is shown in figure 1.1(b). The most advanced dynamic walking humanoid robot is Flame from the Delft University of Technology, shown in figure 1.1(d).

The research on humanoid robots has increased significantly over the last decades, but still a lot of work is in front of us. Namely, the major challenges are highly non linear dynamics of humanoid robots and the fact that these robots should cope with various environments and diversity of tasks. As we will see in this work, a general mathematical formalism about stability that works in all those areas does not yet exist. Even more, a general algorithm to design a stable gait for every situation does not yet exist. Humanoids are enormously complex and that is why the research spreads out over all kinds of scientific fields, from mechanics to electronics, from modeling to control and from informatics to biomechanics.

On one hand, this work focuses on modeling and identification of humanoid robots, and on another, on analysis of stability of bipedal walking. The first objective of this assignment is to deliver a mathematical formalism suitable for dynamical modeling and identification of humanoid bipeds. This formalism should be applied on the on the humanoid robot TUlip, the most advanced humanoid robot in the Netherlands. The second objective is a formal stability analysis of bipedal walking.

The conventional formalisms to derive the equations of motions such as Newton-Euler [30], Lagrange-Euler [6] and TMT [26] are perfectly applicable to humanoid robots. Unfortunately

the complexity of such robot makes it very difficult to derive equations by hand. We will show that application of Denavit Hartenberg convention [27] for modeling robot kinematics can facilitate automatic derivation of Lagrange-Euler equations of motion. In particular, we propose a mathematical formalism for automatic modeling of a general humanoid robot, including impacts and constraints due to contacts with the ground.
Besides correct model algebra, practical usability of the robot model for system analysis, gait design and control purposes, also depends on accuracy of the model parameters. In this report we provide procedures for estimation of the model parameters in dynamic experiments. The first procedure takes care of automatic representation of the robot dynamics in a form which facilitates parameter estimation (so called regressor form). Accomplishing this step in an automatic way greatly reduces the system identification time and represents an important contribution of this work. The second procedure ensures the optimal design of the identification experiments and accurate parameter estimation. Quality of both procedures is successfully demonstrated on TUlip.
Finally, four methods to analyze the stability of bipedal walking, namely zero moment point, Poincaré return map, angular momentum and capture point, are explained shortly and critically reviewed. The first two are used to design and analyze two different stable gaits for a model of a planar bipedal humanoid robot with six actuated degrees of freedom. We discuss strengths and weaknesses of both methods and point out importance of further research on systematic design and analysis of stability of bipedal walking.

The report is build up as follows. In chapter 2 methods for modeling dynamics of humanoid robots are explained. An automatic modeling procedure is proposed and illustrated on the humanoid robot TUlip. Chapter 3 deals with system identification. First, we provide procedures for automatic model representation in regressor form and optimal design of identification experiments. Then we demonstrate these procedures on TUlip. In chapter 4 several stability criteria are reviewed and the most conventional ones are applied on a model of a planar biped. Finally, in chapter 5 we draw conclusions and give recommendations for future work.

Chapter 2

Modeling a humanoid robot

2.1 Introduction

A humanoid robot is a mechatronics system which looks like a human and resembles kinematics of the human body. Often, humanoid robots feature bipedal locomotion. Such robots contain two lower limbs and optional torso. There exists a wide variance in the complexity of bipedal locomotion systems. Some try to mimic the human body in an extraordinary way, like Honda's Asimo or Wabian from the Waseda University of Technology and others do not look like humans, but are able to walk or even make a salto like Spring Flamingo of the Massachusetts Institute of Technology.
To facilitate design, analysis and control of humanoid robots we can derive the models of the robot kinematics and dynamics. The models can be used for offline and online trajectory planning, control design and monitoring of robot states that are critical for stability and safety of robot operation.
In general a humanoid robot can be modeled by a series of rigid bodies interconnected by joints, the multi body model. The equations that describe the motions of these rigid bodies under influence of external forces can be derived in different ways and all together they form the multi body dynamics of the system. Just as in the real systems, also in modeling there is a wide variance in complexity. A humanoid multi body model can perform a stable gait with only three point masses connected with massless links and no actuation [9], whereas there also are models that mimic the human body and have plenty of degrees of freedom [33]. In general, complex models can more accurately describe kinematics and dynamics of a robot at the cost of higher computation effort and knowledge of more robot parameters. Therefore, one should always find a balance between the complexity, accuracy and solvability of a model. Overall, modeling locomotion systems is difficult because they contain switching behavior between different support modes, i.e. single support and double support, impact equations and multiple constraints. In this chapter we first present three methods to model robot dynamics: Newton-Euler [30], Lagrange-Euler [6] and the TMT method [3, 26]. Then we introduce Denavit Hartenberg convention for assignment of coordinate frames to the robot joints [27]. This convention facilitates automatic formulation of the robot dynamics using the Lagrange-Euler method. Application of this method for modeling the humanoid robot TUlip will be explained in more detail, including modeling of ground contact constraints and impacts.

2.2 Modeling methods

2.2.1 Newton-Euler

The classical Newton equations are the fundamental idea behind dynamic modeling of mechanical systems. The equations describe the acceleration of a particle under the influence of a certain net force, this is Newton's second law of motion. Newton's second law of motion for a particle with mass dm exhibiting a velocity $\underline{\dot{x}}$ under influence of a net force $d\underline{F}$ is given by:

$$d\underline{F} = \frac{d}{dt}\left(dm\underline{\dot{x}}\right) \tag{2.1}$$

Furthermore Newton states that if particle A is exerting a force on particle B that particle B at the same time is exerting the opposite force with the same magnitude on particle A, also known as Newton's third law of motion. A rigid body consists of multiple particles constrained to each other by constraint forces. The general equation of motion for a rigid body can be derived using Newton's second and third laws of motion and the so called d'Alembert principle. This principle states that the sum of constraint forces between the particles in a body can be disregarded because these forces do not contribute to the acceleration of the body. In other words, the constraint forces between the particles don't perform work inside a body. The d'Alembert principle for a body containing N particles i, performing a virtual displacement $\delta\underline{x}$, is given as:

$$\sum_{i=1}^{N}\left(d\underline{F}_i - \frac{d}{dt}\left(dm_i\underline{\dot{x}}_i\right)\right)\delta\underline{x} = 0 \tag{2.2}$$

Because the constraint forces in the body can be disregarded the general equation of motion can be derived by summing the net applied forces of (2.1) over all particles in a body A and assuming a constant mass in time.

$$\int_A d\underline{F} = \int_A dm\underline{\ddot{x}} \tag{2.3}$$

$$\underline{F} = m\underline{\ddot{x}} \tag{2.4}$$

In the same way the rotational dynamics of a body are defined in the principle of Euler. This principle describes the angular acceleration $\underline{\ddot{\varphi}}$ of a particle with inertia I under the influence of a net moment $\underline{M}$:

$$\underline{M} = I\underline{\ddot{\varphi}} \tag{2.5}$$

Using (2.3) and (2.5) it is possible to derive the equations of motion of a body or multiple interconnected bodies. In a multi body system the Newton-Euler equations of motion need to be derived using the forces and moments acting on every individual body. To model the interconnection of the bodies algebraic constraints in the joints are added to the set of the differential equations. In this way a set of differential algebraic equations is derived. In most cases this set can not be solved analytically, but has to be solved numerically. A major disadvantage of this approach is that most solvers are not capable to numerically integrate the differential algebraic equations with high accuracy. A phenomenon called drift often occurs in solving differential algebraic equations, which means that the error accumulates in time. An example of how to derive the equations of motion using the Newton-Euler method can be found in Appendix A.1.1.

2.2.2 Lagrange-Euler

To avoid the numerical problems when the Newton-Euler method is used, it is possible to use the Lagrange-Euler method. This method uses a minimum set of coordinates needed to describe the system and fulfill the constraints, the so called generalized coordinates. The basis to derive the equations of motion for series of bodies in a Lagrangian way is energy. Every body in the series contains energy, generally consisting of kinetic and potential energies. The kinetic energy of a body with mass m and inertia I is described by:

$$K = \frac{1}{2} m \underline{\dot{x}}^2 + \frac{1}{2} I \underline{\dot{\varphi}}^2 \tag{2.6}$$

where $\underline{\dot{x}}$ and $\underline{\dot{\varphi}}$ are linear and angular velocities respectively. The potential energy consists of different parts, for example gravitational energy or elastic energy. The gravitational energy of the body at height h in a gravity field of acceleration g is:

$$P = mgh \tag{2.7}$$

while the elastic energy stored in a spring with linear stiffness k and elongation y is given by:

$$P = \frac{1}{2} k y^2. \tag{2.8}$$

As mentioned before, the generalized coordinates q describe the system completely. If the number of generalized coordinates equals the degrees of freedom of the system then the generalized coordinates are called independent and the system is said to be unconstrained. The generalized coordinates can be introduced such as to describe the positions of the centers of mass of all bodies in the system as $\underline{x}$.

$$\underline{x} = \underline{x}\left(\underline{q}\right) \tag{2.9}$$

$$\delta \underline{x} = \frac{\partial \underline{x}}{\partial \underline{q}} \delta \underline{q} \tag{2.10}$$

Using this notation and the d'Alembert principle, given in (2.2), the equations of motion of one or multiple bodies can be derived. The Lagrange-Euler equation is given by:

$$\frac{d}{dt} \frac{\partial K}{\partial \underline{\dot{q}}} - \frac{\partial K}{\partial \underline{q}} + \frac{\partial P}{\partial \underline{q}} = \underline{Q}_{nc}^T \tag{2.11}$$

where $\underline{Q}_{nc}$ denotes the non conservative forces containing damping and friction forces and moments. The complete derivation can be found in literature [6, 30]. The advantage of deriving the equations of motion using the Lagrange-Euler method is that it uses an independent set of generalized coordinates. In this way there is no need to describe all positions and orientations of all bodies in the system together with the constraints imposed by the joints. These equations of motion are therefore easier to solve without numerical troubles. Nevertheless a disadvantage is that with large and complex models it can be hard to determine all energies, friction and damping and to compute all partial derivatives analytically. An example of how to derive the equations of motion using the Lagrange-Euler method can be found in Appendix A.1.2.

2.2.3 TMT

As we have seen in the previous parts there are advantages and disadvantages in the Newton-Euler method as well as in the Lagrange-Euler method. Especially for complex systems with many degrees of freedom it could be useful to combine the advantages of both methods to be able to derive the equations of motion in an easy way and avoid numerical problems when executing the resulting models. This is possible for systems with only holonomic constraints and is basically done in a transformation of constraint dynamics to unconstrained dynamics or from dependent generalized coordinates to independent generalized coordinates [30]. This method is sometimes referred to as the TMT method [3, 26]. First write the positions and orientations of the centers of mass $\underline{x}$ of all bodies in the system as a function of the independent generalized coordinates $\underline{q}$ and differentiate them with respect to time to find analytically the Jacobian that describes the Cartesian velocities as a function of the generalized coordinates.

$$\begin{aligned} \underline{x} &= \underline{T}\left(\underline{q}\right) \\ \dot{\underline{x}} &= \frac{\partial \underline{T}}{\partial \underline{q}}\dot{\underline{q}} \\ \ddot{\underline{x}} &= \frac{\partial \underline{T}}{\partial \underline{q}}\ddot{\underline{q}} + \frac{\partial}{\partial \underline{q}}\left(\frac{\partial \underline{T}}{\partial \underline{q}}\dot{\underline{q}}\right)\dot{\underline{q}} \end{aligned} \tag{2.12}$$

Also the following relation holds between the Cartesian virtual displacements $\delta\underline{x}$ and independent generalized displacements $\delta\underline{q}$:

$$\delta\underline{x} = \frac{\partial \underline{T}}{\partial \underline{q}}\delta\underline{q} \tag{2.13}$$

Combining d'Alembert principle, (2.12) and (2.13) results in the following expression, with $\underline{M}$ the mass matrix and $\underline{F}$ a matrix with all applied forces and torques on the system.

$$\frac{\partial \underline{T}}{\partial \underline{q}}\delta\underline{q}\left(\underline{F} - \underline{M}\left(\frac{\partial \underline{T}}{\partial \underline{q}}\ddot{\underline{q}} + \frac{\partial}{\partial \underline{q}}\left(\frac{\partial \underline{T}}{\partial \underline{q}}\dot{\underline{q}}\right)\dot{\underline{q}}\right)\right) = 0 \tag{2.14}$$

$$\frac{\partial \underline{T}}{\partial \underline{q}}\delta\underline{q}\left(\underline{F} - \underline{M}\left(\frac{\partial \underline{T}}{\partial \underline{q}}\ddot{\underline{q}} + \underline{G}\right)\right) = 0 \tag{2.15}$$

with $\underline{G} = \frac{\partial}{\partial \underline{q}}\left(\frac{\partial \underline{T}}{\partial \underline{q}}\dot{\underline{q}}\right)\dot{\underline{q}}$.
Since $\delta\underline{q}$ is independent and all virtual displacements are allowed, (2.14) can be written as:

$$\frac{\partial \underline{T}}{\partial \underline{q}}\left(\underline{F} - \underline{M}\left(\frac{\partial \underline{T}}{\partial \underline{q}}\ddot{\underline{q}} + \underline{G}\right)\right) = 0 \tag{2.16}$$

Furthermore rearranging (2.16) results in the equations of motion in independent generalized coordinates:

$$\frac{\partial \underline{T}}{\partial \underline{q}}\underline{M}\frac{\partial \underline{T}}{\partial \underline{q}}\ddot{\underline{q}} = \frac{\partial \underline{T}}{\partial \underline{q}}\underline{F} - \frac{\partial \underline{T}}{\partial \underline{q}}\underline{MG} \tag{2.17}$$

The TMT method (2.17) can be used to calculate the equations of motion of a body or multiple interconnected bodies in an efficient way. Namely, it takes advantage of the Newton-Euler method to derive the equations of motion in an easy way without having to think of complex

energy relations. Furthermore it takes advantage of the Lagrange-Euler method because it uses generalized coordinates to avoid numerical problems when solving stiff differential algebraic equations.

This method is very easy and works perfect for 2D models, even in very complex situations, however with the current set up there is a problem when modeling 3D systems. Namely in 3D, two things differ from 2D. Firstly, in 2D the centers of mass can only rotate in the 2D plane, which means that only the corresponding inertia has to be taken into account. In 3D one deals with an inertia matrix with configuration dependent inertia elements, which might be hard to find by hand for complex systems. Secondly, for the same reason, the analytical Jacobian in (2.12) now depends on three rotations in different planes, which, if used to calculate the kinetic energy, does not guarantee a correct result. A geometric Jacobian should be used instead and it might be possible to adapt the basic TMT formulation such that it uses a geometric Jacobian. Nevertheless, an attempt to change the TMT method will not be discusses in this work, because in the next section a different modeling formalism will be presented that can automatically derive the equations of motion. An example of how to derive the equations of motion using the TMT method can be found in Appendix A.1.3.

2.2.4 Lagrange-Euler formulation of robot dynamics based on Denavit Hartenberg convention

To derive Lagrange-Euler equations of motion in an automatic way, we can make use of Denavit Hartenberg convention for description of robot kinematics [8]. The Denavit Hartenberg convention is a commonly used method to assign coordinate frames to different links in a robotic arm. This method ensures that the position and orientation of each frame can be described by only four parameters, which simplifies the kinematic analysis and allows us to derive the equations of motion using the kinetic and potential energy as in the Lagrange-Euler equations [27]. The convention is normally used to model robotic arms with a fixed base and one tip at the end. Humanoid robots, however, do not have a fixed base and in general have two legs which can be seen as tips. This means that we, in some sense, have to modify the standard approach of deriving the equations of motion with the Denavit Hartenberg convention. The fixed base can be introduced with a virtual kinematic chain. This chain connects the inertial frame to the torso of the robot. This virtual chain consists of three prismatic and three revolute joints that describe the position and orientation of the torso of the robot. The links in the virtual arm should be massless not to influence the motions of the robot. Two legs can be added by the derivation of two separate models each representing one of the legs of the robot. These two models can be merged together to get the complete equations of motion.

The Denavit Hartenberg convention assigns the coordinate frames in a smart way so that the position and orientation of each frame i with respect to the previous frame $i-1$ can be represented using a so called homogeneous transformation matrix of the following form:

$$\underline{A}_i^{i-1} = \begin{bmatrix} cos\,(\theta_i) & -sin\,(\theta_i)\,cos\,(\alpha_i) & sin\,(\theta_i)\,sin\,(\alpha_i) & a_i cos\,(\theta_i) \\ sin\,(\theta_i) & cos\,(\theta_i)\,cos\,(\alpha_i) & -cos\,(\theta_i)\,sin\,(\alpha_i) & a_i sin\,(\theta_i) \\ 0 & sin\,(\alpha_i) & cos\,(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \underline{R}_i^{i-1} & \underline{o}_i^{i-1} \\ \underline{0}^T & 1 \end{bmatrix} \quad (2.18)$$

where $\underline{R}_i^{i-1} \in \mathbb{R}^{3\times3}$ and $\underline{o}_i^{i-1} \in \mathbb{R}^{3\times1}$ are the orientation matrix and position vector of frame i with respect to frame $i-1$ respectively. The orientation and position of frame i with respect to the base frame, which is needed for the derivation of the equations of motion, can now be derived using the following equation:

$$\underline{T}_i^0 = \prod_{j=1}^{i} \underline{A}_j^{j-1} = \begin{bmatrix} \underline{R}_i^0 & \underline{o}_i^0 \\ \underline{0}^T & 1 \end{bmatrix} \tag{2.19}$$

where $\underline{T}_i^0$ is called the homogeneous transformation matrix from the base frame to frame i. These matrices describe the position and orientation of each frame relative to the base frame. From this homogeneous transformation matrix the position of the centers of mass of each link with respect to the base frame can be derived.

$$\underline{o}_{ci}{}^0 = \underline{o}_i^0 + \underline{R}_i^0 \underline{o}_{ci}{}^i \tag{2.20}$$

where $\underline{o}_{ci}^i$ is a vector containing the positions of the center of mass of link i relative to frame i.

Velocity kinematics is also needed to derive the equations of motion, since the kinetic energy is calculated with the linear and angular velocity of each center of mass in the system. The linear velocity $\underline{v}_i^0$ and angular velocity $\underline{\omega}_i^0$ of a frame i in a system of m links (i.e. $i \leq m$) can be described by two Jacobians respectively:

$$\underline{v}_i^0 = \underline{J}_{vi}{}^0 \underline{\dot{q}} \tag{2.21}$$

$$\underline{\omega}_i^0 = \underline{J}_{\omega i}{}^0 \underline{\dot{q}} \tag{2.22}$$

where $\underline{J}_{vi}{}^0 \in \mathbb{R}^{3\times m}$ is the linear velocity Jacobian, $\underline{J}_{\omega i}{}^0 \in \mathbb{R}^{3\times m}$ is the angular velocity Jacobian and $\underline{\dot{q}} \in \mathbb{R}^{m\times1}$ is the time derivative of the vector of joint variables. Due to the Denavit-Hartenberg convention, these Jacobians can easily be derived. Namely, in this convention, the z-axis of each frame is always the axis of actuation. Thus frame i is always translated or rotated with respect to $\underline{z}_{i-1}^{i-1} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ with joint variable q_i. Keeping this in mind, we may derive the Jacobians geometrically. For a frame i the Jacobians are:

$$\underline{J}_{vi}{}^0 = \begin{bmatrix} \underline{J}_{vi,1}{}^0 \cdots \underline{J}_{vi,i}{}^0 & \underline{0} \end{bmatrix} \tag{2.23}$$

$$\underline{J}_{\omega i}{}^0 = \begin{bmatrix} \underline{J}_{\omega i,1}{}^0 \cdots \underline{J}_{\omega i,i}{}^0 & \underline{0} \end{bmatrix} \tag{2.24}$$

where $\underline{0} \in \mathbb{R}^{3\times m-i}$ and $\underline{J}_{vi,j}{}^0$ and $\underline{J}_{\omega i,j}{}^0$, with $j = 1 \cdots i$, are the linear and angular velocity Jacobians of frame j respectively. These Jacobians depend on the type of joint and can be found geometrically in the following way. For prismatic joints the following two equations hold [27]:

$$\underline{J}_{vi,j}{}^0 = \underline{z}_{j-1}^0 \tag{2.25}$$

$$\underline{J}_{\omega i,j}{}^0 = \underline{0} \tag{2.26}$$

and similarly, for revolute joints the following two equations hold [27]:

$$\underline{J}_{vi,j}{}^0 = \underline{z}_{j-1}^0 \times \left(\underline{o}_{ci}{}^0 - \underline{o}_{j-1}^0 \right) \tag{2.27}$$

$$\underline{J}_{\omega i,j}{}^0 = \underline{z}_{j-1}^0 \tag{2.28}$$

where $\underline{z}_{j-1}^0 = \underline{R}_{j-1}^0 \underline{z}_{j-1}^{j-1}$. The complete Jacobian of link i can finally be represented as follows:

$$\underline{J}^0 = \begin{bmatrix} \underline{J}_{vi}{}^0 \\ \underline{J}_{\omega i}{}^0 \end{bmatrix} = \begin{bmatrix} \underline{J}_1^0 \cdots \underline{J}_i^0 & \underline{0} \end{bmatrix} \tag{2.29}$$

The linear and angular velocity Jacobians describe the mapping of the linear and angular velocities of the tip of a robotic arm with m joints in joint coordinates to base frame coordinates. In the case of a humanoid robot the complete Jacobian consists of three parts. The first part describes a six degree of freedom virtual kinematic chain attached to the torso of the robot. This chain is physically not present on a real robot, but it is used to model the ability of a humanoid robot to move freely in space. The second and third part of the Jacobian describe the right and left leg respectively, each with m links. The complete Jacobian that describes the right leg from the base frame to the right toe, including the virtual arm, can be written as:

$$\underline{J}_{BR}{}^0 = \begin{bmatrix} \underline{J}_{VA}{}^0 & \underline{J}_R{}^0 \end{bmatrix} \tag{2.30}$$

where

$$\underline{J}_{VA}{}^0 = \begin{bmatrix} \underline{J}_1^0 \cdots \underline{J}_6^0 \end{bmatrix} \tag{2.31}$$

and

$$\underline{J}_R{}^0 = \begin{bmatrix} \underline{J}_{R7}^0 \cdots \underline{J}_{Rm+6}^0 \end{bmatrix} \tag{2.32}$$

For the left side of the robot we can do the same resulting in the complete Jacobian that describes the left leg from the base frame to the left toe, including the virtual arm:

$$\underline{J}_{BL}{}^0 = \begin{bmatrix} \underline{J}_{VA}{}^0 & \underline{J}_L{}^0 \end{bmatrix} \tag{2.33}$$

where

$$\underline{J}_L{}^0 = \begin{bmatrix} \underline{J}_{L7}^0 \cdots \underline{J}_{Lm+6}^0 \end{bmatrix} \tag{2.34}$$

The Jacobian for the right side and the Jacobian for the left side can be merged together to get the Jacobian for the full robot:

$$\underline{J}_T{}^0 = \begin{bmatrix} \underline{J}_{VA}{}^0 & \underline{J}_R{}^0 & \underline{J}_L{}^0 \end{bmatrix} \tag{2.35}$$

The equations of motion of the robot can be derived using the well known Lagrange-Euler equations. These equations are derived in section 2.2.2 using the kinetic and potential energy of every link in the system:

$$\frac{d}{dt}\frac{\partial K}{\partial \underline{\dot{q}}} - \frac{\partial K}{\partial \underline{q}} + \frac{\partial P}{\partial \underline{q}} = \underline{Q}_{nc}^T \tag{2.36}$$

where K and P represent the kinetic and potential energy respectively and $\underline{Q}_{nc}$ is a vector with non conservative forces like damping, friction and applied torques. The kinetic and potential energy can be derived using the kinematic equations derived in the previous section. The kinetic energy for a system or part of a system with links $i = 1 \cdots n$ with mass m_i and a inertia matrix $\underline{I}_i$ evaluated around a coordinate frame parallel to frame i but whose origin is at the center of mass:

$$K = \frac{1}{2}\underline{\dot{q}}^T \left(\sum_{i=1}^{n} m_i \underline{J}_{vi}{}^{0T} \underline{J}_{vi}{}^0 + \underline{J}_{\omega i}{}^{0T} \underline{R}_i^0 \underline{I}_i \underline{R}_i^{0T} \underline{J}_{\omega i}{}^0 \right) \underline{\dot{q}} \tag{2.37}$$

$$= \frac{1}{2}\underline{\dot{q}}^T \underline{D} \underline{\dot{q}} \tag{2.38}$$

$$= \sum_{i,j}^{n} d_{ij} \underline{\dot{q}}_i \underline{\dot{q}}_j \tag{2.39}$$

where $\underline{D}$ is called the inertia matrix. The potential energy of a system or part of a system with links $i = 1 \cdots n$ in a gravity field $\underline{g}$ with its center of mass at position $\underline{o}_{ci}^{\,0}$ is:

$$P = \sum_{i=1}^{n} m_i \underline{g}^T \underline{o}_{ci}^{\,0} \tag{2.40}$$

The equations of motion can be calculated with (2.36), but we can further simplify the algorithm. If it's assumed that there are no external forces acting on the system but the applied torques, it can easily be seen that (2.36) can be rewritten as:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \tag{2.41}$$

where $L = K - P$ and q_i and τ_i are the joint variable and joint torque of link i. So we assume there is no damping or friction acting in the joint and $\underline{Q}^{nc}$ only contains joint torques. Combining (2.37), (2.40) and (2.41), the Lagrange-Euler equations for each link i can be written as:

$$\sum_{j=1}^{n} d_{ij}\underline{\ddot{q}}_j + \sum_{k=1}^{n}\sum_{j=1}^{n}\left(\frac{\partial d_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial d_{kj}}{\partial q_i}\right)\dot{q}_k\dot{q}_j + \frac{\partial P}{\partial q_i} = \tau_i \tag{2.42}$$

which in matrix form can be written as:

$$\underline{D}\underline{\ddot{q}} + \underline{C}\underline{\dot{q}} + \underline{G} = \underline{\tau} \tag{2.43}$$

where $\underline{D} \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\underline{C} \in \mathbb{R}^{n\times n}$ is called the Coriolis and centrifugal matrix and $\underline{G} \in \mathbb{R}^{n\times 1}$ is the gravity vector, n is the number of links in the system. More details can be found in [27]. Note that in the case of a humanoid robot $n = 6 + 2m$, where m is the number of links in one leg, and that (2.35) should be used for the Jacobians to calculate $\underline{D}$ and $\underline{C}$.

The Denavit Hartenberg convention is perfectly suitable for modeling robotic arms and as we showed it can also be used to model a humanoid robot. The derivation described in this section is easy and can be executed numerically without problems. Therefore we use this method derive a model of TUlip in the next section. An example of how to derive the equations of motion using the Denavit Hartenberg convention can be found in Appendix A.1.4.

2.3 Modeling TUlip

In this section we will apply the Lagrange-Euler method from section 2.2.4 on the humanoid robot TUlip. First we will introduce this robot and then formulate its kinematics and dynamics. We will address aspects of ground contact and impact and discuss issues of numerical implementation.

2.3.1 Introducing TUlip

TUlip is a humanoid robot, which is developed primarily as a research object, but it also participates in the yearly RoboCup tournament. A humanoid robot is a robot resembling the human body, it doesn't only look physically like a human body, but it also mimics its

Figure 2.1: Photo of TUlip by David Joosten and a render of TUlip by Tomas de Boer

movements. Humanoid robots can be used to substitute humans in difficult or dangerous work. Evolution took about four billion years to develop into a intelligent human creature, the goal of RoboCup is to achieve this in fifty years. Annually humanoid robots participate in this soccer tournament and the goal is to defeat the human world champion by the year 2050. Through this goal scientists are motivated and they will put a lot of effort in the development of humanoid robots. Challenges are modeling, control and stability analysis of humanoid robots, which incorporate highly nonlinear multimode dynamics, in all environments and under a diversity of tasks. Hence defeating the human world champion is not the only purpose of this tournament, it will also lead to more breakthroughs in all kinds of scientific fields in the coming years. For example new resources will be developed for disabled people and in hospitals humanoid robots can help the personnel with heavy tasks, such as lifting patients. The increasing obsolescence will lead to issues in space and the capacity of caretakers. In the future humanoid robots will be able to take care of aging people. These robots should look like human beings for the comfort of those people. A study on the behavior of children with a robot dog shows that people react in a same way to robot dogs as living ones [20]. Already before fully autonomous humanoid robots will appear in society, parts of these robots and their functionalities will be integrated in care taking systems.

In the Netherlands the research on humanoid robots already shows some results, especially in the field of dynamic walking [15, 36]. To increase the research on humanoid robotics in the Netherlands, Philips and the three universities of technology under the umbrella of 3TU, Eindhoven University of Technology, Delft University of Technology and the University of Twente, collaborate and develope the humanoid robot TUlip. This robot already participated as goal keeper in the RoboCup 2008 and 2009 tournaments and will also participate in the RoboCup 2010 tournament [18]. A photo and render of this robot is shown in figure 2.1.

TUlip is a robot with fourteen degrees of freedom, six in each leg among which three are

located in the hip joint. Consequently TUlip is able to walk in every direction in a human like way. Furthermore TUlip is equipped with arms to stand up when it has fallen onto the ground and a head consisting of two miniature cameras as eyes together with a vestibular sensor. The cameras are movable so that TUlip can track the ball. The body of TUlip and all its limbs are made of black anodized aluminium components, protected by body armor made of soft foam and a tough aramid composite material. This makes the robot very robust to damage. Finally TUlip's 'brain' is a 1 GHz computer with 256 MB ram. This computer and all actuators are fed by a 24 V lithium polymer battery pack which enables the robot to walk autonomously for almost half an hour.

One of the goals of the Dutch robotics team is to let TUlip walk at the RoboCup 2010 tournament. To achieve this, first of all a model is needed to be able to investigate the dynamic behavior of a walking robot, to design suitable controllers and trajectories and to analyze the stability of walking. In this chapter a model of TUlip is derived using the Denavit Hartenberg convention.

2.3.2 Model

TUlip is modeled as a point mass model with thirteen links from which ten can be actuated actively. The model consists of a torso (index 5), two side bodies (indices 6 and 13), two flanks (indices 7 and 14), two hips (8 and 15), two thighs (9 and 16), two shins (10 and 17), two ankles (indices 11 and 18) and two feet (indices 12 and 19) with each four contact points. The degrees of freedom are divided in the following way: two revolute degrees of freedom in both ankles (one passive), one revolute degree of freedom in both knees and three revolute degrees of freedom in both hips. Each link in the model has a length l. The absolute position of the centers of mass of each link with respect to the previous joint are described by $\underline{c}$. Each link has a mass m, concentrated in the center of mass point, and an inertia matrix $\underline{I}$ for the possible rotations. A drawing of the model is shown in figure 2.2.

The coordinate frames are assigned using the Denavit Hartenberg convention and the equations of motion are derived according to section 2.2.4. A virtual kinematic chain with six degrees of freedom is attached to the torso of the robot to let it move freely in Cartesian space. This arm is shown in figure 2.3. Impacts are calculated when the robot makes contact with the ground, these events can occur on the toes and heels of the robot. When such events occur unilateral constraints are added to the equations of motion of the model introducing normal forces on the ground. The friction on the ground is assumed to be high enough to prevent the robot from slipping and impacts occur completely inelastic. The mathematical derivation of the impacts and constraints is described in the next section.

2.3.3 Ground contact and impact

To investigate a walking motion on the model of TUlip, the model should be able to make contact with the ground. The model of TUlip has four contact points per foot, two in front of its ankles representing toes and two behind its ankles representing its heels. If one of these points is in contact with the ground a constraint should be added to the equations of motions, since this point may not penetrate the ground. A schematic drawing of the impact is shown in figure 2.4. We can use the Lagrange multiplier theorem [30] to constrain the motions such

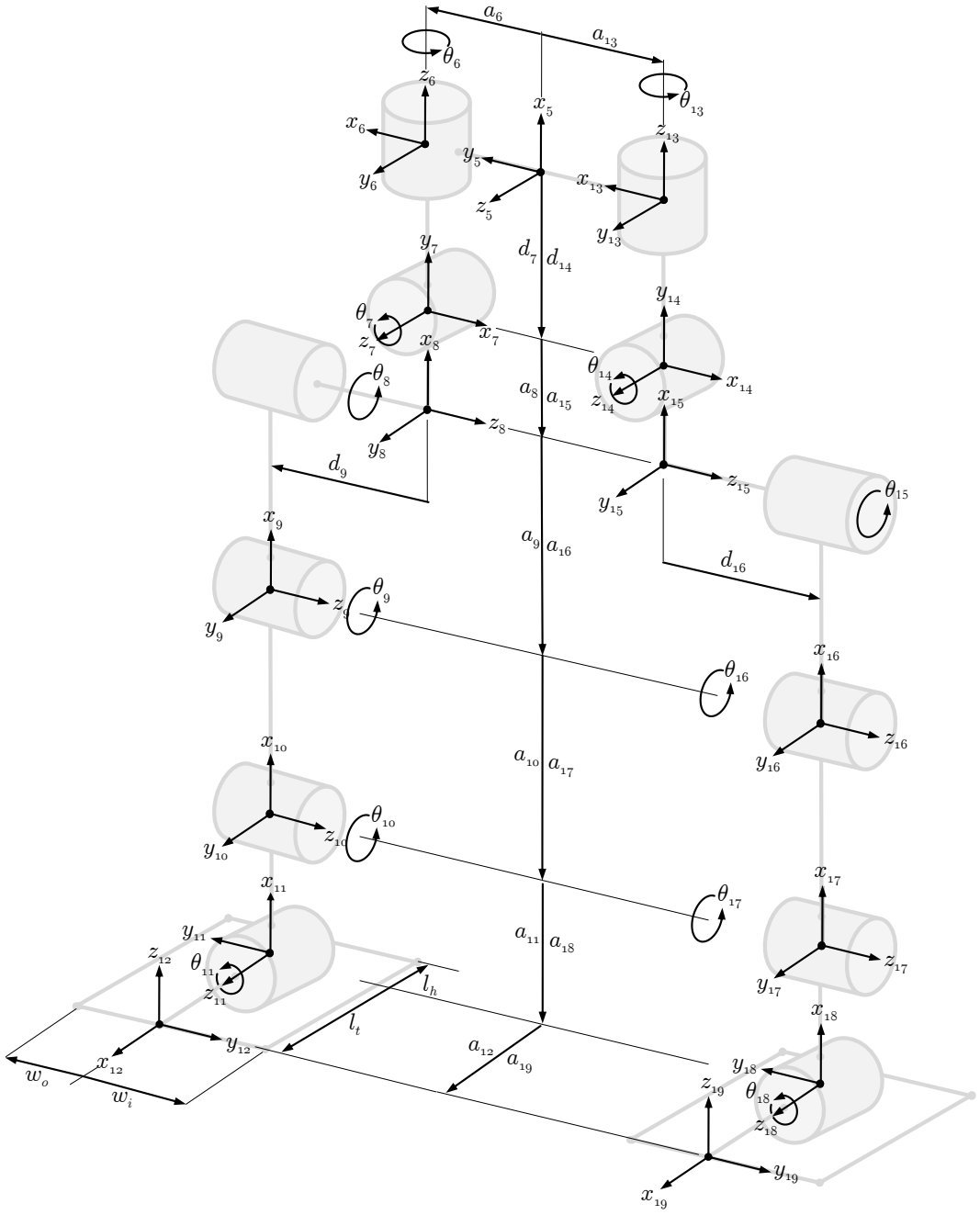


Figure 2.2: Denavit Hartenberg convention of point mass model

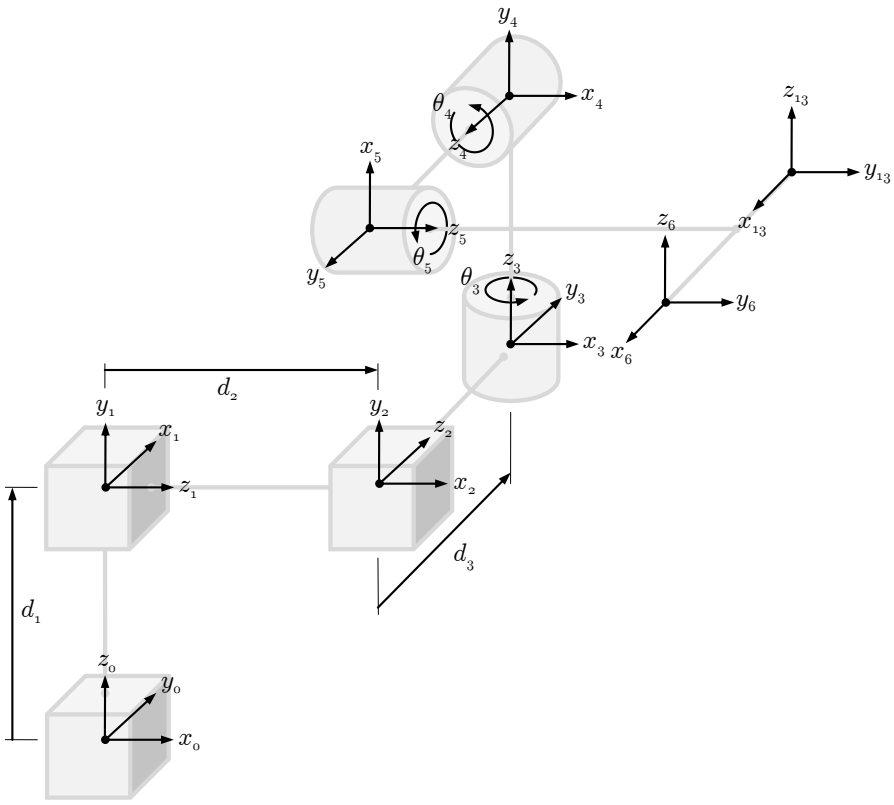


Figure 2.3: Denavit Hartenberg convention of virtual kinematic chain

that no point passes the ground. These ground constraints are holonomic position constraints that can be written as follows:

$$\underline{h}\left(\underline{q}\right) = \underline{0} \tag{2.44}$$

Where $\underline{h}\left(\underline{q}\right)$ is the position of the contact points as a function of the state of the system. These constraints may only be active when a contact point touches the ground, so $\underline{h} \geq \underline{0}$. If a contact point touches the ground, the constraint becomes active and a ground reaction force $\underline{\lambda}$ is introduced, which also may only be positive, since the robot does not stick to the ground. Therefore they form a so called complementarity condition:

$$\underline{h} \geq \underline{0}, \quad \underline{\lambda} \geq \underline{0}, \quad \underline{h}^T \underline{\lambda} = 0 \tag{2.45}$$

Which also can be written as orthogonality condition:

$$\underline{0} \leq \underline{h} \perp \underline{\lambda} \geq \underline{0} \tag{2.46}$$

To add these constraints to the Lagrange-Euler equations we need the variation $\delta\underline{h}$ of a variation $\delta\underline{q}$:

$$\delta\underline{h} = \underline{W}^T \delta\underline{q} \tag{2.47}$$

where $\underline{W}^T = \frac{\partial \underline{h}}{\partial \underline{q}}$. According to the Lagrange multiplier theorem we can combine this position constraint with the equations of motion derived in (2.43):

$$\begin{cases} \underline{D}\ddot{\underline{q}} + \underline{C}\dot{\underline{q}} + \underline{G} & = \underline{\tau} + \underline{W}\underline{\lambda} \\ \underline{W}^T \dot{\underline{q}} & = \underline{0} \end{cases} \tag{2.48}$$

To solve this set of equations, normally, the constraint is written on acceleration level:

$$\frac{\partial}{\partial t}\left(\underline{W}^T \dot{\underline{q}}\right) = \underline{W}^T \ddot{\underline{q}} + \left(\frac{\partial \underline{W}^T}{\partial t} + \frac{\partial \underline{W}^T \dot{\underline{q}}}{\partial \underline{q}}\right) \dot{\underline{q}} \tag{2.49}$$

In this case, the holonomic constraint is not dependent on time, so $\frac{\partial \underline{W}^T}{\partial t} = 0$ and we will define $\underline{w} =: \frac{\partial \underline{W}^T \dot{\underline{q}}}{\partial \underline{q}}$. Rewriting (2.48) gives:

$$\begin{bmatrix} \underline{D} & -\underline{W} \\ \underline{W}^T & \underline{0} \end{bmatrix} \begin{bmatrix} \ddot{\underline{q}} \\ \underline{\lambda} \end{bmatrix} + \begin{bmatrix} \underline{C} \\ \underline{w} \end{bmatrix} \dot{\underline{q}} = \begin{bmatrix} \underline{\tau} \\ \underline{0} \end{bmatrix} \tag{2.50}$$

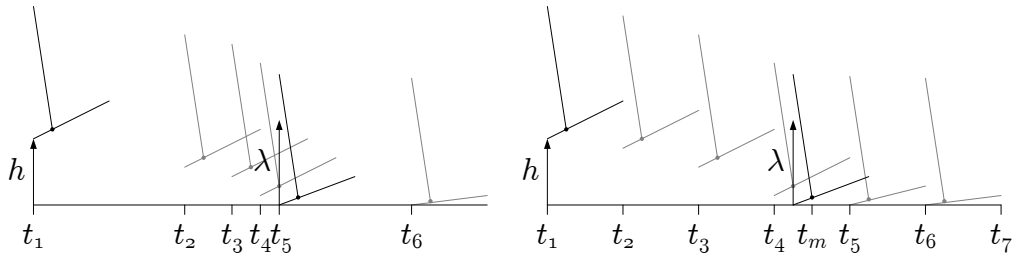
This differential algebraic equation (DAE) can be solved numerically during simulations. When active, the constraints make sure that the contact points can never exceed the floor plane.

If a foot of the robot comes in contact with the ground, an impact occurs. Newton's impact law is used to model these impacts. It relates the velocity of the contact point just after impact ($\underline{W}^T \dot{\underline{q}}^+$) with the velocity just before ($\underline{W}^T \dot{\underline{q}}^-$) through a restitution coefficient e:

$$\underline{W}^T \dot{\underline{q}}^+ = -e\underline{W}^T \dot{\underline{q}}^- \tag{2.51}$$

Furthermore, the law of conservation of momentum should also hold during impacts:

$$\underline{D}\dot{\underline{q}}^+ = \underline{D}\dot{\underline{q}}^- \tag{2.52}$$



(a) Event detection method (b) Time stepping method

Figure 2.4: Two different numerical integration methods suitable with contact dynamics

Now we can combine Newton's impact law (2.51), the conservation of momentum (2.52) and the Lagrange multiplier theorem for the constraint forces to complete the impact equations:

$$\begin{bmatrix} \underline{D} & -\underline{D} \\ \underline{W}^T & \underline{0} \end{bmatrix} \begin{bmatrix} \underline{\dot{q}}^+ \\ \underline{\dot{q}}^- \end{bmatrix} = \begin{bmatrix} \underline{W\lambda} \\ -e\underline{W\dot{q}}^- \end{bmatrix} \tag{2.53}$$

The impact equations (2.53) should be computed every time an impact occurs to find the joint velocities after the impact. After the impact the continuous equations of motion (2.50), if necessary with active constraints, can be solved.

In the model of TUlip we assume a completely inelastic impact, so the restitution coefficient $e = 0$ and the corresponding constraint becomes active immediately. We also assume that the friction between the feet of TUlip and the ground is high enough to prevent the feet from slipping, so there are no friction components implemented in the constraint equations. To conclude, if $\underline{h} > \underline{0}$ the model can move freely in space by solving the equations of motion (2.43), but as soon as a contact point comes in contact with the ground ($\underline{h} = \underline{0}$) the impact equations (2.53) need to be solved. This automatically introduces a constraint ($\underline{h} = \underline{0}$ and $\underline{\lambda} \geq \underline{0}$) and the equations of motion become constrained (2.50). Finally, the contact force can vanish ($\underline{\lambda} = \underline{0}$) as a result of the motion, so that the constraint can be released again ($\underline{h} \geq \underline{0}$) or other constraints can become active.

2.3.4 Numerical implementation

The equations of motion, unconstrained (2.43) as well as constrained (2.50), are highly non linear and can not be solved analytically. Therefore we need to implement them numerically to simulate the model. The switching hybrid behavior of the model due to impacts and constraints becoming active and inactive is very difficult to solve. There are basically two ways to numerically solve differential equations with impacts and unilateral constraints: the event detection method and the time stepping method [19, 23].

In the event detection method the continuous dynamics is integrated numerically using a classical integration scheme until a certain event is detected. This event could be an impact or a release of a contact point. Because of this event, the equations of motion change. Constraints become active or inactive and one needs to take this into account by numerically solving the active set of equations of motion. A certain function can be used to calculate new initial conditions for the next continuous part of the simulation until again an event

occurs. Each time an event occurs the continuous numerical integration is stopped and a discrete event situation is calculated. One needs to keep track of all active and inactive constraints during the simulation by introducing a binary vector $\underline{p}$ with a length equal to the number of all possible constraints. If a constraint is active, the corresponding element of $\underline{p}$ is 1, otherwise it is 0. At every time step the vector $\underline{p}$ should be checked completely to see which constraints need to be added to the equations of motion. If a constraint is inactive, the constraint function $\underline{h}$ is monitored to check if a contact point touches the ground. Then the corresponding elements of $\underline{p}$ are set to 1, taking into account that not all constraints on one foot (three per contact point i.e. one for each Cartesian direction) can be active, because then the system would become over-determined, resulting in a singular set of equations of motion. If a constraint is inactive, the corresponding ground reaction force $\underline{\lambda}$ is monitored to check if a contact point is released from the ground. Then the corresponding elements of $\underline{p}$ are set to 0. In this way it is always guaranteed that the proper constraints are used every time step during the continuous part of the numerical integration of the equations of motion. A schematic drawing of the event detection method is shown in figure 2.4(a). Finally we remark that there are numerical integration schemes that can handle events efficiently if the number of events is not huge.

The time stepping method works in another way, it takes the forces acting on a system into account in an integral way over every time step. This means that it doesn't separate impulsive and finite forces. The equations of motion are rewritten into so called measure differential equations, which are not on the level of forces, but on the level of momenta. In this way only positions and velocities have to be computed, accelerations do not have to be computed and are allowed to be infinite due to impacts. At the beginning of every time step it is determined if contact points are in touch with the ground at the midpoint t_m of the time step. If there are contact points in touch with the ground, the contact problem is solved for the entire time step resulting in the positions and velocities at the end of the time step. It is not necessary to stop the simulation when events occur, because all forces are taken into account through their momenta in an integral way every time step. That means the simulation can always continue no matter how many discontinuities are present in a time step. A schematic drawing of the time stepping method is shown in figure 2.4(b).

The benefit of the time stepping method in comparison with the event detection method is that it is very efficient in simulating systems with a huge number of events. The event detection method has to stop for every event that occurs, while the time stepping method takes them into account in a integral way in a single time step. There are even systems known that exhibit infinitely many events in finite time span, the so called Zeno behavior. The event detection method would not be able to integrate such system, while the time stepping method is able to approximate the result due to such behavior. Nevertheless, a disadvantage of the time stepping method is that it integrates in fixed time steps. In the neighborhood of an event, mostly small time steps are required to obtain an accurate solution, whereas in smooth parts of the systems equations big time step can be used to obtain the same accuracy. So for the time stepping method to work, the time steps should be small enough to cope with all events, but this means that also the smooth parts are integrated with small time steps, which is certainly not necessary. The commonly used event detection methods are able to integrate with variable time steps, so it is possible to integrate fast in smooth parts, but reduce the time steps near events. Overall, if there is not a huge number of events present in a system

it is better to use an event detection algorithm, otherwise use the time stepping method.

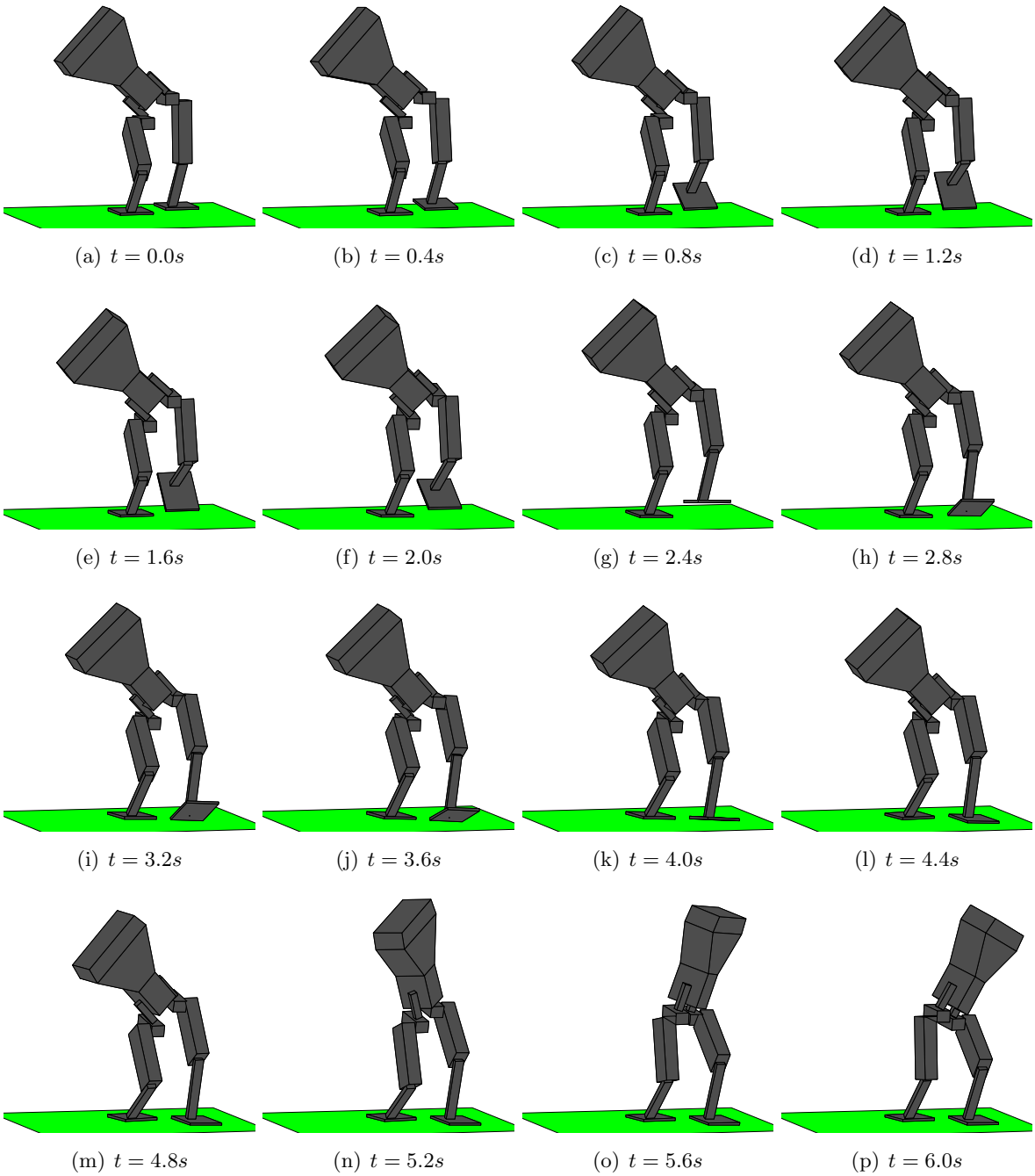
The model of TUlip is simulated using the event detection method for a couple of reasons. Firstly, the simulation time is lower due to the variable step size that can be used. Secondly, event detection algorithms are already implemented in Matlab, the program we will use to simulate our model. The function *ode15s* is used to simulate the model, it automatically takes care of events and decreases its step sizes in less smooth parts of the system. The final reason is that the model doesn't contain a huge number of events, because the impacts are modeled as completely inelastic and after an impact the contact point is immediately constrained to the ground. This prevents the contact point from generating infinitely many events in finite time, so that the event detection method can be used.

The analytical derivation of the equations of motion results in very long expressions for the dynamics. This is a problem since it increases the computational effort drastically. The implementation of the equations of motion of a full model of TUlip could therefore be problematic. A solution is to implement the derivation in an iterative numerical way instead of deriving the equations analytically.

2.3.5 Simulation results

The implemented model of TUlip can be simulated using Matlab. Overall it is very difficult to find a stable gait for a humanoid robot, but it appears to be even more difficult to find a stable gait for this model due to a number of reasons. First of all, the center of mass of TUlip is positioned high above the ground, which decreases the number of possible stable gaits. Furthermore, in the current set up, TUlip has five actuated degrees of freedom per leg, while at least six is needed to allow stable walk in any direction. This further decreases the number of possible stable gaits. Nevertheless, we were able to find one gait using a so called inverse differential kinematic method [7]. The designed gait is based on the zero moment point principle [33], which will be explained later on in chapter 4. The designed gait prescribes all joints of TUlip, such that it makes a stable step. The gait is implemented in the model as reference trajectory and the PD controllers calculate the required torques to realize this trajectory. Snapshots of the model of TUlip performing one step in a simulation are depicted in figure 2.5.

In the simulation the model performs one step with its left leg. It starts with the weight of the upper body positioned above the right foot. The inclination of the upper body is very large because, TUlip's center of mass is positioned very high and TUlip is, at the moment, lacking one actuated degree of freedom in its ankles. In order to be stable, the absence of one degree of freedom forces the right leg to be always perpendicular to the ground and the upper body to be inclined heavily. The performed gait starts with the pre-swing phase, where the left foot is lifted from to ground. At $t = 1.6s$ the left foot completely releases from the ground and the swing phase starts. At $t = 3.2s$ the post-swing phase starts when the left foot hits the ground. The double support phase starts when both feet are in full contact with the ground at $t = 4.4s$. Then the robot starts to shift its center of mass position from the right foot to the left to be able to start the next step. As can be seen, it has to make a complex movement because of the absence of one actuated degree of freedom each ankle.



(a) $t = 0.0s$ (b) $t = 0.4s$ (c) $t = 0.8s$ (d) $t = 1.2s$

(e) $t = 1.6s$ (f) $t = 2.0s$ (g) $t = 2.4s$ (h) $t = 2.8s$

(i) $t = 3.2s$ (j) $t = 3.6s$ (k) $t = 4.0s$ (l) $t = 4.4s$

(m) $t = 4.8s$ (n) $t = 5.2s$ (o) $t = 5.6s$ (p) $t = 6.0s$

Figure 2.5: Snapshots of the model of TUlip performing one step

2.4 Conclusion

In this chapter we explained the most commonly used modeling methods: Newton-Euler, Lagrange-Euler and TMT. We proposed the Denavit Hartenberg convention to model robot kinematics and used this with the Lagrange-Euler method to derive the equations of motion of a humanoid robot. It is shown that this formalism is perfectly usable to model humanoid robots in an automatic way. The Denavit Hartenberg convention describes the placement of the coordinate frames such that the smallest possible equations of motion can be derived. This derivation is based on the Lagrange-Euler method to avoid numerical problems during implementation, which is a known drawback of the Newton-Euler method. The usability of this automatic formalism is illustrated on the humanoid robot TUlip. The TMT method could be usable if current set up would be rewritten for 3D cases by implementing geometric Jacobians. Furthermore, ground contact constraints and impact equations are derived that should be added to the equations of motion when the robot makes contact with the ground. Finally it is shown how the set of equations can be implemented numerically using an event detection or time stepping solver. The model of TUlip is implemented using an event detection algorithm because the computational effort is lower, the algorithms are already implemented in Matlab and the normal behavior of the robot does not introduce a huge number of events, because of the inelastic impact assumption. The major problem with the implementation of the model of TUlip is that the equations of motion are huge. The matrices contain a very large number of expressions which causes a long simulation time. This problem could be solved by using an iterative numerical procedure.

Chapter 3

Identification of a humanoid robot

3.1 Introduction

After a model of a dynamic system has been derived it is of high importance to assign the model parameters such that the model resembles the dynamic system as close as possible. Mostly the parameters consist of geometrical and inertial ones. While geometrical parameters are known in most cases, inertial parameters are not always available, since these can not be measured directly. If not directly measurable, we get these parameters using system identification techniques. Basically, the identification of the inertial parameters can be divided into two methods, namely static and dynamic identification methods.

Static methods measure the inertial parameters while the system does not perform any motion. For example weighting the mass of a link is a static measurement. This kind of identification is simple and accurate, but it is in general not possible to identify all parameters. Furthermore it can be a problem when the system is already mounted together and when links are interconnected and influencing each other. Static identification methods on a humanoid robot are described in [5].

Dynamic identification methods estimate the parameters of a system dynamically, with the system in motion. The input to the system is such that all dynamics is excited and can be measured. The system output is measured and can be compared to a model excited with the same input. The output of the model can be fitted to the measured output of the experiment to estimate the identifiable parameters. Dynamic identification is more difficult than the static one and also more prone to errors due to, for example, measurement noise. In most cases it is even not possible to identify all parameters independently, since parameters normally appear in groups in the equations of motion. Dynamic parameter identification in humanoid robots is described in [1], dynamic parameter identification in general is described in [4, 10, 11, 12, 13, 22, 28, 34].

In section 3.2 static and dynamic identification experiments are explained shortly. In section 3.3 design of an optimal identification experiment for the humanoid robot TUlip is described and results of experimental identification are presented.

3.2 Parameter identification in general

3.2.1 Static identification

Static identification is a way of identifying the system parameters when the system does not move. Examples of static identification experiments are measuring the lengths of the links with a caliper or weighting the masses of the links with a weighting scale. These experiments are simple and can be executed fast and accurately. Generally a system consists of more than one link, so the easiest way to identify the parameters of each link is to measure them before all links are interconnected. In that case, lengths, masses and the center of mass position for each link can be identified easily and accurately. However, when the system is already mounted together more advanced measuring techniques are needed to identify the mass and center of mass of separate links in the system. According to [5], for a humanoid robot it is possible to measure the mass and the center of mass position of each link separately while they are interconnected. In a leg for example, one can measure the gravitational moment at different locations along the whole leg and parts of the leg when some links are fixed to the ground. In this way the parameters can be determined for the whole leg and each part and by means of substraction every link can be identified individually.

Overall, static identification can generally identify all geometrical parameters of a system and some inertial parameters i.e. the masses of each link. However it is in general very hard and sometimes even impossible to identify all inertial parameters with a static identification experiment. It is preferable to identify as many parameters as possible in a static experiment because this makes the dynamic identification easier, faster and more accurate.

3.2.2 Dynamic identification

The purpose of dynamic identification is to estimate inertial parameters of the robot dynamical model. The parameters are identified dynamically, so with the system in motion. To facilitate parameter estimation, the model should be rewritten in a more convenient form, the so called regressor form. In this form, the unknown model parameters are extracted from the differential equations of motions and stored in a vector $\underline{\vartheta}_0$. This vector is a set of parameters, not the minimal one, and its elements can be either single inertial parameters or combinations of these parameters. The combination of parameters are consequence of the fact that not all inertial parameter appear linearly in the equations of motion. The regressor form of the model containing n_p parameters and n_e equations of motions is given by:

$$\underline{\zeta} = \underline{R}_0 \left(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}}\right) \underline{\vartheta}_0 \tag{3.1}$$

Where $\underline{\vartheta}_0 \in \mathbb{R}^{n_p}$ is the vector of parameters, $\underline{\zeta} \in \mathbb{R}^{n_e}$ is a vector of inputs to the system and matrix $\underline{R}_0 \in \mathbb{R}^{n_p \times n_e}$ is called the regressor [4, 10, 28, 34]. The parameter set $\underline{\vartheta}_0$ can be transformed in a base parameters set $\underline{\vartheta}$. The base parameter set is the minimal set of parameters that still describe the system completely [11, 12]. This set should be as small as possible to solve (3.1). The set is as small as possible if all columns in matrix $\underline{R}_0$ are linearly independent. To find the set of base parameters for which this holds an algorithm described in [17] can be used. This algorithm works in the following way. The regressor matrix $\underline{R}_0$ contains expressions of known constant parameters and joint variables that vary in time. The joint variables mostly appear in the expressions inside trigonometric functions. A function of

joint variables that appears in several expressions of the regressor or in only one, is called a fundamental function. The fundamental functions are stored in a vector $\underline{f} \in \mathbb{R}^{n_f}$, where n_f is the number of fundamental functions in $\underline{R}_0$, whereas the known parameters are stored in a vector $\underline{b}_{ij} \in \mathbb{R}^{n_f}$ such that the following holds:

$$\underline{R}_{0,ij} = \underline{f}^T \underline{b}_{ij} \tag{3.2}$$

where i and j represent a row and column index in regressor matrix $\underline{R}_0$, so for each element $R_{0,ij}$ in $\underline{R}_0$ there is a different vector $\underline{b}_{ij}$. All these vectors can be stored in one matrix $\underline{B} \in \mathbb{R}^{n_{fe} \times n_p}$, where $n_{fe} = n_f n_e$, in the following way:

$$\underline{B} = \begin{bmatrix} \underline{b}_{11} & \cdots & \underline{b}_{1n_p} \\ \vdots & \ddots & \vdots \\ \underline{b}_{n_{fe}1} & \cdots & \underline{b}_{n_{fe}n_p} \end{bmatrix} \tag{3.3}$$

This matrix $\underline{B}$ can be transformed to the Echelon form $\underline{B}_E$ by performing a Gauss-Jordan elimination. The result is an upper triangular matrix with the following form:

$$\underline{B}_E = \begin{bmatrix} \underline{B}_{Eu} \\ \underline{0} \end{bmatrix} \tag{3.4}$$

This matrix consists of three different types of columns. The first type is a column in which all elements are zero. This means that the corresponding column in regressor $\underline{R}_0$ is not needed for parameter identification and can be removed. The second type is a column in which all elements are zero, except one element is 1. This means that the corresponding column in regressor $\underline{R}_0$ is necessary and not linear dependent on others. The last type is a column with zeros and known parameter elements. This means that the corresponding column in regressor $\underline{R}_0$ is necessary, but linear dependent on others. This column can be removed. The corresponding base parameter vector $\underline{\vartheta}$ can be derived in the following way:

$$\underline{\vartheta} = \underline{B}_{Eu} \underline{\vartheta}_0 \tag{3.5}$$

This results in the complete minimal regressor form with base parameter set:

$$\underline{\zeta} = \underline{R}\left(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}}\right) \underline{\vartheta} \tag{3.6}$$

The length of the equations of motion depends primarily on the complexity of the system. The complexer the system is the more expressions are needed to describe the highly non linear dynamics. One can imagine that it could be very hard and time consuming to manually derive the regressor from the equations of motion as in (3.1). That is why we developed an automatic algorithm that can separate the different expressions in the equations of motion and put them in the proper matrix. The algorithm utilizes the fact that one can convert the equations of motion to text based strings. Within these strings it is easy to search for operators like $+$, $-$, $*$ and $/$, that divide the equation in different (mostly trigonometric) expressions. Each expression belongs to a certain group: an unknown parameter, known parameter, fundamental function or input. In this way we can work ourselves through all equations and put every expression in the proper matrix. Consequently we can build up the regressor matrix in an automated way. To the best of our knowledge this is a pioneering attempt to provide an automatic procedure to build the regressor form from a set of equations

of motion. The algorithm turns out to be fast and can be used on very complex systems. It already gave a positive value to another identification project on a Philips robotic arm [32]. The Matlab code of this algorithm can be found in Appendix B.

In most cases the system contains more base parameters than equations of motion, which means that not all base parameters can uniquely be determined from (3.1). That is why we estimate parameters along trajectories of several data points. At each data point, input and output are retrieved. For all data points n_t we create regressors from the system model of the form:

$$\underline{y} = \underline{H}\underline{\vartheta} \tag{3.7}$$

where $\underline{y} = \begin{bmatrix} \underline{\zeta}(t_1) \\ \vdots \\ \underline{\zeta}(t_{n_t}) \end{bmatrix}$ is the vector of input data,

and $\underline{H} = \begin{bmatrix} \underline{R}\left(\underline{q}(t_1), \underline{\dot{q}}(t_1), \underline{\ddot{q}}(t_1)\right) \\ \vdots \\ \underline{R}\left(\underline{q}(t_{n_t}), \underline{\dot{q}}(t_{n_t}), \underline{\ddot{q}}(t_{n_t})\right) \end{bmatrix}$ is the output data matrix.

In this way typically more equations are obtained with respect to the number of base parameters. The parameters should be estimated by fitting the model to the experimental data by means of a least squares or maximum likelihood algorithm. [4, 22, 28].

Finally we remark that from (3.7) we see that $\underline{H}$ is dependent on the robot trajectory, so in order to accurately retrieve all base parameters it is crucial to design the trajectory that excites all dynamics included in the system. This class of trajectories is called persistently exciting trajectories [4, 13, 28, 31], and will be derived in the next section. An example of how to derive the regressor form from the equations of motion can be found in Appendix A.2.

3.2.3 Persistently exciting trajectories

To identify the parameters of a system, persistently exciting trajectories are needed. These trajectories move the links in such a way that all dynamics in the system are excited. Moreover, the trajectories also need to lie within a range of feasible joint angles, because the motions on the robot are constrained to certain boundaries. One way to find a persistently exciting trajectory is to minimize the condition number of the regressor matrix $\kappa(\underline{H})$, while taking into account the constraints on the trajectories.

$$\min_{\underline{q}_t} \kappa\left(\underline{H}\left(\underline{q}_t, \underline{\dot{q}}_t, \underline{\ddot{q}}_t\right)\right) \tag{3.8}$$

$$\underline{c}_{lb,p} \leq \underline{q}_t \leq \underline{c}_{ub,p}$$
$$\underline{c}_{lb,v} \leq \underline{\dot{q}}_t \leq \underline{c}_{ub,v}$$
$$\underline{c}_{lb,a} \leq \underline{\ddot{q}}_t \leq \underline{c}_{ub,a}$$

where, $\underline{c}_{lb,p}$, $\underline{c}_{lb,v}$ and $\underline{c}_{lb,a}$ are the lower boundary constraints and $\underline{c}_{ub,p}$, $\underline{c}_{ub,v}$ and $\underline{c}_{ub,a}$ are the upper boundary constraints on position, velocity and acceleration respectively. As one can see, this is a constrained non linear optimization problem and there exist several optimization algorithms to solve this problem in a local fashion. Solutions usually depend on the initial

conditions, since the considered problem is not convex. A brute force method to find a persistently exciting trajectory is to choose an array of arbitrary joint angles as initial guess, and optimize this array to the lowest condition number. Because this method can be time consuming, it is also possible to postulate the trajectory as a finite Fourier series, so that the number of arguments to optimize decreases [31]. The optimization problem then changes to

$$\min_{\underline{a}_i, \underline{\omega}_i, \underline{\phi}_i, \underline{b}_i} \kappa\left(\underline{H}\left(\underline{q}_t, \dot{\underline{q}}_t, \ddot{\underline{q}}_t\right)\right) \tag{3.9}$$

$$\begin{aligned}
\underline{c}_{lb,p} &\leq \underline{q}_t \leq \underline{c}_{ub,p} \\
\underline{c}_{lb,v} &\leq \dot{\underline{q}}_t \leq \underline{c}_{ub,v} \\
\underline{c}_{lb,a} &\leq \ddot{\underline{q}}_t \leq \underline{c}_{ub,a}
\end{aligned}$$

where $\underline{H}$ is the regressor matrix and $\underline{a}_i$, $\underline{\omega}_i$, $\underline{\phi}_i$ and $\underline{b}_i$, $i = 1 \cdots n$, are coefficients of the finite Fourier series where n is the order of the serie:

$$\begin{aligned}
\underline{q}_t &= \sum_{i=1}^{n} \underline{a}_i \sin\left(\underline{\omega}_i t + \underline{\phi}_i\right) + \underline{b}_i \\
\dot{\underline{q}}_t &= \sum_{i=1}^{n} \underline{a}_i \underline{\omega}_i \cos\left(\underline{\omega}_i t + \underline{\phi}_i\right) \\
\ddot{\underline{q}}_t &= \sum_{i=1}^{n} -\underline{a}_i \underline{\omega}_i^2 \sin\left(\underline{\omega}_i t + \underline{\phi}_i\right)
\end{aligned} \tag{3.10}$$

Another often used approach is to postulate the trajectory based on a polynomial. Hereby, the number of optimization parameters also decreases, which fastens the optimization process. The optimization problem becomes:

$$\min_{\underline{a}_i} \kappa\left(\underline{H}\left(\underline{q}_t, \dot{\underline{q}}_t, \ddot{\underline{q}}_t\right)\right) \tag{3.11}$$

$$\begin{aligned}
\underline{c}_{lb,p} &\leq \underline{q}_t \leq \underline{c}_{ub,p} \\
\underline{c}_{lb,v} &\leq \dot{\underline{q}}_t \leq \underline{c}_{ub,v} \\
\underline{c}_{lb,a} &\leq \ddot{\underline{q}}_t \leq \underline{c}_{ub,a}
\end{aligned}$$

where $\underline{a}_i$, $i = 0 \cdots n$, are the coefficients of the polynomials of order n:

$$\begin{aligned}
\underline{q}_t &= \underline{a}_n t^n + \underline{a}_{n-1} t^{n-1} + \cdots + \underline{a}_1 t + \underline{a}_0 \\
\dot{\underline{q}}_t &= n\underline{a}_n t^{n-1} + (n-1)\,\underline{a}_{n-1} t^{n-2} + \cdots + 2\underline{a}_2 t + \underline{a}_1 \\
\ddot{\underline{q}}_t &= \left(n^2 - n\right)\underline{a}_n t^{n-2} + \left((n-1)^2 - (n-1)\right)\underline{a}_{n-1} t^{n-3} + \cdots + 6\underline{a}_3 t + 2\underline{a}_2
\end{aligned} \tag{3.12}$$

It turns out that this approach yields suitable trajectories in an efficient way, partly because it is fairly easy to implement.

Table 3.1: Geometric parameters

Length $[m]$				Center of mass position $[m]$					
l_{6y}	0.145	l_{6z}	0.34	c_{6x}	0.099	c_{6y}	0.098	c_{6z}	0.225
l_7	0.12			c_{7x}	0.054	c_{7y}	0.000	c_{7z}	0.007
l_{8y}	0.065	l_{8z}	0.035	c_{8x}	0.054	c_{8y}	0.000	c_{8z}	0.007
l_9	0.275			c_{9x}	0.054	c_{9y}	0.030	c_{9z}	0.082
l_{10}	0.275			c_{10x}	0.000	c_{10y}	0.021	c_{10z}	0.131
l_{11}	0.01			c_{11x}	0.000	c_{11y}	0.000	c_{11z}	0.050
l_{12}	0.15			c_{12x}	0.087	c_{12y}	0.068	c_{12z}	0.005
l_{13y}	0.145	l_{13z}	0.34	c_{13x}	0.099	c_{13y}	0.098	c_{13z}	0.225
l_{14}	0.12			c_{14x}	0.054	c_{14y}	0.000	c_{14z}	0.007
l_{15y}	0.065	l_{15z}	0.035	c_{15x}	0.054	c_{15y}	0.000	c_{15z}	0.007
l_{16}	0.275			c_{16x}	0.054	c_{16y}	0.030	c_{16z}	0.082
l_{17}	0.275			c_{17x}	0.000	c_{17y}	0.021	c_{17z}	0.131
l_{18}	0.01			c_{18x}	0.000	c_{18y}	0.000	c_{18z}	0.050
l_{19}	0.15			c_{19x}	0.087	c_{19y}	0.068	c_{19z}	0.005

Table 3.2: Foot contact geometric parameters

Length $[m]$			
l_{12t}	0.15	l_{12h}	0.06
w_{12i}	0.065	w_{12o}	0.085
l_{19t}	0.15	l_{19h}	0.06
w_{19i}	0.065	w_{19o}	0.085

3.3 Parameter identification on TUlip

The parameters that describe the geometric and inertial properties of the TUlip model are listed in table 3.1, 3.2 and 3.3 respectively. These parameters are determined using static identification experiments on the robot TUlip at the Delft University of Technology [14]. As can be seen, it is possible to identify all parameters separately using a static identification experiment. However, to execute these experiments, the robot needs to be demounted, which is not always possible. That is why we have developed a dynamic identification method, so that the experiments can easily be repeated when something changes on the robot.

From [1] it turns out that it is possible to identify all parameters of a humanoid robot, even the parameters of the non actuated torso of the robot. The full model including constraints and impacts should be evaluated. Therefore, the Cartesian position and orientation of the robot should be measured with an inertial sensor, gyroscope or vision based motion capture system as well as the contact forces with the floor with force sensors on the feet. Furthermore, the robot should be capable of performing a stable walking gait, which is persistently exciting the robot dynamics according to [1]. At this moment, TUlip has no inertial sensor or gyroscope. Feet sensors do not work properly and a vision based motion capture system is not available. For the rest, TUlip can not yet walk in a stable manner, so we are not able to identify all robot parameters at once using a dynamical identification experiment. That is why we only identify the parameters of both legs of TUlip by hanging the robot on a stand, fixing the torso

Table 3.3: Inertial parameters

Mass $[kg]$		Moment of inertia $[kg \cdot m^2]$					
m_6	8.594	I_{6x}	0.27217	I_{6y}	0.20518	I_{6z}	0.07047
m_7	0.614	I_{7x}	0.00002	I_{7y}	0.00095	I_{7z}	0.00056
m_8	0.614	I_{8x}	0.00002	I_{8y}	0.00095	I_{8z}	0.00056
m_9	2.141	I_{9x}	0.01467	I_{9y}	0.01134	I_{9z}	0.00148
m_{10}	0.315	I_{10x}	0.00396	I_{10y}	0.00368	I_{10z}	0.00009
m_{11}	0.001	I_{11x}	0.00001	I_{11y}	0.00001	I_{11z}	0.00001
m_{12}	0.366	I_{12x}	0.00040	I_{12y}	0.00157	I_{12z}	0.00205
m_{13}	8.594	I_{13x}	0.27217	I_{13y}	0.20518	I_{13z}	0.07047
m_{14}	0.614	I_{14x}	0.00002	I_{14y}	0.00095	I_{14z}	0.00056
m_{15}	0.614	I_{15x}	0.00002	I_{15y}	0.00095	I_{15z}	0.00056
m_{16}	2.141	I_{16x}	0.01467	I_{16y}	0.01134	I_{16z}	0.00148
m_{17}	0.315	I_{17x}	0.00396	I_{17y}	0.00368	I_{17z}	0.00009
m_{18}	0.001	I_{18x}	0.00001	I_{18y}	0.00001	I_{18z}	0.00001
m_{19}	0.366	I_{19x}	0.00040	I_{19y}	0.00157	I_{19z}	0.00205

and performing dynamical identification experiments on each leg separately. The parameters of the torso can be found with a static identification experiment.

3.3.1 Regressor form for stiff drive trains

To execute a dynamic identification experiment, the model should be written in the regressor form. In this form the parameters will certainly appear in combinations and not individually as stated in table 3.1 and 3.3. It turns out that it is very hard and maybe even impossible to extract single parameters out of these combinations due to the complexity and high amount of different terms in the equations. That is why certain terms need to be neglected in the model using the following reasoning. The humanoid robot TUlip has a Maxon motor in each joint. These motors are not directly coupled to the corresponding links, but gearboxes are installed in between. In this way a higher torque can be applied to the link with the same motor, but some extra friction is introduced. Nevertheless, if the gearbox ratio is high enough, this also ensures that the robot dynamics gets decoupled, because the control of one actuator will hardly be influenced by motions at other joints. Assuming this is the case, the original equations of the model can be simplified. The actuators can be added to the robot dynamical model in the following way. Firstly, we recover the description of the model from the previous chapter:

$$\underline{\tau}_L = \underline{D}\ddot{\underline{q}} + \underline{C}\dot{\underline{q}} + \underline{G} \tag{3.13}$$

where $\underline{\tau}_L$ is the torque applied to the link, $\underline{D} \in \mathbb{R}^{n \times n}$ is the inertia matrix, $\underline{C} \in \mathbb{R}^{n \times n}$ is called the Coriolis and centrifugal matrix and $\underline{G} \in \mathbb{R}^{n \times 1}$ is the gravity vector, n is the number of links in the system. An actuator model for a stiff drive train is shown in figure 3.1. In this figure, M represents the actuator, N represents the gearbox and L is the load. An expression for the actuator model of link i can be derived as follows:

$$\tau_{M,i} = I_{M,i}\ddot{\theta}_{M,i} + B_{M,i}\dot{\theta}_{M,i} + F_{c,i}sgn\left(\dot{\theta}_{M,i}\right) + \frac{\tau_{L,i}}{N_i} \tag{3.14}$$

where $\tau_{M,i}$ is the torque applied by the actuator, $\theta_{M,i}$ is the rotation of the motor shaft, $I_{M,i}$ is the inertia of the motor shaft, $B_{M,i}$ and $F_{c,i}$ are the friction parameters, $\tau_{L,i}$ is the torque

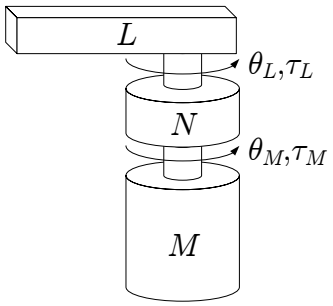


Figure 3.1: Actuator model with stiff drive train

from the link, and N_i is the gearbox ratio. This gearbox ratio couples the rotation of the motor shaft $\theta_{M,i}$ with the joint rotation $\theta_{L,i}$ in the following way:

$$\begin{aligned} \dot{\theta}_{M,i} &= N_i \dot{\theta}_{L,i} \\ \ddot{\theta}_{M,i} &= N_i \ddot{\theta}_{L,i} \end{aligned} \tag{3.15}$$

Now we can combine (3.13), (3.14) and (3.15) to write the equation of motion with the actuator implemented as follows:

$$\tau_{M,i} = I_{M,i} N_i \ddot{\theta}_{L,i} + B_{M,i} N_i \dot{\theta}_{L,i} + F_{c,i} sgn\left(\dot{\theta}_{L,i}\right) + \frac{1}{N_i} \sum_{j=1}^{n} D_{ij} \ddot{\theta}_{L,j} + \frac{1}{N_i} \sum_{j=1}^{n} C_{ij} \dot{\theta}_{L,j} + \frac{1}{N_i} G_i \tag{3.16}$$

where D_{ij}, C_{ij} and G_i are element (i, j) out of matrices $\underline{D}$, $\underline{C}$ and $\underline{G}$ respectively. From (3.16) we can conclude that some terms can be neglected. For example, the Coriolis and centrifugal term $\frac{C_{ij}}{N_i}$ depends on $\underline{q}$ and $\underline{\dot{q}}$, whereas the gravitation term $\frac{G_{ij}}{N_i}$ only depends on $\underline{q}$ and the gravity acceleration g. So, for low angular velocities $\underline{\dot{q}}$ the Coriolis and centrifugal term can be neglected. We decided to keep the gravity terms, since these will contribute to the robot dynamics at least g times more than the Coriolis and centrifugal terms. The final model for each link, from which a regressor form can be derived, is as following:

$$\tau_{M,i} = N_i \left(I_{M,i} \ddot{\theta}_{L,i} + B_{M,i} \dot{\theta}_{L,i} \right) + F_{c,i} sgn\left(\dot{\theta}_{L,i}\right) + \frac{1}{N_i} \left(\sum_{j=1}^{n} D_{ij} \ddot{\theta}_{L,j} + G_i \right) \tag{3.17}$$

From (3.17), we can derive the regressor form as in section 3.2.2. As can be seen, by adding the actuators to the model new parameters N_i, $I_{m,i}$, $B_{m,i}$ and $F_{c,i}$ are introduced. This is not a problem, because some of these parameters are known from the actuator supplier, whereas others should be estimated. With (3.17) we are able to derive the regressor form.

3.3.2 Regressor form for flexible drive trains

The actuator model presented in section 3.3.1 only works if the drive train between the motor and the load is stiff. However, TUlip has flexibilities in some of its drive trains. Therefore, we can not use the actuator model with stiff drive trains. The flexibilities in some of TUlip's drive trains are introduced by series elastic actuators. These actuators decouple the motor dynamics from the load dynamics. Discontinuities in the velocity, for example caused by

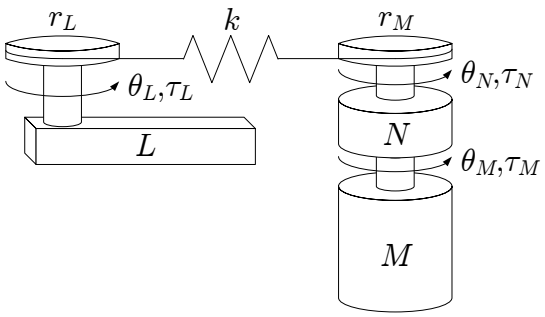


Figure 3.2: Actuator model with flexible drive train

impacts of the feet on the ground, are not transferred to the motor in this way, preventing big positioning errors on the motor side. On the other hand, due to the series elastic actuation an accurate positioning of the load position is hard to achieve, since the controller bandwidth is limited to the stiffness of the drive train. A model of the series elastic actuation present in TUlip is shown in figure 3.2. Herein, r_L and r_M are radii of the pulleys on the motor and load side respectively and k is the spring stiffness. All parameters in this figure are known, except for the spring stiffness. This parameter can be dynamically identified in the following way. Firstly, an expression for the flexible drive train of link i needs to be derived. The elongation of the spring Δl_i is:

$$\Delta l_i = r_{M,i}\theta_{N,i} - r_{L,i}\theta_{L,i} \tag{3.18}$$

where $\theta_{N,i} = \frac{\theta_{M,i}}{N_i}$. Due to this elongation the spring force $F_{s,i}$ is:

$$F_{s,i} = k_i \Delta l_i = k_i \left(r_{M,i}\theta_{N,i} - r_{L,i}\theta_{L,i}\right) \tag{3.19}$$

The spring force acts on the pulley on the motor side causing a torque in the gearbox of:

$$\tau_{N,i} = F_{s,i} r_{M,i} = k_i r_{M,i} \left(r_{M,i}\theta_{N,i} - r_{L,i}\theta_{L,i}\right) \tag{3.20}$$

Finally this torque is transferred to the motor which results in an expression of the motor torque as a function of the spring stiffness, the pulley radii, the gearbox ratio and the motor and load angles:

$$\tau_{M,i} = \frac{\tau_{N,i}}{N_i} = k_i \frac{r_{M,i}}{N_i} \left(r_{M,i}\frac{\theta_{M,i}}{N_i} - r_{L,i}\theta_{L,i}\right) \tag{3.21}$$

In (3.21) there is neither friction nor motor inertia included, but we can add these easily:

$$\tau_{M,i} = I_{M,i}\ddot{\theta}_{M,i} + B_{M,i}\dot{\theta}_{M,i} + F_{c,i} sgn\left(\dot{\theta}_{M,i}\right) + k_i \frac{r_{M,i}}{N_i} \left(r_{M,i}\frac{\theta_{M,i}}{N_i} - r_{L,i}\theta_{L,i}\right) \tag{3.22}$$

With (3.22) a simple regressor form can be derived to identify the friction terms and the spring stiffness of the drive train. The motor inertia and gearbox ratio are known from the actuator supplier, the pulley radii can be measured easily and the joint angles and motor torque are controlled during an experiment. So the regressor form can be derived as follows:

$$\tau_{M,i} - I_{M,i}\ddot{\theta}_{M,i} = \begin{bmatrix} \dot{\theta}_{M,i} & sgn\left(\dot{\theta}_{M,i}\right) & \frac{r_{M,i}}{N_i}\left(r_{M,i}\frac{\theta_{M,i}}{N_i} - r_{L,i}\theta_{L,i}\right) \end{bmatrix} \begin{bmatrix} B_{M,i} \\ F_{c,i} \\ k_i \end{bmatrix} \tag{3.23}$$

After identification, we can use the spring stiffness to compute the actual torque that was supplied to the load during the experiment in a similar way as before. Namely the force acting on the load side is the same as in (3.19), so the actual torque applied to the load is:

$$\tau_{L,i} = F_{s,i} r_{L,i} = k_i r_{L,i} \left(r_{M,i} \theta_{N,i} - r_{L,i} \theta_{L,i} \right) \tag{3.24}$$

Finally this torque can be used to derive the complete regressor form of the load dynamics using the following equation:

$$\tau_{L,i} = \sum_{j=1}^{n} D_{ij} \ddot{\theta}_{L,j} + G_i \tag{3.25}$$

From (3.25), we can derive the regressor form as in section 3.2.2. To conclude, we created a framework to identify all necessary parameters both in stiff and flexible joints. The expressions (3.17) and (3.25) apply to all links in the system and can be used to compute all load parameters, motor friction coefficients and the stiffness of the drive train.

3.3.3 Testing the algorithm on the model

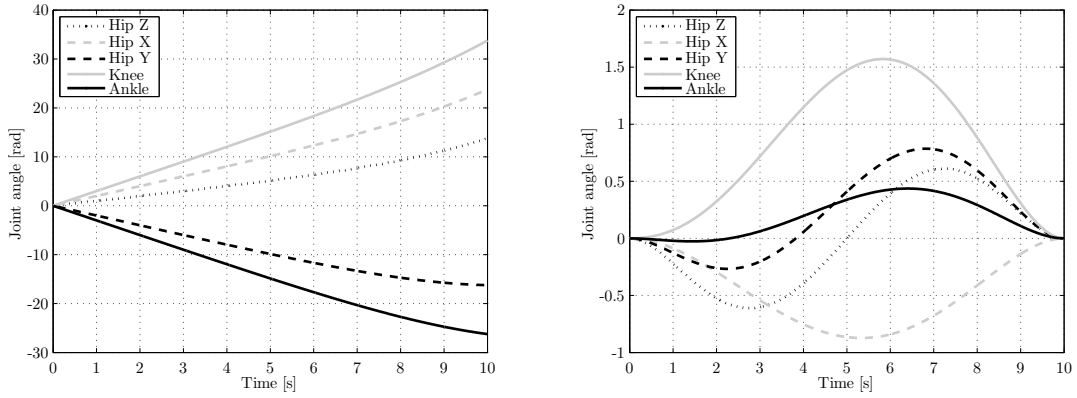
Before we start experiments on the real robot, we first investigate if the derived algorithm works in a simulation on the dynamic model of the robot. Firstly, for simplicity, we isolate one leg with five degrees of freedom from the dynamic model derived in the previous chapter. With this simplification we presume that the torso of the robot is fixed and that we can identify both legs separately. The resulting equations of motion can be rewritten in the regressor form of (3.6) using the algorithm presented in section 3.2.2. This procedure results in the regressor matrix: $\underline{R} \in \mathbb{R}^{5\times 14}$, the base parameter vector $\underline{\theta} \in \mathbb{R}^{14}$ and the input vector $\underline{\tau} \in \mathbb{R}^5$. This means there are five equations and fourteen base parameters describing the dynamic model. Moreover, each link has two additional unknown friction parameters, that can be added to the base parameter set as mentioned in section 3.3.1. Flexibilities in the drive trains are not taken into account in this model.

For this regressor form we can try to find persistently exciting trajectories. We optimize the condition number of the regressor matrix in accordance with (3.11) using Matlab. As initial guess the trajectories shown in figure 3.3(a) are used. As can be seen, these trajectories turn all links around several times, which means that they are infeasible on the robot TUlip. The optimization algorithms, however, is able to find feasible trajectories, shown in figure 3.3(b). The optimization routine minimizes the condition number of the regressor matrix from $3.2661 \cdot 10^8$ to 747.0162. The resulting trajectories are first tested in simulation and later on used in experiments to identify the parameters of TUlip.

In a Matlab simulation, simple PD controllers calculate the torques required to track the optimal trajectories. In figure 3.4 the actual joint angles from the simulation are depicted. As can be seen, the tracking of the trajectories is good.

The output data are put into the regressor matrix $\underline{H}$ and input vector $\underline{y}$ of (3.7). The resulting system of equations is solved using the least squares method. With increasing time the parameters are converging to certain values as can be seen in figures 3.5(a) and 3.5(b).

Finally we can compare the actual parameters with the parameters found by the identification



(a) Initial guess trajectories

(b) Final trajectories

Figure 3.3: Optimization to find persistently exciting trajectories

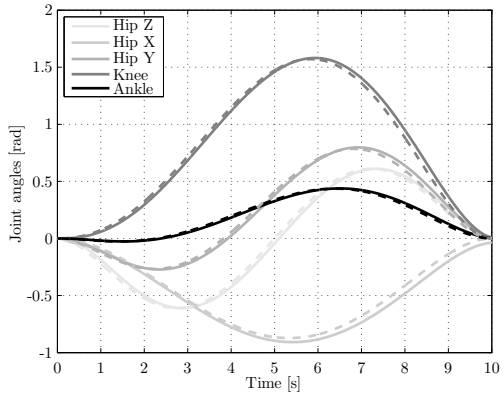
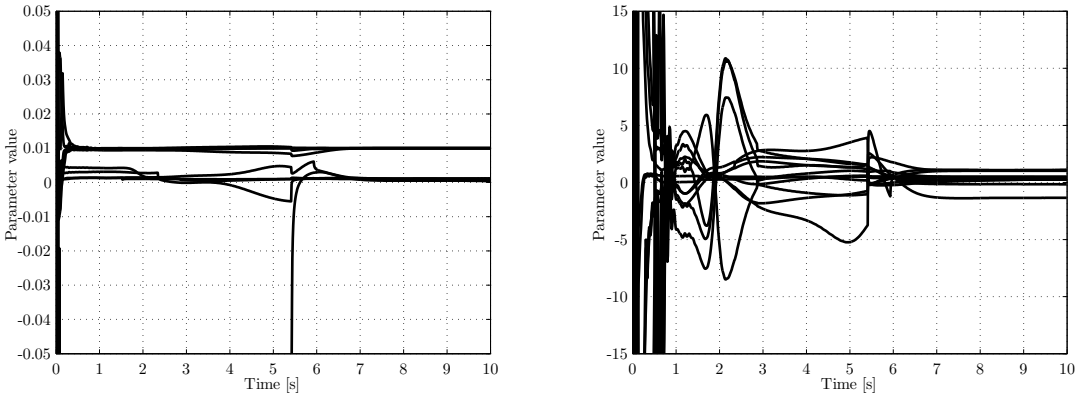


Figure 3.4: Tracking of optimized persistently exciting trajectories in simulation, solid: actual joint angles; dashed: reference trajectory



(a) Friction parameters

(b) Base parameters

Figure 3.5: Convergence of parameters in identification in simulation

Table 3.4: Actual and identified parameter values during simulation

(a) Friction parameters

Parameter	Actual	Identified
B_{m1}	0.0100	0.0100
F_{c1}	0.0001	0.0001
B_{m2}	0.0100	0.0100
F_{c2}	0.0001	0.0001
B_{m3}	0.0100	0.0100
F_{c3}	0.0001	0.0001
B_{m4}	0.0100	0.0100
F_{c4}	0.0001	0.0002
B_{m5}	0.0100	0.0100
F_{c5}	0.0001	0.0003

(b) Base parameters

Parameter	Actual	Identified
θ_1	1.9298	1.9559
θ_2	0.3445	0.3479
θ_3	0.0902	0.0843
θ_4	0.3727	0.3791
θ_5	0.0855	0.0666
θ_6	0.1252	0.1331
θ_7	0.0175	0.0406
θ_8	0.0702	0.0478
θ_9	0.0014	0.0141
θ_{10}	0.1000	0.1100
θ_{11}	0.6800	0.6429
θ_{12}	0.6800	0.7105
θ_{13}	0.1000	0.0908
θ_{14}	0.5000	0.4670

algorithm. In tables 3.4(a) and 3.4(b) one can see that the parameters are identified with high accuracy. This implies that the algorithm works and can be used on the real humanoid robot TUlip, which will be presented in the next section.

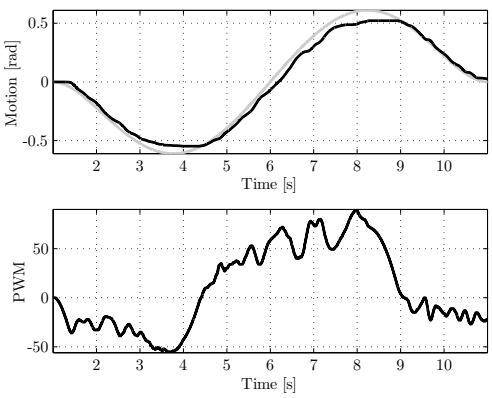
3.3.4 Experiments on TUlip

To identify the physical parameters of TUlip we performed a series of experiments. During these experiments, the movements of the robot are prescribed by persistently exciting trajectories computed in the previous section. The optimal trajectories are simultaneously realized by means of PD control to the robot and the trajectory tracking results are shown in figures 3.6(a), 3.6(b), 3.6(c), 3.6(d) and 3.6(e). The sampling rate is 1000 Hz.

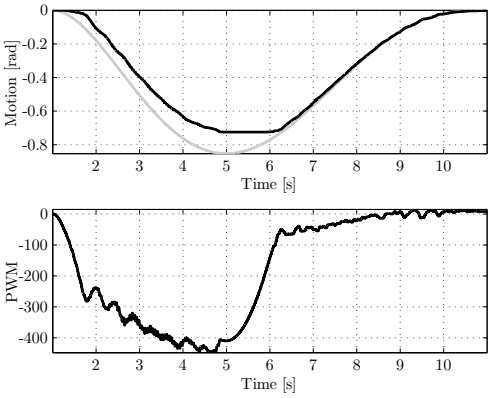
The tracking behavior of TUlip using ordinary PD controllers is very poor in some joints. Despite the fact that the used gains were as high as possible without destabilizing the system, the tracking behavior could not be improved more. This is mostly caused by the series elastic actuation, which is not adjusted to achieve a high position tracking performance using PD controllers alone. Moreover, the ankle actuation, which is the poorest of all, uses so called Bowden cables that contain high non linear friction. This friction can never be compensated using mere PD controllers.

With the experimental data we can estimate the parameters of TUlip using the framework presented in section 3.3.1 and 3.3.2. In TUlip, the hip Y (figure 3.6(c)), knee (figure 3.6(d)) and ankle (figure 3.6(e)) drive trains contain series elastic actuation. So for these three joints, we firstly have to identify the spring stiffness and motor friction properties using (3.23). The convergence of the estimated parameters is shown in figures 3.7(a) and 3.7(b).

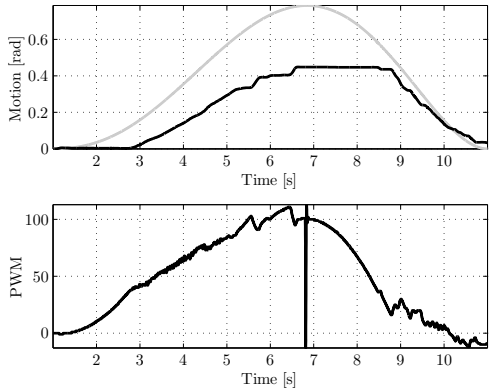
As can be seen from these figures, the estimation process converges to certain values over



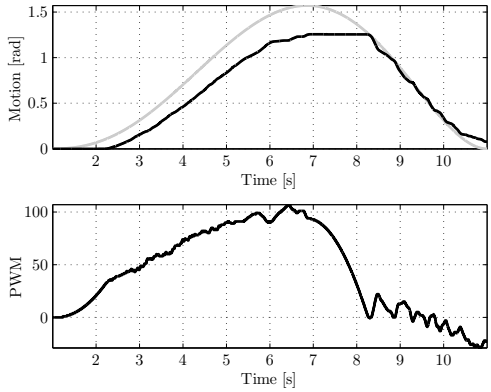
(a) Hip Z



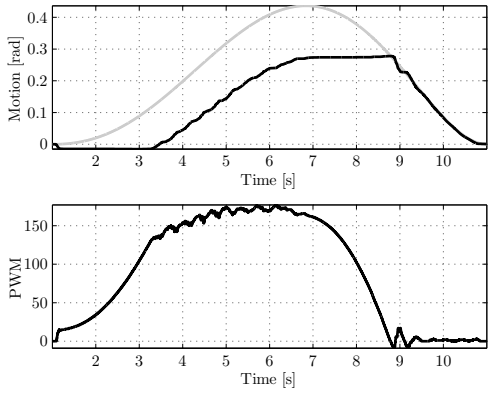
(b) Hip X



(c) Hip Y

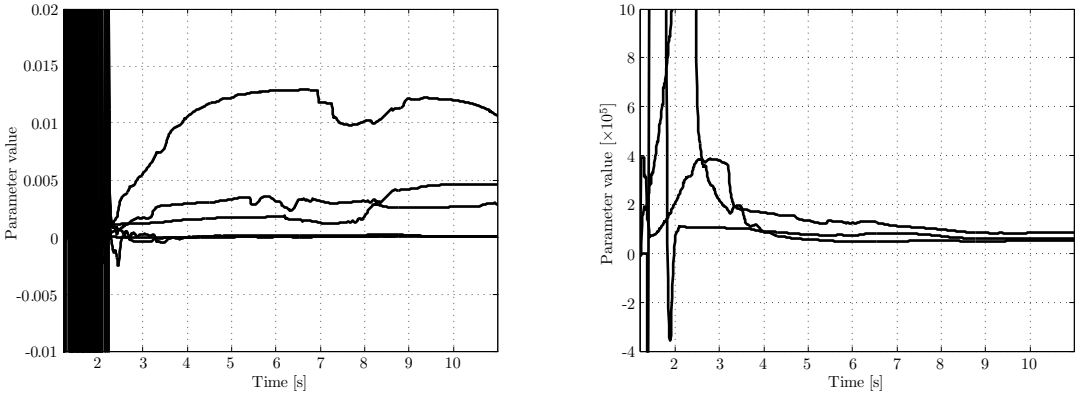


(d) Knee



(e) Ankle

Figure 3.6: Tracking behavior for each joint, upper figure: trajectory (grey) and joint angle (black); lower figure: input signal to the motor



(a) Friction parameters (b) Stiffness parameters

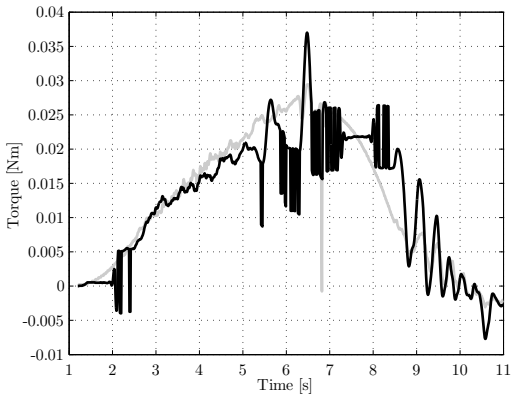
Figure 3.7: Convergence of parameters during flexible drive train identification

Table 3.5: Estimated parameters of TUlip

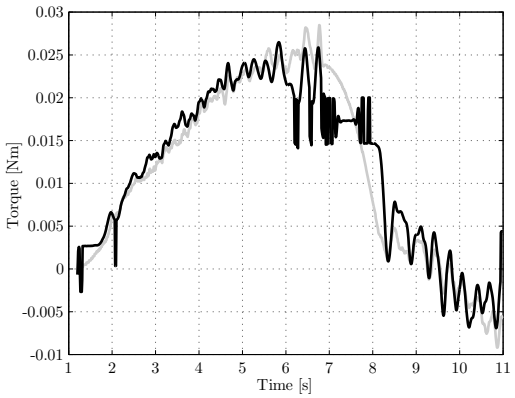
Parameter	Estimated value	Standard deviation
B_{m3}	0.00008	0.0000045
F_{c3}	0.00456	0.0000745
k_3	83008	131
B_{m4}	0.00008	0.0000021
F_{c4}	0.00268	0.0001168
k_4	59966	90
B_{m5}	0.00004	0.0000227
F_{c5}	0.01178	0.0004170
k_5	51394	801

time as more data are taken into account. This indicates that possible errors are averaged out and that the estimated parameters become more accurate. The estimated stiffness and motor friction parameters are stated in table 3.5 in SI units. These values are the averages of the final 2000 data points, with corresponding standard deviation. Especially the order of the stiffness values gives confidence, since we know that the stiffness of the full springs is 34 N/mm. The springs used on TUlip are shortened, which increases the stiffness. Despite the fact that estimations of the stiffness match expected values, we will further validate the identification process. Input torques are calculated from the model (3.21) with the identified parameters and compared to the measured torques from the experiment. These validations are shown in figures 3.8(a), 3.8(b) and 3.8(c).

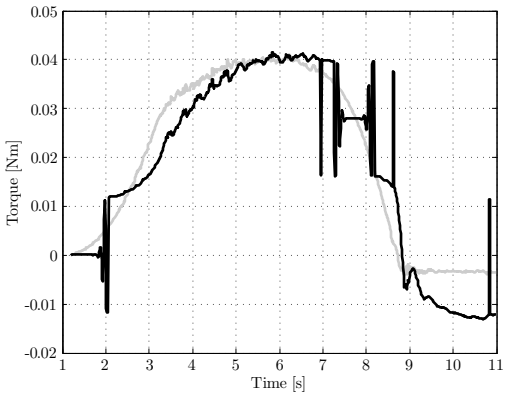
The overlap in these figures is high. The scattering behavior starting around 7 seconds is caused by the stick slip behavior of friction most probably, our friction model in (3.23) is not accurate enough. Replacing the sign function by a hyperbolic tangent did improve the results slightly. Nevertheless, from these results we might again conclude that the estimated parameters are of sufficient quality and useful for dynamic identification of parameters of the robot rigid dynamics. The estimated spring stiffnesses of table 3.5 are used to calculate the load torques according to (3.24). These torques are used in a regressor of (3.25) and combined



(a) Hip Y



(b) Knee



(c) Ankle

Figure 3.8: Validation of identified parameters for the flexible drive trains: measured torque (grey) and calculated torque (black)

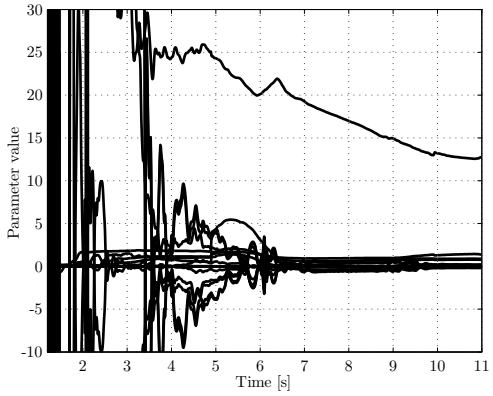


Figure 3.9: Convergence of the base parameter for the full identification process

with the regressor of the hip Z and the hip X joints of TUlip, that contain stiff drive trains (3.17). The convergence of this estimation is shown in figure 3.9.

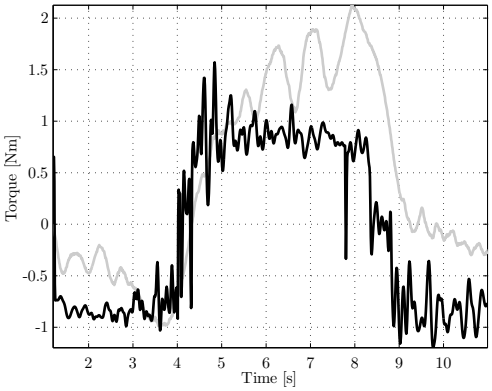
The convergence of the estimation process over time is rather good. This indicates that possible errors are averaged out and that the estimated parameters become more accurate over time. The estimated parameter combinations are again listed in table 3.6 for 2000 data points with the corresponding standard deviation.

From these values and their standard deviations it can be concluded that most parameters are identified with a confident accuracy. But especially the moments of inertia appear to be difficult to identify, which is caused by the fact that they only appear in the equations motion divided by the square of the gearbox ratio. This makes them very small in comparison with other parameters and therefore difficult to identify. However, this also implies that they are of minor importance in the equations of motion and it is not of high importance to identify them with high accuracy. The identified parameter values can be further validated by computing the torque with the estimated parameters and the measured joint angles using (3.17) and by comparing them with the measured torques. These validations are shown in figures 3.10(a), 3.10(b), 3.10(c), 3.10(d) and 3.10(e).

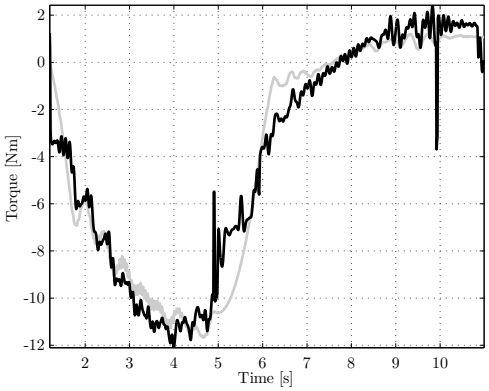
From these figures we can say that in most joints, the identification gives proper results. As expected, the ankle joint seems to give the worst results. Most likely this is caused by the fact that it is not modeled appropriate enough. Namely in this joint significant nonlinear friction is present, that is not accurately enough taken into account in the model.

3.4 Conclusion

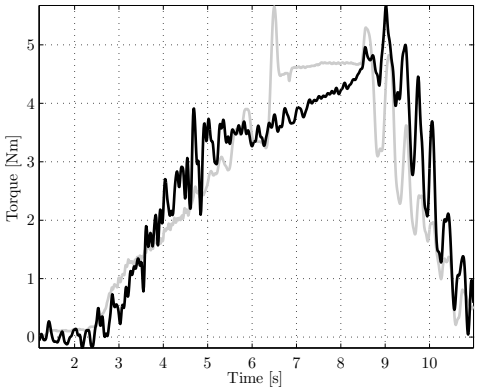
In this chapter we described how to identify all geometric and inertial parameters that describe the model. Static identification and dynamic identification are explained and used to identify the parameters on the humanoid robot TUlip. The model of TUlip has been written in the regressor form using an automatic algorithm. An optimization routine minimizes the regressor to find persistently exciting trajectories. These trajectories are tested in simulations



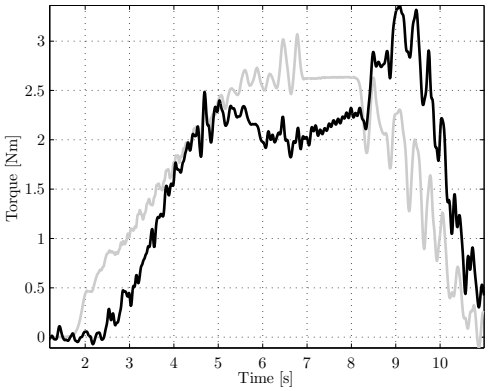
(a) Hip Z



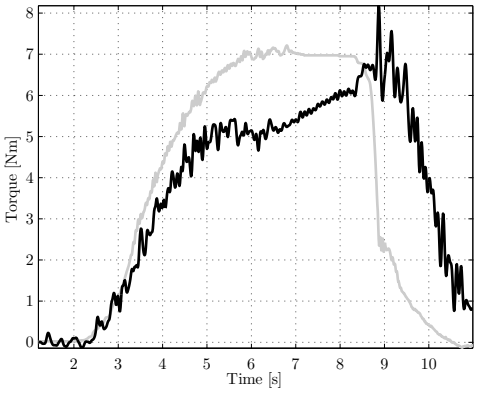
(b) Hip X



(c) Hip Y



(d) Knee



(e) Ankle

Figure 3.10: Validation of identified parameters for the full model: measured torque (grey) and calculated torque (black)

Table 3.6: Estimated parameters of TUlip

Parameter combination	Estimated value	Standard deviation
$c_{6x}^2 m_6 a_{11}^2 m_5 + a_{11}^2 m_4 + a_{11}^2 \frac{1}{d_9} c_{3z} m_3$	0.01942	0.15698
$c_{6x} m_6 + a_{11} m_5 + a_{11} m_4 + a_{11} \frac{1}{d_9} c_{3z} m_3$	0.90143	0.01490
$m_6 + m_5 + m_4 + \frac{1}{d_9} c_{3z} m_3$	1.31041	0.11251
$m_5 c_{5x}^2 + a_{10}^2 m_4 a_{10}^2 \frac{1}{d_9} c_{3z} m_3$	0.05399	0.16661
$m_5 c_{5x} + a_{10} m_4 + a_{10} \frac{1}{d_9} c_{3z} m_3$	0.76691	0.01017
$c_{4x}^2 m_4 + a_9^2 \frac{1}{d_9} c_{3z} m_3$	-0.25609	0.10250
$c_{4x} m_4 + \frac{1}{d_9} a_9 c_{3z} m_3$	-0.14063	0.02693
$c_{3z}^2 m_3 - d_9 c_{3z} m_3$	-0.04169	0.05321
$m_3 - \frac{1}{d_9} c_{3z} m_3$	13.17218	0.53670
I_{6y}	0.04489	0.16541
I_{5z}	0.09691	0.17372
I_{4z}	0.14063	0.10245
I_{2y}	0.01008	0.03265
I_{3y}	0.28832	0.14656
B_{m1}	0.00004	0.00002
F_{c1}	0.00861	0.00023
B_{m2}	0.00042	0.00004
F_{c2}	0.02167	0.00103

using the model of TUlip and then used in experiments on the real robot to identify unknown model parameters. Eventually, promising results were found so we can conclude that dynamic identification on a humanoid robot is possible. To further improve the identification results, another friction model should be used, since the friction model used in this work turned out not to be good enough.

Chapter 4

Stability of bipedal walking

4.1 Introduction

Stability of humanoid robots is hard to analyze since these systems are non-linear, under actuated, feature combined continuous and discrete modes and do not necessarily behave periodic. The common stability and robustness criteria, such as eigenvalues, gain and phase margin, Lyapunov stability, etc., can be applied on certain modes, such as moving one leg or keeping a certain torso orientation, but are often insufficient to characterize all modes together. The difficulties arise from the fact that a humanoid robot is a highly non linear under actuated dynamical system, which is subject to impacts, variable external forces and discrete changes in system dynamics. Therefore specific techniques have been developed to analyze stability of biped walking. Here, we mention the methods of zero moment point [33], foot rotation indicator [25], Poincaré return maps [15, 35] and the theory of capture points [25, 24]. We will explain these criteria shortly in the next sections, and we will point out if these criteria are necessary and sufficient to perform stable bipedal motion.
First we need to define what is a stable bipedal walking system. We consider bipedal walking stable if it is carried out without falling. We realize that this is a very general definition that may have many interpretations. At the same time, this definition is restrictive enough to include only those situations where the robot is allowed to touch the ground only with its feet. If any other point on the robot touches the ground, we characterize this as a fall. Consequently, sitting or crawling are out of scope of biped walking.

In this section we review several criteria for stability of bipedal walking. Two the most often used, namely ZMP and Poincaré, are presented in more detail and examined on an academic model of a planar biped with six actuated degrees of freedom.

4.2 Stability criteria for a humanoid robot

4.2.1 Zero moment point and foot rotation indicator

A well known and often used approach in the field of bipedal systems to design stable bipedal walking is the so called method of zero moment point. The zero moment point (ZMP) is the location on the ground where the net moment generated from the ground reaction forces is strictly perpendicular to the ground. In this point a net force acts that prevents the biped

from tipping around the foot as long as the ZMP lies inside the support polygon of the foot. When the ZMP lies on the edge of the support polygon, the robot might start tipping. Since the location of the ZMP depends on the ground reaction forces, the ZMP can never lie outside the foot support polygon. Therefore the foot rotation indicator (FRI), sometimes also called fictitious zero moment point (FZMP), is the point on the ground where the net ground reaction force would have to act to keep the foot stationary given the state of the biped and the accelerations of its joints. A force at this point also acts such that the net moment around the foot is zero, but this point can lie outside the foot support polygon and is therefore a more general form of the ZMP. It gives a positive as well as a negative margin with respect to the support polygon for control purposes. The ZMP and FRI points can be measured through the ground reaction forces with force sensors on the feet of the robot (in this case ZMP and FRI are often called center of pressure, CoP) or by analytically calculating it using the joint accelerations.

In most bipedal systems nowadays, the ZMP and FRI are used to generate stable walking gaits in terms of the reference. When these trajectories are accurately followed by joint controllers, the robots can perform stable gaits. By online adaptation of the trajectories it is possible to correct the robot for unforeseen disturbances like a push or rough terrain. Although this method works well, it is in general not sufficient nor necessary for the analysis of the complete range of motions of bipedal systems. The ZMP is not sufficient because when you switch off a standing robot, it will collapse, but the ZMP remains in its support polygon. For motions like running the ZMP is also not necessary, because during some phases of running there is no contact with the ground, while according to the stability definition, the biped is still stable.

4.2.2 Poincaré return maps

A Poincaré return map can be used to investigate periodic motions of a dynamical system. A stable dynamical system exhibiting a periodic motion will return to approximately the same state in its state space after each period, it will perform a limit cycle. On smooth systems, for small deviations, this behavior might follow a linear relation [15]:

$$\Delta \underline{q}_{n+1} = \underline{K} \Delta \underline{q}_n \tag{4.1}$$

with $\Delta \underline{q}_n$ the deviation from the limit cycle on cycle n and $\underline{K}$ is called the linear return matrix. The eigenvalues of this matrix will determine if the system performs a stable periodic orbit. If the eigenvalues lie within the unit circle, the deviation will eventually disappear and the system will converge to a stable limit cycle. Eventually measuring the eigenvalues of the linear return matrix is a good criteria to characterize the stability of periodic motions. Walking is a periodic motion, and this method is therefore commonly used in passive dynamic walkers [9, 36]. However, we have to be very careful in applying the Poincaré return map method on humanoid robots, because walking is in fact a multimode dynamical behavior. A humanoid robot interacts with the ground when it performs a gait. Due to impacts and ground contact constraints, the dynamics of the system can change suddenly, which means that the continuous dynamics is interrupted by discrete events. The Poincaré return map method is only intended to analyze smooth systems [35, 19].

Another drawback of this method is that it can only be used for periodic motions with

small deviations from the limit cycle. In general the walking motions of bipedal systems are periodic, but there can occur situations when the motions are not periodic anymore. For example starting, stopping and turning are not periodic, and the stability can in general not be determined using a Poincaré return map. Therefore, this method is not sufficient nor necessary for analyzing the complete motions range of bipedal systems.

4.2.3 Angular momentum

A new approach to determine how humans are able to maintain their stability in a high number of different motions is investigating the angular momentum of a humans torso during these motions. It appears that humans are regulating the angular momentum of their center of mass during standing, walking and running. A common hypothesis is that the angular momentum during a motion should be minimized in order to keep stability. If there is a need for balance, one can use the preserved angular moment to balance, such as lunging the upper body or windmilling the arms when being pushed. Although this method is not yet investigated thoroughly, we can already state that it is not a sufficient nor a necessary criterion for stability. Not necessary because one could make all kinds of movements with the upper body while moving and not sufficient because a biped can still fall over while keeping its angular momentum at zero.

4.2.4 Capture point

The criterion of angular momentum is related to the method of capture points. This method stems from the fact that angular and linear momenta are coupled when a bipedal robot falls. Furthermore, linear momentum is directly coupled to the speed of the robot, so by controlling the speed one could control the angular momentum and stability in a human-like way. The way to control the speed of the robot is to use capture points. A capture point is a point on the ground where the bipeds center of pressure must lie in order to prevent the biped from falling. When a robot begins to fall, the capture point shifts over the ground. If the robot is able to put its center of pressure on the capture point, for example by making a step, the robot becomes stable again. It could be that in certain situations the capture point is not reachable in one step. The robot can then make two or more steps to stabilize itself. The basic idea of the capture points is that when the robot gets destabilized and falls, there is a trade off between the time of falling and the time needed to make a step. If the destabilization is small, the robot can recover without making a step, if it is bigger, the robot should make one or more steps to recover and eventually, if the destabilization is too big, the robot will fall. So far this criterion is very theoretical, but it seems to be necessary, sufficient and at the same time reasonably human-like. Also, it appears that a very simple model of an inverted pendulum can be applied to calculate the angular momentum of a more complex robot [24] and then this can be used for stability analysis.

4.3 Planar model

Since the complexity of gait design and stability analysis grows exponentially with the number of degrees of freedom of the biped, it has been chosen, in the scope of this project, to use a planar model for gait design and stability analysis. The model consists of a torso, two upper legs, two lower legs and two feet. The corresponding equations of motion are derived using

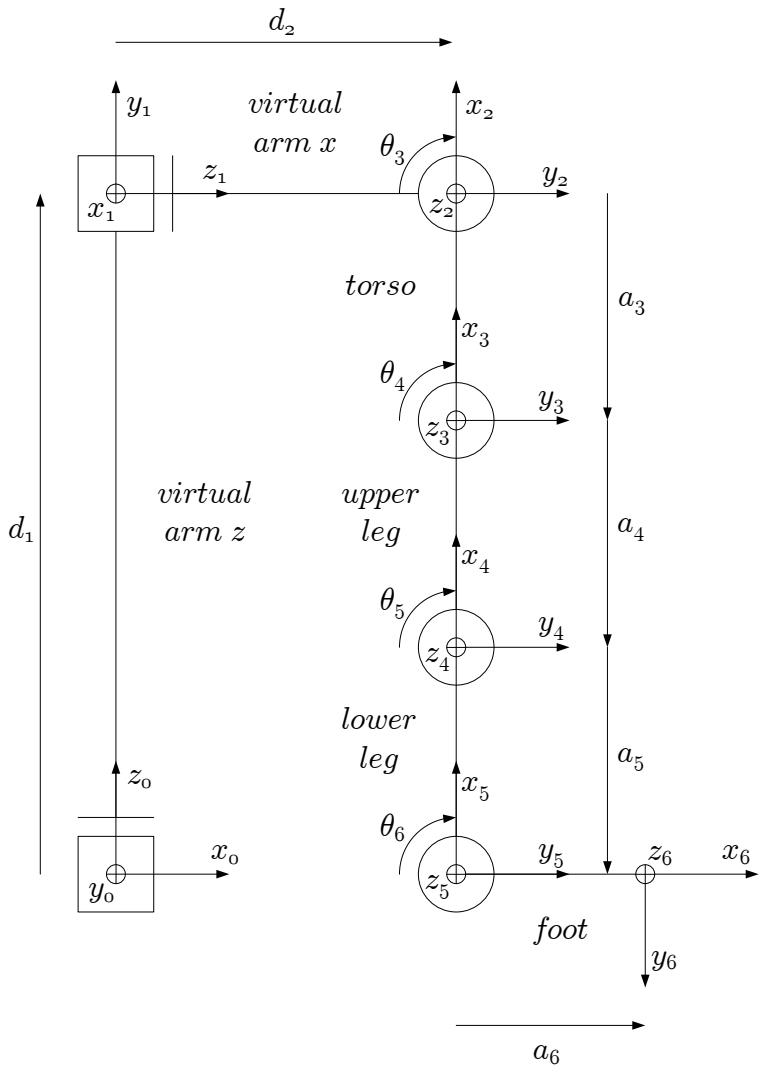


Figure 4.1: Denavit Hartenberg convention for the planar model

the methodology explained in section 2.2.4. Two contact points are modeled on both feet representing the toe and the heel. According to our stability condition only these points are allowed to make contact with the ground. Contact impulses with the ground are modeled being completely inelastic and during contact, it is assumed that ground friction is high enough to prevent the feet from slipping. The kinematic model of the right leg, according to the DH convention, is shown in figure 4.1. The left leg is identical and added to this model as described in 2.2.4.

4.4 Gait design and analysis

4.4.1 Introduction

Gait design is a an essential part in the process of letting a biped walk. The gait determines the style of walking as well as the step size and speed of the biped. The gait consists of

certain joint trajectories that prescribe the joint motions as a function of time. These joint trajectories can be predefined or online calculated during a walk. In the whole state space of the biped motion there are infinitely many combinations of joint trajectories, but we are only interested in the joint trajectories that make sure that the biped can walk without falling, since this is required by the definition of stability. If a set of trajectories satisfies the stability criterium, we can call them stable and characterize them as a stable gait. To find a particular set of joint trajectories that forms a stable gait, from all possible state trajectories of the biped, is a strenuous job. One can imagine that there are a lot of different ways a biped can stably walk. Unfortunately, even finding one particular motion that is stable step after step is difficult. Fortunately, there exist certain methods to determine stable gaits. These methods can basically be subdivided into two categories based on the type of gait they create.

The first and mostly used category consists of so called ZMP based gaits. These are gaits that consist of trajectories in which the biped is stable in every configuration they prescribe. One could stop the motion of the biped at any moment and the biped would still be standing stably on the ground. To achieve this, the momentum of the biped should be kept low and therefore the joint accelerations and velocities are constricted to serious bounds, constraining the speed of the walk. Bipeds that walk in this way exist in various sizes and probably the most famous one is Honda's Asimo. Methods that generate these gaits are mostly based on the zero moment point stability criterion. Although this criterion is not necessary as we saw before, it is in most but the extreme cases sufficient and certainly easy to use.

The second category consists of so called limit cycle walking gaits. Robots executing those gaits can not be stopped at any moment without falling. It is far more difficult to design limit cycle walking gaits, since the search space is less restricted. The biped needs to be at approximately the same state at the end of every step, what happens between two successive steps is not important as long as it doesn't violate the definition of stability. Leaving the restrictions gives a major advantage: the walking speeds can be much higher and by proper use of gravity one can show that energy consumption decreases significantly. Since this category is more challenging it is only of interest by researchers in recent years and there are so far no methods known in literature that can provide complete limit cycle walking gaits.

4.4.2 ZMP based gait design

According to [7] there are different methods to design a ZMP based gait. A class of methods called inverted pendulum mode methods, approximates the dynamics of the biped by an inverted pendulum. Similarly to the capture point stability criterion, the location for the foot placement can be calculated to counterbalance the tilting motion of the torso during a step. Since the equations of motion of an inverted pendulum are well known, it is possible to calculate a Cartesian trajectory for the center of mass position of the biped. One can solve the inverse kinematics of the robot to find joint trajectories that satisfy the Cartesian trajectory of the center of mass.

Another way of designing a ZMP based gait is to use walking primitives. Walking primitives are successive statically stable postures of the biped interconnected with smooth joint trajectories. Since it is quiet easy to find statically stable postures for the planar model, we will use this method to design a ZMP based gait. Statically stable postures can be found by

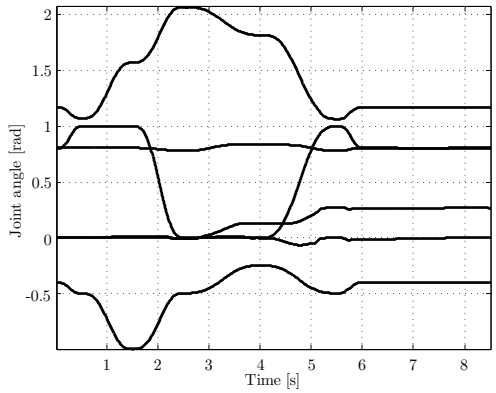


Figure 4.2: Joint angles as a function of time for a step with gait based on ZMP

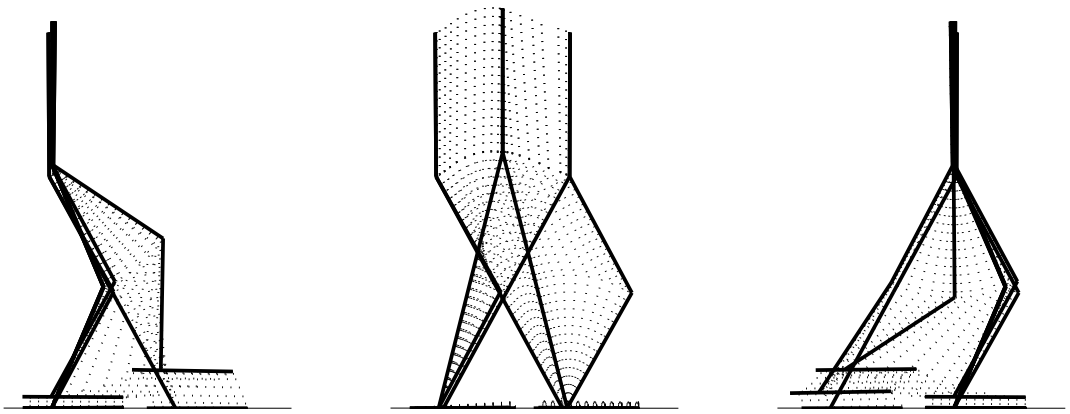
ensuring that the ZMP lies inside the support polygon of the feet. For the planar model, this is not difficult, because one can keep the torso of the robot always above the stance foot. The torso is the heaviest part of the robot and therefore this will guarantee that the ZMP stays inside the support polygon if the joint velocities between two successive postures are not too high. As stated, the stable postures define the joint positions at certain moments during the gait. These joint positions can be interconnected with splines, a cosine velocity profile [21] or a Bezier curve [16] to produce the complete gait.

In figure 4.2 the joint trajectories are shown including the stable postures and in figures 4.3(a), 4.3(b) and 4.3(c) snapshots of one step are illustrated. The start configuration of this step is the same as the end configuration so one can perform multiple steps to walk. The model is symmetric in the sagittal plane, so a step with the left leg is exactly the same as a step with the right leg.

4.4.3 Limit cycle walking gait design

As mentioned before there are no methods available in literature that can mathematically generate complete limit cycle walking gaits. Ongoing research and growing interest in this field show an increasing rate of progression. Despite the fact that so far no one found the holy grail that tells us how humans are capable of walking, a lot of interesting aspects of the human locomotion have been investigated [2, 15, 36]. These aspects will be mentioned later on and can certainly help us finding a limit cycle walking gait, but it is still very time consuming to find the limit cycle walking gait for complex bipeds [29], even if you are an experienced person in the field. But with the planar model, using the locomotion aspects of [2, 15] as guidelines, one may find a limit cycle walking gait in a reasonable time.

In [2, 15] some small, but rather important aspects of the human-like gait are investigated. These aspects include swing leg retraction, ankle actuation, upper body control, active lateral foot placement and knee flexion during push off. Taking these aspects into account we are able to design a gait for the planar model with a minor amount of tuning. The joint trajectories of the gait are depicted in figure 4.4 and in figures 4.5(a), 4.5(b), and 4.5(c) snapshots of one step are illustrated. In this case, the start configuration is not the same as the end



(a) Left single support (b) Double support (c) Right single support

Figure 4.3: Snapshots of a step with gait based on ZMP, solid: walking primitives; dashed: motion between walking primitives

configuration of the step. This means that successive steps are not the same, but the solution can converge to a certain limit cycle as more and more steps proceed. This convergence will be investigated with the Poincaré return map analysis.

4.4.4 ZMP gait analysis

The ZMP method can not only be used to design gaits, it can also check whether a gait is stable. The position of the ZMP can be calculated in different ways, by forward kinematics or by measuring the contact forces. From the contact forces the center of pressure on the foot can be calculated, which is very useful in online computation of the ZMP in experiments, but for simulations, the calculation based on forward kinematics is more practical. The complete derivation of the ZMP position can be found in literature, but [7] showed that for small velocities a far more simplified version can be used, the so called the cart-table model. An illustration of this model is shown in figure 4.6. It represents a driving cart with mass m on a massless table. The cart represents the position of the center of mass of the biped, and the table leg has the same shape and size as the support polygon of the biped. For this model we can derive the moment equilibrium around point p, the point where the table would tip over under influence of the cart:

$$\tau = -mg\left(x_{CoM} - p\right) + m\ddot{x}_{CoM} z_{CoM} \tag{4.2}$$

From the ZMP definition ($\tau = 0$) we can derive the ZMP in x and y:

$$x_{ZMP} = x_{CoM} - \frac{\ddot{x}_{CoM}}{g} z_{CoM} \tag{4.3}$$

$$y_{ZMP} = y_{CoM} - \frac{\ddot{y}_{CoM}}{g} z_{CoM} \tag{4.4}$$

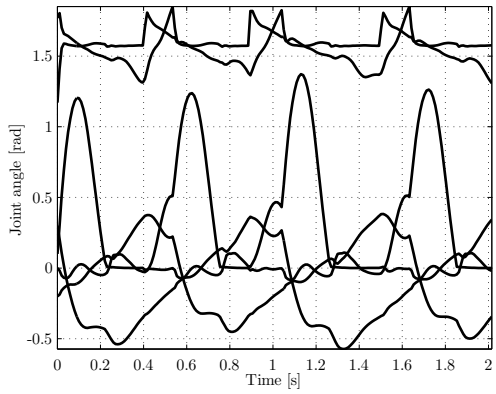
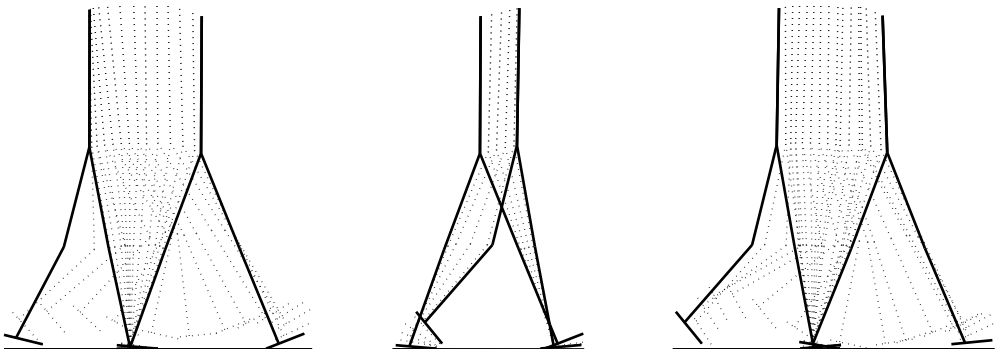


Figure 4.4: Joint angles as a function of time for a step with gait based on limit cycle walking guidelines



(a) Left single support (b) Double support (c) Right single support

Figure 4.5: Snapshots of a step with gait based on limit cycle walking guidelines

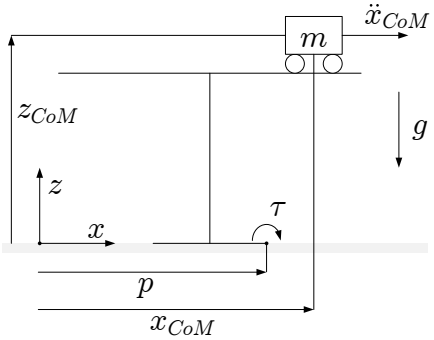
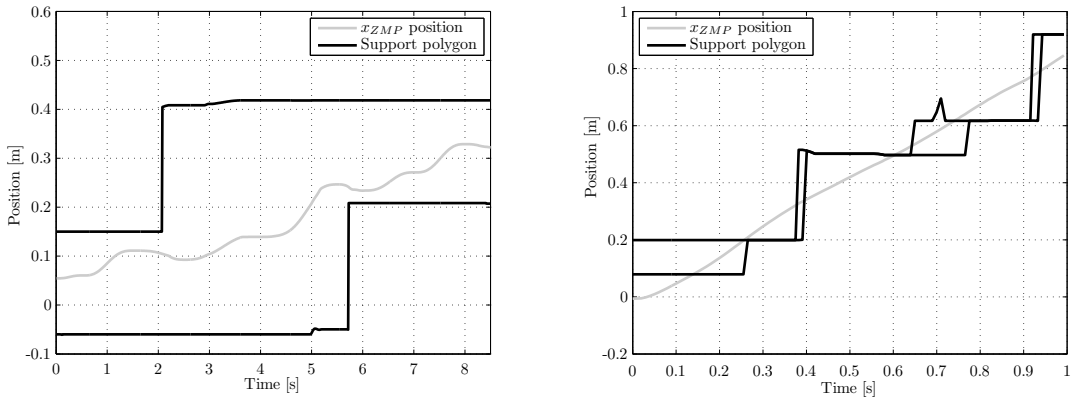


Figure 4.6: Cart-table model



(a) ZMP based gait

(b) Limit cycle walking gait

Figure 4.7: Position of the ZMP during one step

Here the position of the ZMP depends on the acceleration and state of the biped, because x_{CoM}, y_{CoM} and z_{CoM} are the components of the center of mass position vector $\underline{r}_{CoM}$ which can be calculated as follows:

$$\underline{r}_{CoM} = \frac{\sum_{i=1}^{n} m_i \underline{r}_i}{\sum_{i=1}^{n} m_i} \tag{4.5}$$

where m_i is the mass and $\underline{r}_i$ the position of the center of mass of link i.

Now we can calculate if the designed gaits are stable according to the ZMP criterion. One step is simulated with the planar model using the ZMP based as well as the limit cycle walking gait. If the position of the ZMP stays inside the support polygon, the ZMP criterion would qualify this gait as stable. In figures 4.7(a) and 4.7(b), the ZMP path from the simulation is plotted as a function of time for the ZMP based and limit cycle walking gait respectively. The support polygon is also plotted as a function of time in the same figure. Since the planar model is a 2D model, the ZMP can only move in the x direction on the horizontal ground plane.

These figures show what one initially could expect. The ZMP of the ZMP based gait stays inside the support polygon. According to the ZMP criterion this gait is stable. Nevertheless, the ZMP of the limit cycle walking gait does not stay inside the support polygon all the time. At some instances the support polygon is even just one point (i.e. in figure 4.7(b) a single black line in time) as the robot only touches the ground with one point. This agrees with figure 4.5(a), because the stance foot already starts to lift off just before impact of the swing foot, leaving only the stance toe on the ground. At these instances it would be remarkable if the ZMP stayed exactly on the stance toe. At these instances, same as in some other regions, the gait would be classified as unstable by the ZMP stability criterion. With these two examples we can see that the ZMP stability criterion is sufficient (leaving extreme cases as mentioned in section 4.2.1), but certainly not necessary to analyze the stability of a gait.

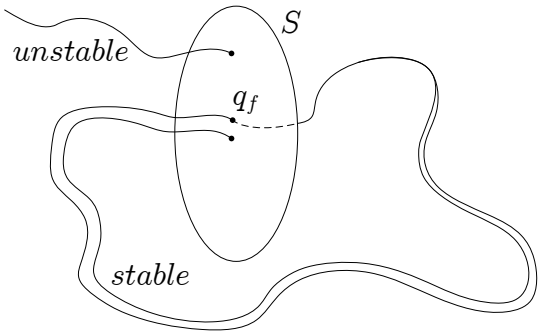


Figure 4.8: Poincaré return map

4.4.5 Poincaré gait analysis

Despite the fact that the Poincaré return map method only guarantees stability of smooth systems, we will carefully try to conclude something about the stability of the designed walking gaits. A walking gait is in principle a cyclic motion and if a system performs a cyclic motion the system will return to the same state at the end of every period of the cycle. An illustration of this behavior is depicted in figure 4.8. During its cyclic motion a system's state travels through the state space and eventually returns to the same point where it started. One can now define a certain lower dimensional subspace $\underline{S}$, which is intersected by the cyclic motion. The intersection point is called a fixed point and the location of this point is dependent on the choice of $\underline{S}$. In fact, every point on the cycle could be a fixed point intersecting a different map $\underline{S}$.

In a stable walking gait the system returns to approximately the same state every step. One can make a Poincaré map at, for example, every start of a step, just after heelstrike. This mapping of the cyclic nonlinear dynamics is called the stride function in humanoid terminology [15]. The stride function determines the state of the system after one cycle (state $\underline{q}_{n+1}$) with respect to the current state $\underline{q}_n$:

$$\underline{q}_{n+1} = \underline{S}\left(\underline{q}_n\right) \tag{4.6}$$

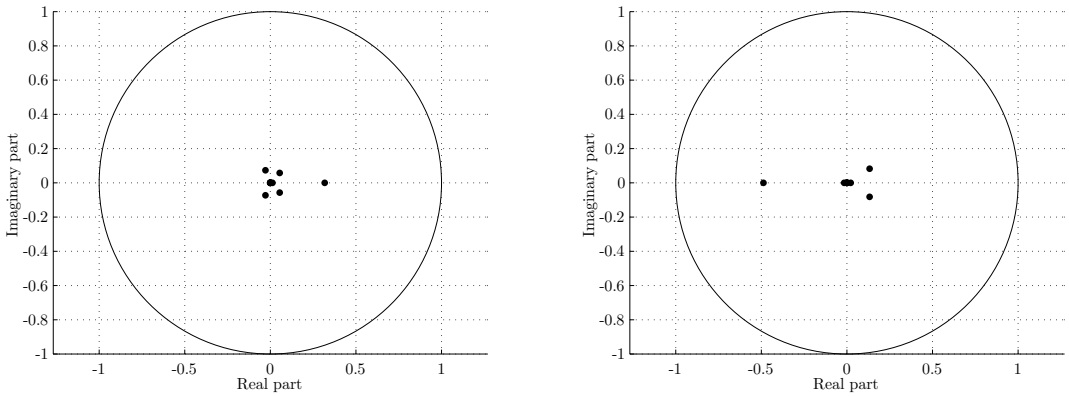
Only if the motion is perfectly cyclic, the state $\underline{q}$ is a fixed point $\underline{q}_f$:

$$\underline{q}_f = \underline{S}\left(\underline{q}_f\right) \tag{4.7}$$

The stability of the cyclic motion can be analyzed by inserting small deviations from the fixed point into the stride function. If the state returns to the fixed point, the cyclic motion is stable. Linearizing the stride function around these small deviations can tell us if the state will return to the fixed point:

$$\underline{S}\left(\underline{q}_f + \Delta\underline{q}\right) \approx \underline{q}_f + \underline{\underline{K}}\Delta\underline{q} \tag{4.8}$$

where $\underline{\underline{K}} = \frac{\partial \underline{S}}{\partial \underline{q}}$, the so called monodromy or linear return matrix. This matrix determines if the state of the system returns to the fixed point for small deviations. Namely, the motion is stable, if the eigenvalues of this matrix lie within the unit circle, i.e. the deviations decrease



(a) ZMP based gait

(b) Limit cycle walking gait

Figure 4.9: Position of the eigenvalues of the Poincaré return map

every cycle. These eigenvalues are called the Floquet multipliers.

In most cases, and also in our planar model case, the stride function can not be determined analytically. The system needs to be solved by numerical integration and the monodromy matrix can be estimated using a finite difference approximation. We can analyze the designed gaits once more to see if the Poincaré return map stability criterion gives the same results as the ZMP stability criterion. Again one step is simulated with the planar model performing a ZMP based as wel as a limit cycle walking gait. We can calculate the monodromy matrix and analyze the eigenvalues for each gait. These results are shown in figures 4.9(a) and 4.9(b) and a list of the identified eigenvalues can be found in table 4.1(a) and 4.1(b). A fixed point gait has been found by trail and error for both the ZMP based as well as the limit cycle walking situation. The initial conditions for these gaits are perturbed a small amount ($1mrad$ and $1mrad/s$ respectively) and the end result is analyzed. From the difference between the fixed point gait and the perturbed gait the monodromy matrix $\underline{K}$ can be calculated numerically. The eigenvalues of this matrix determine the cyclic stability of the gaits. Perturbing the cartesian coordinates doesn't influence the motion of the robot, so the end situation of the perturbed gait will be exactly the same as for the fixed one. That results in an eigenvalue of one, which the Poincaré map also should have. Since the cartesian perturbation can not destabilize the model, we did not include these perturbations in the research, leaving only 14 eigenvalues.

From these results we can conclude the following. All eigenvalues of both gaits lie in the unit disc. This means that the designed gaits would be cyclically stable according to the Poincaré stability criterion if we were analyzing a smooth system. However, the planar model exhibits continuous behavior, interrupted by discrete events. That is why we can not guarantee the stability of the gaits, but we might conclude some other things. Roughly half of the eigenvalues are zero, which means that they do not have a strong influence on the gait. The largest eigenvalues correspond to the orientation and angular velocity of the torso of the biped. This can be explained by the fact that the torso is the heaviest part of the robot and the highest above the ground. So the energy needed to compensate for a small deviation is the biggest for

Table 4.1: Eigenvalues of Poincaré return map

(a) ZMP based gait

λ	Eigenvalue
λ_1	0.3178
λ_2	$-0.0285 + 0.0730i$
λ_3	$-0.0285 - 0.0730i$
λ_4	$0.0545 + 0.0578i$
λ_5	$0.0545 - 0.0578i$
λ_6	0.0127
λ_7	$-3.1771 \cdot 10^{-4} + 4.2667 \cdot 10^{-4}i$
λ_8	$-3.1771 \cdot 10^{-4} - 4.2667 \cdot 10^{-4}i$
λ_9	$1.4851 \cdot 10^{-4}$
λ_{10}	$-2.7185 \cdot 10^{-5}$
λ_{11}	$1.2318 \cdot 10^{-5}$
λ_{12}	$-4.7939 \cdot 10^{-6}$
λ_{13}	$-1.9989 \cdot 10^{-6}$
λ_{14}	$-3.3579 \cdot 10^{-7}$

(b) Limit cycle walking gait

λ	Eigenvalue
λ_1	-0.4873
λ_2	$0.1326 + 0.0819i$
λ_3	$0.1326 - 0.0819i$
λ_4	0.0222
λ_5	-0.0159
λ_6	0.0005
λ_7	$1.5954 \cdot 10^{-5} + 4.5911 \cdot 10^{-5}i$
λ_8	$1.5954 \cdot 10^{-5} - 4.5911 \cdot 10^{-5}i$
λ_9	$-1.5233 \cdot 10^{-5}$
λ_{10}	$-1.9101 \cdot 10^{-5}$
λ_{11}	$-3.1599 \cdot 10^{-6}$
λ_{12}	$8.8410 \cdot 10^{-7}$
λ_{13}	$-3.8626 \cdot 10^{-8}$
λ_{14}	$9.6776 \cdot 10^{-10}$

this part of the biped. Clearly, this also results in a bigger difference in position, orientation and velocity of all links at the end of the step.

Overall, another drawback of the Poincaré stability criterion is that it can be applied only to periodic motions, since a fixed point motion is needed for the analysis. The considered class of gaits is obviously restrictive which implies that, in general, the Poincaré stability criterion is not sufficient nor necessary to analyze the stability of bipedal motions.

4.4.6 Conclusion

In this chapter we explained why it is so difficult to qualify a certain bipedal motion as stable or unstable. At this moment, no mathematical criteria of general stability of bipedal walking exist. Stability is still characterized primarily by means of qualitative criteria. In specific cases, some stability methods give us an objective qualification of stability. For an actuated robot, fulfillment of the ZMP criterion is sufficient for stability. Unfortunately this criterion guarantees stability for a rather restrictive class of walking gaits. If the ZMP criterion is not met, that does not mean that stable walking is not possible. A Poincaré return map can certainly tell if a motion converges to a limit cycle for smooth systems. However, humanoid robots do not exhibit smooth behavior in general, so this method can not be used to prove stability of arbitrary gaits. Nevertheless, it can be used to extract more information from the dynamics than using the ZMP method. Furthermore, limit cycles like gaits are just one class of all walking gaits, so Poincaré return method is not generally applicable. General stability qualification that includes all possible gaits an arbitrary biped can make is still not possible. Nevertheless, the existing stability criteria can be used to design a wide range of bipedal motions. We showed how to derive two different kinds of gaits: a ZMP based as well as a limit cycle walking gait. Furthermore we showed how to analyse the stability of a certain gait. Overall we might conclude that the existing stability criteria are applicable only in specific

cases, so we are still in infancy regarding general qualification of stability of bipedal walking. Consequently, further research is needed to achieve a general stability criterion.

Chapter 5

Conclusions and recommendations

5.1 Conclusions

5.1.1 Modeling

In this report we investigated different ways of modeling a humanoid robot. The Newton-Euler method incorporates the internal and external forces applied to the links in the system, which results in a differential algebraic set of equations. Consequently, the major drawback of this method is that it is hard to numerically solve the system. The Lagrange-Euler method in general solves this problem because it uses generalized coordinates. The disadvantage of the Lagrange-Euler method is that for complex systems, the analytical computation of the partial derivatives becomes cumbersome. The TMT method combines the advantages of the Newton-Euler as well as the Lagrange-Euler method, but with the current set up it is not possible to derive the correct equations of motion in 3D, due to the use of an analytical Jacobian. Hence, as the first contribution of this work, we propose a general framework for automatic derivation of equations of motion of a humanoid robots. This framework is based on the Denavit Hartenberg convention for describing robot kinematics and the modeling method of Lagrange-Euler. Practical usability of the framework is illustrated on the humanoid robot TUlip. Equations of motion of TUlip are easily and efficiently derived and then implemented numerically using an event detection solver. This solver has been chosen because of its time efficiency against other alternatives, for instance the time-stepping method. A general drawback of the event detection method, not being suitable if many events occur, does not hold for the model of TUlip. Namely, the existing model does feature a limited number of events because of an inelastic impact assumption. Furthermore, by considering only four contact points per feet, we practically limit the variety of events. The resulting analytical model is very complex and, consequently, numerical simulations take a long time. However, complexity of the dynamical equations is a general characteristics of humanoid robots.

5.1.2 Identification

As the second contribution, we present an algorithm that derives the regressor form of equations of motion automatically. This algorithm is also successfully validated on the humanoid robot TUlip. The kinematic and inertial parameters that appear in the model are identified in dynamic experiments. The parameters are estimated for each leg separately, because in the current set up it is not possible to identify the complete robot at once. Experiments

have been done on TUlip and the unknown parameters from the model have been estimated. The quality of the resulting estimates is verified in experiments, confirming that the proposed approach can be used for identification of a humanoid robot.

5.1.3 Stability

As the final contribution, we show that the conventional criteria for analysis of stability of biped walking are neither necessary nor sufficient in general sense. Namely, these criteria can not qualify stability in all different environments and the complete diversity of tasks a humanoid should do. We illustrate this using the criteria ZMP and Poincaré return map. The former criterion is used to design a stable gait. A trial and error approach is used to design a gait for limit cycle walking, since no systematic approach exists to design such a gait. Stability of both gaits can be analyzed by means of the Poincaré map, keeping in mind that overall humanoid robots exhibit non-smooth behavior, while the ZMP criterion fails for the limit cycle walking gait. This is an illustrative example of limitations of the existing criteria for stability analysis.

5.2 Recommendations

5.2.1 Theory

During this research experience has been gained about modeling, identification and stability of humanoid robots. This experience could be used to further increase the accuracy of the model as well as to further investigate stability criteria. Regarding modeling, the TMT method could be rewritten such that it incorporates the use of geometrical Jacobians and could be used for 3D modeling. Furthermore, the model of TUlip, derived in this report, could be implemented using a time stepping numerical solver. This might lead to some advantages, such as the possibility to model ground friction and inelastic impacts which could be numerically problematic in an event detection solver. However, one should always weigh the accuracy of the model against the computational time, as we expect that a time stepping solver requires more execution time.

Modeling the non linear friction in the ankles of TUlip would improve the identification process. Furthermore, if Cartesian position and orientation of the torso of the robot is available together with pressure information on its feet, it should be possible to identify all parameters of the complete robot at once.

Regarding stability, new concepts should be derived that cover a larger space of the bipeds motion as we will eventually need a complete theory that guarantees stability in every possible situation and environment. We realize that this is a task for years and years of research.

5.2.2 TUlip

During experiments on TUlip, a couple of mechanical issues are noticed. Firstly, TUlip lacks ankle X actuation, which clearly is needed for stable walking and should be added as soon as possible. Furthermore, the ankle Y actuation is only in one direction actively controlled, the other direction is controlled by a spring, which is not desirable. Recommended is to change

this in a bilateral actuation and since this actuation also faces high friction, we believe that the entire ankle Y actuation design should be revised.

Finally we remark that TUlip is primarily designed for limit cycle walking. Its center of mass is placed as high as possible and the position bandwidth is low due to the series elastic actuation. It is believed that in order to walk in a limit cycle, the priority should not lie on decreasing the position error as far as possible. However, for statically stable walking, such as using the zero moment point, this is crucial. Thus it is recommended to choose between one and the other, because in the writers opinion it is impossible to design a humanoid based on both principles.

Bibliography

[1] K. Ayusawa, G. Venture, and Y. Nakamura. Identification of humanoid robots dynamics using floating-base motion dynamics. *IEEE/RSJ international conference on intelligent robots and systems*, pages 2854–2859, 2008.

[2] S.W. Boere and P.W.M. van Zutven. Knee flexion during push off in limit cycle walkers. Technical report, Eindhoven University of Technology, department of Mechanical Engineering, 2008.

[3] M. Brubaker. Physics based priors for human pose tracking. Master's thesis, University of Toronto, department of Computer Science, 2006.

[4] G. Calafiore, M. Indri, and B. Bona. Robot dynamic calibration: optimal excitation trajectories and experimental parameter estimation. *Journal of robotic systems 18(2)*, pages 55–68, 2001.

[5] S. Collins. Flame system identification results. Technical report, Delft University of Technology, 2008.

[6] B. de Kraker and D. van Campen. *Mechanical Vibrations*. Maastricht: Shaker Publishing, the Netherlands, 2001.

[7] M.H.P. Dekker. Zero-moment point method for stable biped walking. Technical report, Eindhoven University of Technology, department of Mechanical Engineering, 2009.

[8] J. Denavit and R.S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of applied mechanics*, page 215221, 1955.

[9] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman. The simplest walking model: Stability, complexity and scaling. *ASME Journal of Biomechanical Engineering*, 1998.

[10] M. Gautier and W. Khalil. On the identification of the inertial parameters of robots. *Proceedings of the 27th IEEE conference on decision and control*, pages 2264–2269, 1988.

[11] M. Gautier and W. Khalil. Identification of the minimum inertial parameters of robots. *Proceedings of the IEEE conference on robotics and automation*, pages 1529–1534, 1989.

[12] M. Gautier and W. Khalil. Direct calculation of minimum set of inertial parameters of serial robots. *IEEE transactions on robotics and automation*, pages 368–373, 1990.

[13] M. Gautier and W. Khalil. Exciting trajectories for the identification of base inertial parameters of robots. *Proceedings of the 30th IEEE conference on decision and control*, pages 494–499, 1991.

[14] K. Griffioen. Tulip measurement methods. Technical report, Delft University of Technology, 2009.

[15] D.G.E. Hobbelen. *Limit cycle walking*. PhD thesis, Delft University of Technology, 2008.

[16] K.G. Jolly, R. Sreerama Kumar, and R. Vijayakumar. A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems*, pages 23–33, 2008.

[17] H. Kawasaki, T. Shimizu, and K. Kanzaki. Symbolic analysis of the base parameters for closed-chain robots based on the completion procedure. *Proceedings of the 1996 IEEE international conference on robotics and automation*, pages 1781–1786, 1996.

[18] G. Lakemeyer, E. Sklar, D.G. Sorrenti, and T. Takahashi. *RoboCup 2006: Robot Soccer World Cup X*. Berlin: Springer Verlag, Germany, 2007.

[19] R. Leine and H. Nijmeijer. *Dynamics and bifurcation of non smooth mechanical systems*. Berlin: Springer Verlag, Germany, 2004.

[20] G.F. Melson, P.H. Kahn, A.M. Beck, B. Friedman, T. Roberts, and E. Garrett. Robots as dog? - children's interactions with the robotic dog aibo and a live australian shepard. *Conference on human factors in computing systems*, pages 1649 – 1652, 2005.

[21] M.Vukobratovic and M.Kircanski. *Kinematics and trajectory synthesis of manipulation robots*. Berlin: Springer Verlag, Germany, 1986.

[22] M.M. Olsen and H.G. Petersen. A new method for estimating parameters of a dynamic robot model. *IEEE transactions on robotics and automation, vol 17, no 1*, pages 95–100, 2001.

[23] E.A.C.A. Peeters. Legged locomotion: towards reinforcement learning dynamics interpretation. Master's thesis, Eindhoven University of Technology, 2009.

[24] J.E. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: a step toward humanoid push recovery. *Proceedings of the 2006 IEEE-RAS international conference on humanoid robots*, pages 200–207, 2006.

[25] J.E. Pratt and R. Tedrake. Velocity-based stability margins for fast bipedal walking. *First Ruperto Carola symposium: fast motions in biomechanics and robots*, 2005.

[26] A.L. Schwab and R.Q. van der Linde. Multibody dynamics b. Technical report, Delft University of Technology, department of Engineering Mechanics, 1997.

[27] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*. New York: John Wiley and Sons, Inc., 2006.

[28] J. Swevers, C. Ganseman, D.B. Tkel, J. de Schutter, and H. van Brussel. Optimal robot excitation and identification. *IEEE transactions on robotics and automation, vol 13, no 5*, pages 730–740, 1997.

[29] Y. Takahashi. Systematic controller design for limit cycle walking. Master's thesis, Delft University of Technology, 2009.

[30] N. van de Wouw. *Multibody Dynamics*. Eindhoven: University Press Facilities, the Netherlands, 2007.

[31] B.J.M. van Schaik. Base parameter identification and decoupling of a double scara robot. Master's thesis, Eindhoven University of Technology, 2008.

[32] M.P. Verrijt. Model-based prediction of joint torques on an anthropomorphic robot arm. Technical report, Eindhoven University of Technology, department of Mechanical Engineering, 2009.

[33] M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic. *Scientific Fundamentals of Robotics 7: Biped Locomotion*. New York: Springer, the United States, 1990.

[34] R. Waiboer and R. Aarts. Modelling and identification of a six axes industrial robot. *Proceedings of the ASME international design engineering technical conferences and computers and information in engineering conference*, pages 1–10, 2005.

[35] E.R. Westervelt, J.W. Grizzle, C. Chevallereau, J.H. Choi, and B. Morris. *Feedback control of dynamic bipedal robot locomotion*. New York: CRC Press, 2007.

[36] M. Wisse. *Essentials of dynamic walking*. PhD thesis, Delft University of Technology, 2004.

Appendix A

Double pendulum example

In this appendix, all theories and algorithms presented and described in this work are further explained using analytical examples. These examples are based on a model of a double pendulum, shown in figures A.1(a), A.1(b) and A.1(c).

This 2D model consists of two inter-connected links (indices 1 and 2 respectively) with mass m, moment of inertia I, length l and center of mass position c. The center of mass positions in Cartesian space are described by x and y and the generalized coordinates are indicated by θ.

A.1 Modeling

In this section, the equations of motion of the double pendulum are derived using four different methods. We will see that they all result in the same expressions.

A.1.1 Newton Euler

Deriving the equations of motion using Newton Euler, first all forces and moments acting on the bodies need to be defined. In figure A.1(c) all forces, reaction forces and moments that act on each body are depicted. The force equilibria can be derived in horizontal and vertical direction, according to Newton these are equal to the acceleration of the body times its mass:

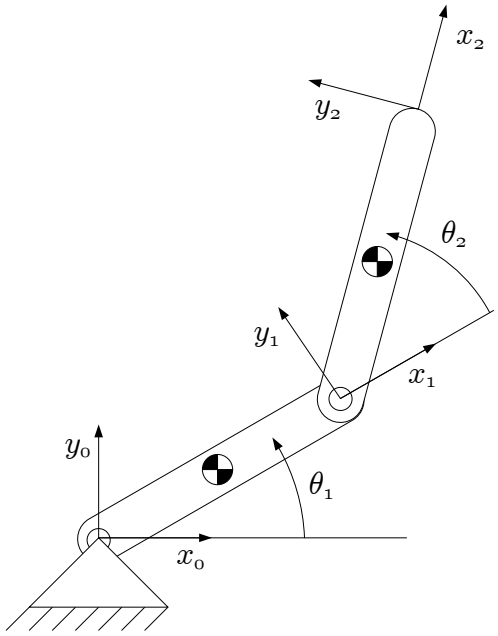
$$m_1\ddot{x}_1 = F_{h1} - F_{h2} \tag{A.1}$$
$$m_1\ddot{y}_1 = F_{v1} - F_{v2} - F_{z1} \tag{A.2}$$

$$m_2\ddot{x}_2 = F_{h2} \tag{A.3}$$
$$m_2\ddot{y}_2 = F_{v2} - F_{z2} \tag{A.4}$$

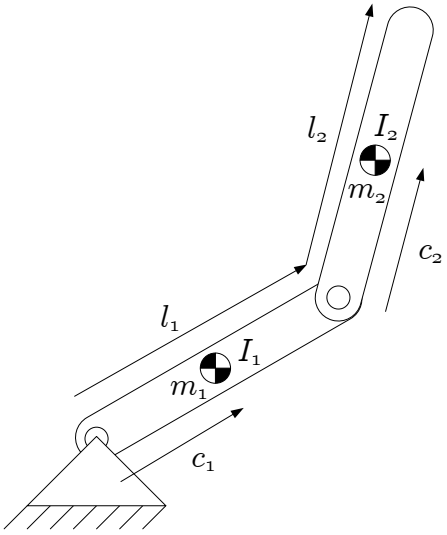
Euler states that the moment equilibria are equal to the moment of inertia times the angular acceleration of the body:

$$I_1\ddot{\varphi}_1 = \tau_1 - \tau_2 + F_{h1}c_1\sin(\varphi_1) + F_{h2}d_1\sin(\varphi_1) - F_{v1}c_1\cos(\varphi_1) - F_{v2}d_1\cos(\varphi_1) \tag{A.5}$$

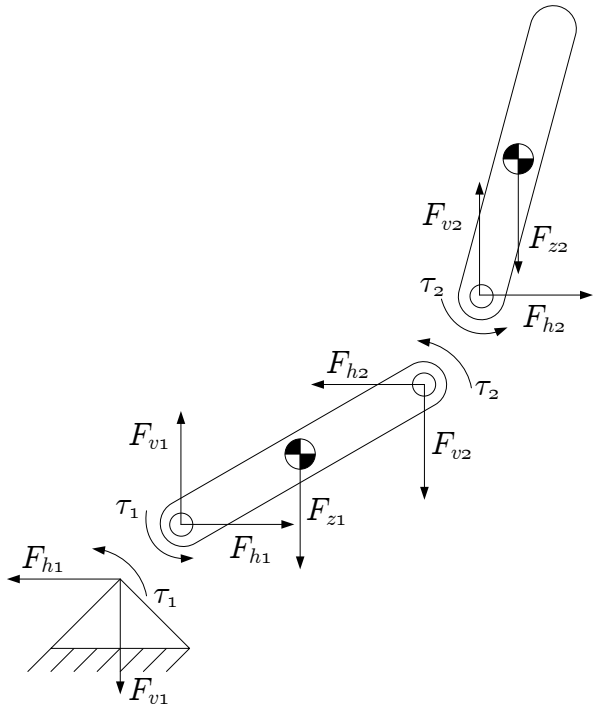
$$I_2(\ddot{\varphi}_1 + \ddot{\varphi}_2) = \tau_2 + F_{h2}c_2\sin(\varphi_2) - F_{v2}c_2\cos(\varphi_2) \tag{A.6}$$



(a) Coordinates



(b) Parameters



(c) Forces and torques

Figure A.1: Drawing of the double pendulum model

where $d_1 = l_1 - c_1$, $\varphi_1 = \theta_1$ and $\varphi_2 = \theta_1 + \theta_2$. These six equations are described by ten unknowns: the six body accelerations and four constraint forces. To solve the system four more equations are needed, the equations that describe the constraints of the joints. Body 1 is attached to the fixed world by a revolute joint, this point should be kept in place:

$$x_1 - c_1 \cos(\varphi_1) = 0 \tag{A.7}$$
$$y_1 - c_1 \sin(\varphi_1) = 0 \tag{A.8}$$

Furthermore, bodies 1 and 2 are attached to each other by a revolute joint:

$$x_1 + d_1 \cos(\varphi_1) = x_2 - c_2 \cos(\varphi_2) \tag{A.9}$$
$$y_1 + d_1 \sin(\varphi_1) = y_2 - c_2 \sin(\varphi_2) \tag{A.10}$$

Differentiating these twice with respect to time leads to the constraint equations:

$$\ddot{x}_1 + c_1 \cos(\varphi_1)\dot{\varphi}_1^2 + c_1 \sin(\varphi_1)\ddot{\varphi}_1 = 0 \tag{A.11}$$
$$\ddot{y}_1 + c_1 \sin(\varphi_1)\dot{\varphi}_1^2 - c_1 \cos(\varphi_1)\ddot{\varphi}_1 = 0 \tag{A.12}$$
$$\ddot{x}_1 - d_1 \cos(\varphi_1)\dot{\varphi}_1^2 - d_1 \sin(\varphi_1)\ddot{\varphi}_1 = \ddot{x}_2 + c_2 \cos(\varphi_2)\dot{\varphi}_2^2 + c_2 \sin(\varphi_2)\ddot{\varphi}_2 \tag{A.13}$$
$$\ddot{y}_1 - d_1 \sin(\varphi_1)\dot{\varphi}_1^2 + d_1 \cos(\varphi_1)\ddot{\varphi}_1 = \ddot{y}_2 + c_2 \sin(\varphi_2)\dot{\varphi}_2^2 - c_2 \cos(\varphi_2)\ddot{\varphi}_2 \tag{A.14}$$

The Newton-Euler equations and the constraint equations form the complete set of the equations of motion that describe the dynamics of the system. This set of equations can analytically be solved for τ_1 and τ_2 to write it in a more common like way using the fact that g is the gravitational constant, $d_1 = l_1 - c_1$, $\varphi_1 = \theta_1$, $\varphi_2 = \theta_1 + \theta_2$, $F_{z1} = m_1 g$, $F_{z2} = m_2 g$ and $\underline{q} = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^T$:

$$\underline{\tau} = \underline{D}\left(\underline{q}\right)\ddot{\underline{q}} + \underline{C}\left(\underline{q}, \dot{\underline{q}}\right)\dot{\underline{q}} + \underline{G}\left(\underline{q}\right) \tag{A.15}$$

where

$$\underline{\tau} = \begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} \tag{A.16}$$
$$\underline{D} = \begin{bmatrix} m_1 c_1^2 + m_2\left(l_1^2 + c_2^2 + 2 l_1 c_2 \cos(\theta_2)\right) + I_1 + I_2 & m_2\left(c_2^2 + l_1 c_2 \cos(\theta_2)\right) + I_2 \\ m_2\left(c_2^2 + l_1 c_2 \cos(\theta_2)\right) + I_2 & m_2 c_2^2 + I_2 \end{bmatrix} \tag{A.17}$$
$$\underline{C} = \begin{bmatrix} -m_2 l_1 c_2 \sin(\theta_2)\dot{\theta}_2 & -m_2 l_1 c_2 \dot{\theta}_2 \sin(\theta_2)\left(\dot{\theta}_1 + \dot{\theta}_2\right) \\ m_2 l_1 c_2 \sin(\theta_2)\dot{\theta}_1 & 0 \end{bmatrix} \tag{A.18}$$
$$\underline{G} = \begin{bmatrix} (m_1 c_1 + m_2 l_1) g \cos(\theta_1) + m_2 c_2 g \cos(\theta_1 + \theta_2) \\ m_2 c_2 g \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{A.19}$$

A.1.2 Lagrange-Euler

To derive the equations of motion using Lagrange, energy relations need to be found. First, consider the position of the centers of mass as a function of the generalized coordinates:

$$x_1 = c_1 \cos(\theta_1) \tag{A.20}$$
$$y_1 = c_1 \sin(\theta_1) \tag{A.21}$$

$$x_2 = l_1 \cos(\theta_1) + c_2 \cos(\theta_1 + \theta_2) \tag{A.22}$$
$$y_2 = l_1 \sin(\theta_1) + c_2 \sin(\theta_1 + \theta_2) \tag{A.23}$$

Differentiating these expressions to time results in the velocities of the centers of mass as function of the generalized coordinates:

$$\dot{x}_1 = -c_1 \sin(\theta_1)\,\dot{\theta}_1 \tag{A.24}$$
$$\dot{y}_1 = c_1 \cos(\theta_1)\,\dot{\theta}_1 \tag{A.25}$$

$$\dot{x}_2 = (-l_1 \sin(\theta_1) - c_2 \sin(\theta_1 + \theta_2))\,\dot{\varphi}_1 - c_2 \sin(\theta_1 + \theta_2)\,\dot{\varphi}_2 \tag{A.26}$$
$$\dot{y}_2 = (l_1 \cos(\theta_1) + c_2 \cos(\theta_1 + \theta_2))\,\dot{\varphi}_1 + c_2 \cos(\theta_1 + \theta_2)\,\dot{\varphi}_2 \tag{A.27}$$

The kinematic and potential energy of this system now become:

$$\begin{aligned} K &= \frac{1}{2}\left(m_1\left(\dot{x}_1^2 + \dot{y}_1^2\right) + m_2\left(\dot{x}_2^2 + \dot{y}_2^2\right) + I_1\dot{\theta}_1^2 + I_2\left(\dot{\theta}_2 + \dot{\theta_2}^2\right)\right) \\ &= \frac{1}{2}\left(m_1 c_1^2 + m_2 l_1^2 + m_2 c_2^2 + 2m_2 l_1 c_2 \cos(\theta_2) + I_1 + I_2\right)\dot{\varphi}_1^2 \\ &\quad + \frac{1}{2}\left(m_2 c_2^2 + I_2\right)\dot{\varphi}_2^2 + \left(m_2 c_2^2 + m_2 l_1 c_2 \cos(\theta_2) + I_2\right)\dot{\varphi}_1\dot{\varphi}_2 \end{aligned} \tag{A.28}$$

$$P = m_1 g y_1 + m_2 g y_2 \tag{A.29}$$
$$= m_1 g c_1 \cos(\varphi_1) + m_2 g\left(l_1 \sin(\theta_1) + c_2 \cos(\theta_1 + \theta_2)\right) \tag{A.30}$$

The kinematic and potential energy relations can be used to fill in the Lagrange equation:

$$\frac{d}{dt}\frac{\partial K}{\partial \underline{\dot{q}}} - \frac{\partial K}{\partial \underline{q}} + \frac{\partial P}{\partial \underline{q}} = \underline{Q}_{nc}^T \tag{A.31}$$

where $\underline{q} = \begin{bmatrix}\varphi_1 & \varphi_2\end{bmatrix}^T$ and $\underline{Q}_{nc} = \begin{bmatrix}\tau_1 & \tau_2\end{bmatrix}$ in this case. Filling everything in and rearranging, this finally results in the equations of motion of the double pendulum system:

$$\underline{\tau} = \underline{D}\left(\underline{q}\right)\ddot{\underline{q}} + \underline{C}\left(\underline{q}, \dot{\underline{q}}\right)\dot{\underline{q}} + \underline{G}\left(\underline{q}\right) \tag{A.32}$$

where

$$\underline{\tau} = \begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} \tag{A.33}$$
$$\underline{D} = \begin{bmatrix} m_1 c_1^2 + m_2\left(l_1^2 + c_2^2 + 2l_1 c_2 \cos(\theta_2)\right) + I_1 + I_2 & m_2\left(c_2^2 + l_1 c_2 \cos(\theta_2)\right) + I_2 \\ m_2\left(c_2^2 + l_1 c_2 \cos(\theta_2)\right) + I_2 & m_2 c_2^2 + I_2 \end{bmatrix} \tag{A.34}$$
$$\underline{C} = \begin{bmatrix} -m_2 l_1 c_2 \sin(\theta_2)\,\dot{\theta}_2 & -m_2 l_1 c_2 \dot{\theta}_2 \sin(\theta_2)\left(\dot{\theta}_1 + \dot{\theta}_2\right) \\ m_2 l_1 c_2 \sin(\theta_2)\,\dot{\theta}_1 & 0 \end{bmatrix} \tag{A.35}$$
$$\underline{G} = \begin{bmatrix} (m_1 c_1 + m_2 l_1)\, g \cos(\theta_1) + m_2 c_2 g \cos(\theta_1 + \theta_2) \\ m_2 c_2 g \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{A.36}$$

A.1.3 TMT

Using the TMT method, we first define $\underline{R}_1$ and $\underline{R}_2$, the rotation matrices of link 1 and link 2 with respect to base frame respectively:

$$\underline{R}_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) \end{bmatrix} \tag{A.37}$$

$$\underline{R}_2^0 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) \end{bmatrix} \tag{A.38}$$

Then the position of joints 1 and 2 is defined as:

$$\underline{j}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{A.39}$$

$$\underline{j}_2 = \underline{R}_1 \begin{bmatrix} l_1 \\ 0 \end{bmatrix} = \begin{bmatrix} l_1\cos(\theta_1) \\ l_1\sin(\theta_1) \end{bmatrix} \tag{A.40}$$

Now the positions and orientations of the center of mass of link 1 and link 2 with respect to base frame become:

$$\underline{p}_1 = \underline{j}_1 + \begin{bmatrix} \underline{R}_1^0 \begin{bmatrix} c_1 \\ 0 \end{bmatrix} \\ \theta_1 \end{bmatrix} = \begin{bmatrix} c_1\cos(\theta_1) \\ c_1\sin(\theta_1) \\ \theta_1 \end{bmatrix} \tag{A.41}$$

$$\underline{p}_2 = \underline{j}_2 + \begin{bmatrix} \underline{R}_2^0 \begin{bmatrix} c_2 \\ 0 \end{bmatrix} \\ \theta_1+\theta_2 \end{bmatrix} = \begin{bmatrix} l_1\cos(\theta_1)+c_2\cos(\theta_1+\theta_2) \\ l_1\sin(\theta_1)+c_2\sin(\theta_1+\theta_2) \\ \theta_1+\theta_2 \end{bmatrix} \tag{A.42}$$

$$\underline{T} = \begin{bmatrix} \underline{p}_1 \\ \underline{p}_2 \end{bmatrix} \tag{A.43}$$

Furthermore, the complete mass matrix of the system can be described by:

$$\underline{M} = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_2 \end{bmatrix} \tag{A.44}$$

The applied forces to the system are described by:

$$\underline{F} = M \begin{bmatrix} 0 \\ -g \\ 0 \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \tau_1-\tau_2 \\ 0 \\ 0 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -m_1 g \\ \tau_1-\tau_2 \\ 0 \\ -m_2 g \\ \tau_2 \end{bmatrix} \tag{A.45}$$

Combining everything, we can fill in the TMT expression:

$$\frac{\partial \underline{T}}{\partial \underline{q}} \underline{M} \frac{\partial \underline{T}}{\partial \underline{q}} \ddot{\underline{q}} = \frac{\partial \underline{T}}{\partial \underline{q}} \underline{F} - \frac{\partial \underline{T}}{\partial \underline{q}} \underline{M}\underline{g} \tag{A.46}$$

Table A.1: Denavit Hartenberg parameters for double pendulum model

Link	a_i	α_i	d_i	θ_i
1	l_1	0	0	θ_1
2	l_2	0	0	θ_2

where $\underline{g} = \frac{\partial}{\partial \underline{q}} \left(\frac{\partial T}{\partial \underline{q}} \dot{\underline{q}} \right) \dot{\underline{q}}$. Filling in and rearranging gives the complete equations of motion of the double pendulum system:

$$\underline{\tau} = \underline{D}\left(\underline{q}\right) \ddot{\underline{q}} + \underline{C}\left(\underline{q}, \dot{\underline{q}}\right) \dot{\underline{q}} + \underline{G}\left(\underline{q}\right) \tag{A.47}$$

where

$$\underline{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{A.48}$$

$$\underline{D} = \begin{bmatrix} m_1 c_1^2 + m_2 \left(l_1^2 + c_2^2 + 2 l_1 c_2 \cos\left(\theta_2\right) \right) + I_1 + I_2 & m_2 \left(c_2^2 + l_1 c_2 \cos\left(\theta_2\right) \right) + I_2 \\ m_2 \left(c_2^2 + l_1 c_2 \cos\left(\theta_2\right) \right) + I_2 & m_2 c_2^2 + I_2 \end{bmatrix} \tag{A.49}$$

$$\underline{C} = \begin{bmatrix} -m_2 l_1 c_2 \sin\left(\theta_2\right) \dot{\theta}_2 & -m_2 l_1 c_2 \dot{\theta}_2 \sin\left(\theta_2\right) \left(\dot{\theta}_1 + \dot{\theta}_2 \right) \\ m_2 l_1 c_2 \sin\left(\theta_2\right) \dot{\theta}_1 & 0 \end{bmatrix} \tag{A.50}$$

$$\underline{G} = \begin{bmatrix} \left(m_1 c_1 + m_2 l_1 \right) g \cos\left(\theta_1\right) + m_2 c_2 g \cos\left(\theta_1 + \theta_2\right) \\ m_2 c_2 g \cos\left(\theta_1 + \theta_2\right) \end{bmatrix} \tag{A.51}$$

A.1.4 Denavit Hartenberg

Using the Denavit Hartenberg convention firstly the DH parameters need to be defined, where a_i is the link length, α_i the link twist, d_i the link offset and θ_i the joint angle. For a definition see [27]. The DH parameters for the double pendulum model are defined in table A.1. With the DH parameters we can define the homogenous transformation matrices from frame i to frame $i-1$:

$$\underline{A}_1^0 = \begin{bmatrix} \cos\left(\theta_1\right) & -\sin\left(\theta_1\right) & 0 & l_1 \cos\left(\theta_1\right) \\ \sin\left(\theta_1\right) & \cos\left(\theta_1\right) & 0 & l_1 \sin\left(\theta_1\right) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.52}$$

$$\underline{A}_2^1 = \begin{bmatrix} \cos\left(\theta_2\right) & -\sin\left(\theta_2\right) & 0 & l_2 \cos\left(\theta_2\right) \\ \sin\left(\theta_2\right) & \cos\left(\theta_2\right) & 0 & l_2 \sin\left(\theta_2\right) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.53}$$

Furthermore, the homogenous transformation matrices from frame 1 and 2 to the base frame are given by:

$$\underline{T}_1^0 = \underline{A}_1^0 = \begin{bmatrix} \cos\left(\theta_1\right) & -\sin\left(\theta_1\right) & 0 & l_1 \cos\left(\theta_1\right) \\ \sin\left(\theta_1\right) & \cos\left(\theta_1\right) & 0 & l_1 \sin\left(\theta_1\right) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \underline{R}_1^0 & \underline{o}_1^0) \\ \underline{0} & 1 \end{bmatrix} \tag{A.54}$$

$$\underline{T}_2^0 = \underline{A}_0^1 \underline{A}_1^2 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & l_1\cos(\theta_1)+l_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & l_1\sin(\theta_1)+l_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \underline{R}_2^0 & \underline{o}_2^0) \\ \underline{0} & 1 \end{bmatrix} \tag{A.55}$$

Next the positions of the centers of mass of link i with respect to frame i are defined:

$$\underline{o}_{c,1}^1 = \begin{bmatrix} -l_1 + c_1 \\ 0 \\ 0 \end{bmatrix} \tag{A.56}$$

$$\underline{o}_{c,2}^2 = \begin{bmatrix} -l_2 + c_2 \\ 0 \\ 0 \end{bmatrix} \tag{A.57}$$

These can easily be transformed to the positions of the centers of mass of link 1 and 2 with respect to the base frame:

$$\underline{o}_{c1}{}^0 = \underline{o}_1^0 + \underline{R}_1^0 \underline{o}_{c1}{}^1 = \begin{bmatrix} c_1\cos(\theta_1) \\ c_1\sin(\theta_1) \\ 0 \end{bmatrix} \tag{A.58}$$

$$\underline{o}_{c2}{}^0 = \underline{o}_2^0 + \underline{R}_2^0 \underline{o}_{c2}{}^2 = \begin{bmatrix} l_1\cos(\theta_1)+c_2\cos(\theta_1+\theta_2) \\ l_1\sin(\theta_1)+c_2\sin(\theta_1+\theta_2) \\ 0 \end{bmatrix} \tag{A.59}$$

The velocity Jacobians can now be derived geometrically using these positions and the definitions presented in [27]:

$$\underline{J}_{v1}{}^0 = \begin{bmatrix} \underline{J}_{v1,1}{}^0 & \underline{0} \end{bmatrix} \tag{A.60}$$

$$\underline{J}_{v2}{}^0 = \begin{bmatrix} \underline{J}_{v2,1}{}^0 & \underline{J}_{v2,2}{}^0 \end{bmatrix} \tag{A.61}$$

$$\underline{J}_{\omega1}{}^0 = \begin{bmatrix} \underline{J}_{\omega1,1}{}^0 & \underline{0} \end{bmatrix} \tag{A.62}$$

$$\underline{J}_{\omega2}{}^0 = \begin{bmatrix} \underline{J}_{\omega2,1}{}^0 & \underline{J}_{\omega2,2}{}^0 \end{bmatrix} \tag{A.63}$$

where

$$\underline{J}_{v1,1}{}^0 = \underline{z}_0^0 \times \left(\underline{o}_{c1}{}^0 - \underline{o}_0^0\right) = \begin{bmatrix} -c_1\sin(\theta_1) \\ c_1\cos(\theta_1) \\ 0 \end{bmatrix} \tag{A.64}$$

$$\underline{J}_{v2,1}{}^0 = \underline{z}_0^0 \times \left(\underline{o}_{c2}{}^0 - \underline{o}_0^0\right) = \begin{bmatrix} -l_1\sin(\theta_1)-c_2\sin(\theta_1+\theta_2) \\ l_1\cos(\theta_1)+c_2\cos(\theta_1+\theta_2) \\ 0 \end{bmatrix} \tag{A.65}$$

$$\underline{J}_{v2,2}{}^0 = \underline{z}_1^0 \times \left(\underline{o}_{c2}{}^0 - \underline{o}_1^0\right) = \underline{R}_1^0 \underline{z}_1^1 \times \left(\underline{o}_{c2}{}^0 - \underline{o}_1^0\right) = \begin{bmatrix} -c_2\sin(\theta_1+\theta_2) \\ c_2\cos(\theta_1+\theta_2) \\ 0 \end{bmatrix} \tag{A.66}$$

and

$$\underline{J}_{\omega 1,1}{}^{0} = \underline{z}_0^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.67}$$

$$\underline{J}_{\omega 2,1}{}^{0} = \underline{z}_0^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.68}$$

$$\underline{J}_{\omega 2,2}{}^{0} = \underline{z}_1^0 = \underline{R}_1^0 \underline{z}_1^1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{A.69}$$

with $\underline{z}_0^0 = \underline{z}_1^1 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ and $\underline{o}_0^0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$.
Finally, we can substitute the velocity Jacobians into the kinetic energy relation:

$$K = \frac{1}{2}\dot{\underline{q}}^T \left(\sum_{i=1}^{2} m_i \underline{J}_{vi}{}^{0T} \underline{J}_{vi}{}^{0} + \underline{J}_{\omega i}{}^{0T} \underline{R}_i^0 \underline{I}_i \underline{R}_i^{0T} \underline{J}_{\omega i}{}^{0} \right) \dot{\underline{q}} \tag{A.70}$$

where $\underline{I}_i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_i \end{bmatrix}$ the inertia matrix for link 1 and 2.
The potential energy is defined as:

$$P = \sum_{i=1}^{n} m_i \underline{g}^T \underline{o}_{ci}{}^{0} \tag{A.71}$$

where $\underline{g} = \begin{bmatrix} 0 & g & 0 \end{bmatrix}$.
Filling these in into the Lagrange equations gives the equations of motion:

$$\frac{d}{dt}\frac{\partial K}{\partial \dot{\underline{q}}} - \frac{\partial K}{\partial \underline{q}} + \frac{\partial P}{\partial \underline{q}} = \underline{Q}_{nc}^T \tag{A.72}$$

Rearranging eventually gives us again the same equations of motion in matrix form:

$$\underline{\tau} = \underline{D}\left(\underline{q}\right)\ddot{\underline{q}} + \underline{C}\left(\underline{q}, \dot{\underline{q}}\right)\dot{\underline{q}} + \underline{G}\left(\underline{q}\right) \tag{A.73}$$

where

$$\underline{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{A.74}$$

$$\underline{D} = \begin{bmatrix} m_1 c_1^2 + m_2\left(l_1^2 + c_2^2 + 2 l_1 c_2 \cos\left(\theta_2\right)\right) + I_1 + I_2 & m_2\left(c_2^2 + l_1 c_2 \cos\left(\theta_2\right)\right) + I_2 \\ m_2\left(c_2^2 + l_1 c_2 \cos\left(\theta_2\right)\right) + I_2 & m_2 c_2^2 + I_2 \end{bmatrix} \tag{A.75}$$

$$\underline{C} = \begin{bmatrix} -m_2 l_1 c_2 \sin\left(\theta_2\right)\dot{\theta}_2 & -m_2 l_1 c_2 \dot{\theta}_2 \sin\left(\theta_2\right)\left(\dot{\theta}_1 + \dot{\theta}_2\right) \\ m_2 l_1 c_2 \sin\left(\theta_2\right)\dot{\theta}_1 & 0 \end{bmatrix} \tag{A.76}$$

$$\underline{G} = \begin{bmatrix} \left(m_1 c_1 + m_2 l_1\right) g \cos\left(\theta_1\right) + m_2 c_2 g \cos\left(\theta_1 + \theta_2\right) \\ m_2 c_2 g \cos\left(\theta_1 + \theta_2\right) \end{bmatrix} \tag{A.77}$$

A.2 Identification

The derived equations of motion from section 2.2.4 can now be used to find a regressor form for a dynamical identification process. For simplicity, we will again use the actuator dynamics to neglect certain terms in the equations of motion and we assume that the drive trains are stiff. Then, the system can be written like:

$$\tau_{M,i} = N_i \left(I_{M,i}\ddot{\theta}_i + B_{M,i}\dot{\theta}_i \right) + F_{c,i} sgn\left(\dot{\theta}_i\right) + \frac{1}{N_i}\left(\sum_{j=1}^{n} D_{ij}\ddot{\theta}_j + G_i \right) \tag{A.78}$$

where $\tau_{M,i}$ is the actuator torque, N_i is the gearbox ratio, $I_{M,i}$ is the actuator inertia and $B_{M,i}$ and $F_{c,i}$ are friction parameters, $i = 1, 2$:

$$\begin{aligned} \tau_{M,1} = {} & N_1 \left(I_{M,1}\ddot{\theta}_1 + B_{M,1}\dot{\theta}_1 \right) + F_{c,1} sgn\left(\dot{\theta}_1\right) \\ & + \frac{1}{N_1}\Big(\left(m_1 c_1^2 + m_2 \left(l_1^2 + c_2^2 + 2 l_1 c_2 C_2 \right) + I_1 + I_2 \right)\ddot{\theta}_1 \\ & + \left(m_2 \left(c_2^2 + l_1 c_2 C_2 \right) + I_2 \right)\ddot{\theta}_2 + \left(m_1 c_1 + m_2 l_1 \right) g C_1 + m_2 c_2 g C_{12} \Big) \end{aligned} \tag{A.79}$$

$$\begin{aligned} \tau_{M,2} = {} & N_2 \left(I_{M,2}\ddot{\theta}_2 + B_{M,2}\dot{\theta}_2 \right) + F_{c,2} sgn\left(\dot{\theta}_2\right) \\ & + \frac{1}{N_2}\Big(\left(m_2 \left(c_2^2 + l_1 c_2 C_2 \right) + I_2 \right)\ddot{\theta}_1 + \left(m_2 c_2^2 + I_2 \right)\ddot{\theta}_2 + m_2 c_2 g C_{12} \Big) \end{aligned} \tag{A.80}$$

where $C_1 = \cos(\theta_1)$, $C_2 = \cos(\theta_2)$ and $C_{12} = \cos(\theta_1 + \theta_2)$. This can be put in a regressor form for unknown parameters $B_{M,1}$, $F_{c,1}$, m_1, c_1, I_1, $B_{M,2}$, $F_{c,2}$, m_2, c_2 and I_2. After rearranging and substituting, the equations can be written as:

$$\underline{\zeta} = \underline{R}_0 \underline{\vartheta}_0 \tag{A.81}$$

where

$$\underline{\zeta} = \begin{bmatrix} \tau_{M,1} - I_{M,1} N_1 \ddot{\theta}_1 \\ \tau_{M,2} - I_{M,2} N_2 \ddot{\theta}_2 \end{bmatrix} \tag{A.82}$$

$$\underline{R}_0 = \begin{bmatrix} \frac{1}{N_1}\left(g C_{12} + 2 l_1 C_2 \ddot{\theta}_1 + l_1 C_2 \ddot{\theta}_2 \right) & \frac{1}{N_2}\left(g C_{12} + l_1 C_2 \ddot{\theta}_1 \right) \\ \frac{1}{N_1} g C_1 & 0 \\ \frac{l_1}{N_1}\left(g C_1 + \ddot{\theta}_1 l_1 \right) & 0 \\ \frac{1}{N_1}\ddot{\theta}_1 & 0 \\ \frac{1}{N_1}\left(\ddot{\theta}_1 + \ddot{\theta}_2 \right) & \frac{1}{N_2}\left(\ddot{\theta}_1 + \ddot{\theta}_2 \right) \\ N_1 \dot{\theta}_1 & 0 \\ sgn\left(\dot{\theta}_1\right) & 0 \\ 0 & N_2 \dot{\theta}_2 \\ 0 & sgn\left(\dot{\theta}_2\right) \end{bmatrix}^T \tag{A.83}$$

$$\underline{\vartheta}_0 = \begin{bmatrix} c_2 m_2 \\ c_1 m_1 \\ m_2 \\ c_1^2 m_1 + I_1 \\ c_2^2 m_2 + I_2 \\ B_{M,1} \\ F_{c,1} \\ B_{M,2} \\ F_{c,2} \end{bmatrix} \tag{A.84}$$

This is the regressor form for the double pendulum system. In $\underline{R}_0$ some terms are still dependent, which means we can use the algorithm of [1], to find the base regressor form. Every element $\underline{R}_{0,ij}$, $i = 1, 2$ and $j = 1 \cdots 9$, in $\underline{R}_0$ can be written as:

$$\underline{R}_{0,ij} = \underline{f}^T \underline{b}_{ij} \tag{A.85}$$

where

$$\underline{f}^T = \begin{bmatrix} C_1 C_2 & S_1 S_2 & C_2 \ddot{\theta}_1 & C_2 \ddot{\theta}_2 & C_1 & \ddot{\theta}_1 & \ddot{\theta}_2 & \dot{\theta}_1 & sign\left(\dot{\theta}_1\right) & \dot{\theta}_2 & sign\left(\dot{\theta}_2\right) \end{bmatrix} \tag{A.86}$$

and

$$\underline{B} = \begin{bmatrix} \underline{b}_{11} & \underline{b}_{12} & \underline{b}_{13} & \underline{b}_{14} & \underline{b}_{15} & \underline{b}_{16} & \underline{b}_{17} & \underline{b}_{18} & \underline{b}_{19} \\ \underline{b}_{21} & \underline{b}_{22} & \underline{b}_{23} & \underline{b}_{24} & \underline{b}_{25} & \underline{b}_{26} & \underline{b}_{27} & \underline{b}_{28} & \underline{b}_{29} \end{bmatrix} \tag{A.87}$$

$$= \begin{bmatrix} \frac{g}{N_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-g}{N_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{2l_1}{N_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{l_1}{N_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{g}{N_1} & \frac{gl_1}{N_1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{l_1^2}{N_1} & \frac{1}{N_1} & \frac{1}{N_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{N_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & N_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \frac{g}{N_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-g}{N_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{l_1}{N_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{N_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{N_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.88}$$

Herein, $\underline{f}$ is the vector of fundamental functions and $\underline{B}$ is the matrix with known constants. We can perform a Gauss-Jordan elimination to find the dependent columns of $\underline{R}_0$:

$$\underline{B}_E = \left[\begin{array}{ccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \frac{-1}{l_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{l_1^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c} \underline{B}_{Eu} \\ \hline \underline{0} \end{array}\right] \tag{A.89}$$

As can be seen from this elimination, the fourth column of $\underline{R}_0$ is dependent on the second and third column. This means we can remove column 4 from $\underline{R}_0$ to find the base regressor matrix $\underline{R}$ and base parameter set $\underline{\vartheta}$:

$$\underline{\zeta} = \underline{R}\underline{\vartheta} \tag{A.90}$$

where

$$\underline{\zeta} = \begin{bmatrix} \tau_{M,1} - I_{M,1}N_1\ddot{\theta}_1 \\ \tau_{M,2} - I_{M,2}N_2\ddot{\theta}_2 \end{bmatrix} \tag{A.91}$$

$$\underline{R} = \begin{bmatrix} \frac{1}{N_1}\left(gC_{12} + 2l_1C_2\ddot{\theta}_1 + l_1C_2\ddot{\theta}_2\right) & \frac{1}{N_2}\left(gC_{12} + l_1C_2\ddot{\theta}_1\right) \\ \frac{1}{N_1}gC_1 & 0 \\ \frac{l_1}{N_1}\left(gC_1 + \ddot{\theta}_1 l_1\right) & 0 \\ \frac{1}{N_1}\left(\ddot{\theta}_1 + \ddot{\theta}_2\right) & \frac{1}{N_2}\left(\ddot{\theta}_1 + \ddot{\theta}_2\right) \\ N_1\dot{\theta}_1 & 0 \\ sgn\left(\dot{\theta}_1\right) & 0 \\ 0 & N_2\dot{\theta}_2 \\ 0 & sgn\left(\dot{\theta}_2\right) \end{bmatrix}^T \tag{A.92}$$

$$\underline{\vartheta} = \underline{B}_{Eu}\underline{\vartheta}_0 = \begin{bmatrix} c_2 m_2 \\ m_1 c_1 - \frac{1}{l_1}\left(m_1 c_1^2 - I_1\right) \\ m_2 + \frac{1}{l_1^2}\left(m_1 c_1^2 + I_1\right) \\ c_2^2 m_2 + I_2 \\ B_{M,1} \\ F_{c,1} \\ B_{M,2} \\ F_{c,2} \end{bmatrix} \tag{A.93}$$

Appendix B

Regression algorithm

In this appendix, the algorithm that can find the regressor form automatically, is presented in Matlab code.

```
function [D t y] = regression(eom,ff,kp)

D = {};
t = {};
f = {};
B = {};

for r = 1:size(eom,1)
    D = [D; cell(1,size(D,2))];
    e = char(sort(expand(eom(r))));

    % add + operator to the first expression if not negative
    if  strcmp(e(1),'-')
        e = ['+' e];
    end

    % search for all + and - operators in the equation
    pmop = sort([strfind(e, '+') strfind(e, '-') length(e)+1]);

    % split expression in parameters and dynamics
    for i=1:length(pmop)-1
        % seperate each expression between + and - operators
        pms = e(pmop(i)+1:pmop(i+1)-1);

        % in each expression search for * and / operators
        mdop = sort([1 strfind(pms, '*') strfind(pms, '/') length(pms)+1]);

        p = '';
        d = '';
        a = '';
        for j=1:length(mdop)-1
            % seperate each variable between * and / operators
            mds = pms(mdop(j):mdop(j+1)-1);

            % test if variable is known or unknown and store in
            % corresponding vector
            isk = [];
            for k=1:length(kp)
                isk = [isk strfind(mds,kp{k})];
```

```
    end

    isf = [];
    for k=1:length(ff)
        isf = [isf strfind(mds,ff{k})];
    end

    if strcmp(mds(1),'*')
        mds = mds(2:end);
    elseif strcmp(mds(1), '/')
        mds = ['1' mds];
    end

    if isempty(isk) && isempty(isf) && isempty(str2num(mds))
        p = [p mds '*'];
    elseif isempty(isf)
        d = [d mds '*'];
    else
        a = [a mds '*'];
    end
end

% clean up string expressions to symbolize them
if isempty(p)
    p = '1';
else
    p = p(1:end-1);
end
if isempty(d)
    d = '1';
else
    d = d(1:end-1);
end
a = a(1:end-1);

n = find(strcmp(t,p),1);

% create matrices to find base parameter set
if isempty(n)
    t = [t;p];
    D{r,end+1} = [e(pmop(i)) d '*' a];
    n = size(D,2);
else
    D{r,n} = [D{r,n} e(pmop(i)) d '*' a];
end

m = find(strcmp(f,a),1);

if isempty(m)
    f = [f;a];
    B{length(f),n,r} = [e(pmop(i)) d];
else
    if n <= size(B,2) && r <= size(B,3)
        B{m,n,r} = [Bm,n,r e(pmop(i)) d];
    else
        B{m,n,r} = [e(pmop(i)) d];
    end
```

```
        end
    end
end

% symbolize all matrices
for i = 1:length(t)
    ts(i) = sym(t{i});
end

for i = 1:size(D,1)
    for j = 1:size(D,2)
        if  isempty(D{i,j})
            Ds(i,j) = sym(D{i,j});
        end
    end
end

Bt = [];
for i = 1:size(B,3)
    Bt = [Bt; B(:,:,i)];
end

n = find(ts==1, 1);
if  isempty(n)
    ts(n) = [];
    y = Ds(:,n);
    Ds(:,n) = [];
    Bt(:,n) = [];
else
    y = zeros(size(Ds,1),1);
end

for i = 1:size(Bt,1)
    for j = 1:size(Bt,2)
        if  isempty(Bt{i,j})
            Bs(i,j) = sym(Bt{i,j});
        end
    end
end

t = ts;
D = Ds;

% calculate base parameter set
Be = rref(Bs);
Bt = Be;

i = 1;
while i<=size(Bt,2)
    if  (Bt(i,i) == 1)
        Bt = [Bt(:,1:i-1) Bt(:,i+1:end)];
        D = [D(:,1:i-1) D(:,i+1:end)];
    else
        i = i + 1;
    end
end
```

```
B1 = Be(1:size(Bt,2),:);
t = B1*t.';
```