Adaptive control on bipedal humanoid robots

R. Beerens

DC 2014.013

Open-space report

Coach(es): ir. P.W.M. van Zutven

Supervisor: prof. dr. H. Nijmeijer

Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics & Control Group

Eindhoven, February, 2014

Abstract

In the past, many different control strategies have been developed in order to let a humanoid robot perform stable walking. Most of these strategies depend on accurate knowledge of the system parameters such as masses, moments of inertia and positions of centers of mass of each link of the robot. Numerical values of these parameters can be obtained by means of offline identification experiments which can be very time-consuming. Adaptive control techniques may offer a solution. Since the adaptive control mechanisms estimate important dynamic parameters, the controller performance may be improved.

Model Reference Adaptive Control (MRAC) theory is presented along with a modification so that parameter estimation can be done using non-persistently exciting reference trajectories. Also a proof of concept is presented. In order to use adaptive control theory, the robot's equations of motion must be rewritten in regressor form. There are several regression methods known to be applied to serial manipulators. When a humanoid robot stands with one foot on the ground, it can be considered as a serial manipulator with the other foot as its end effector. Due to the large number of degrees of freedom in humanoid robots (e.g. humanoid robot TUlip has 12 degrees of freedom), the equations of motion are complex, so that it is not possible to derive the regressor form using conventional methods. A regression method that is capable of deriving the regressor for robots with large equations of motion is presented and a MATLAB algorithm is created which derives the equations of motion based on the input of DH-parameters and then determines the regressor form.

The application of adaptive control on humanoid robots is not as straightforward as it is on serial manipulators. Since the humanoid robot has two feet, there are two sets of equations of motion: one with the left foot as its stance foot (base) and the right foot as end effector, and one the other way around. The parameter estimation thus needs to switch between two controllers. This may cause problems. A hybrid adaptive control mechanism is investigated, where the information about the adaptation is exchanged between two controllers. A simulation on a simple walk robot shows that this method may work on symmetric bipedal robots and robots with limited unknown parameters to be estimated. Also an adaptive controller based on the robot's dynamics with constraints is investigated. Here only one set of equations of motion is evaluated while the constraints (which do not enter in the parameter estimation laws) are switching. If the Lagrange multipliers can be measured (humanoid robots typically have multiple force sensors in their feet, so this assumption is justified), such an adaptive control mechanism may be useful. A simulation study supports these findings.

Preface

This report is the outcome of an open-space project carried out within the Dynamics and Control group at the Mechanical Engineering department of the Eindhoven University of Technology in the Netherlands.

I would like to thank my coach ir. Pieter van Zutven for his assistance and guidance during this project and my supervisor prof. dr. Henk Nijmeijer for his supervision and feedback. I also would like to thank the members of the Tech United humanoid team for showing me what it is like to be in a team developing a humanoid robot. I also would like to thank ing. Henk van Rooij for his assistance while working on the SE-rack computer systems.

Ruud Beerens, January 2014

Nomenclature

Symbols

γ	Scaling factor for parameter estimation
Γ	Positive definite matrix gain for parameter estimation
θ	Parameter set
Λ	Vector of Lagrange multipliers
C	Matrix containing centripetal and Coriolis terms
e_θ	Error between the true- and estimated parameter values
e_r	Error between the true position and position defined by a measure of tracking accuracy
e_q	Error between the true- and desired positions
E	Third order tensor for mapping the inertia parameters
g	Vector containing gravitational terms
G	Sum of the derivatives of the mass matrix w.r.t. joint angles
H	Matrix representing the generalized directions
${}^iI_i$	Moment of inertia tensor about the origin of DH-frame i
${}^iI_{Gi}$	Moment of inertia tensor about the center of mass of link i
$\bar{J}$	Vector of inertia parameters
J_v	Position DH Jacobian
J_ω	Orientation DH Jacobian
K_d	Positive definite matrix corresponding to derivative action in a controller
K_p	Positive definite matrix corresponding to static gain in a controller
L	Positive definite gain matrix
m	Link mass
M	Mass matrix
p_{iG_i}	Location of the center of mass of link i w.r.t. the origin of DH-coordinate frame i
P	Constraint Jacobian
q	Vector of generalized coordinates, i.e. positions
q_d	Vector of desired positions
q_r	Measure of tracking accuracy
Q	Third order tensor for mapping the locations of centers of mass
R	Rotation matrix
S	Skew-symmetric matrix

t	Time
T	Kinetic energy
u	Input torques vector / Controller input
U	Potential energy
v	Synthetic input control law
V	Lyapunov function
W	Regressor matrix
W_d	Regressor matrix based on desired positions
W_r	Regressor matrix based on the measure of tracking accuracy q_r
X	Regressor components of the first term of the Lagrange equations of motion
Y	Regressor components of the second term of the Lagrange equations of motion
Z	Regressor components of the third term of the Lagrange equations of motion
x, y, ϕ, ψ, θ	Coordinates

Abbreviations

DH	Denavit-Hartenberg
MRAC	Model Reference Adaptive Control

Notations

$\dot{\square}$, $\ddot{\square}$	First derivative, second derivative w.r.t. time
$\hat{\square}$	Estimate
$\square^\top$	Transpose of a vector or matrix

Contents

1 Introduction

As we can see in the world around us more and more human tasks, varying from simple to very complex, are ceded to robots and other automatic systems. Some clear examples are robotic systems in industrial processes and robotic systems that assist doctors in complex surgery. An active field of research involves robots that are able to assist in the health care sector. For example, if robots could take over some human tasks in the household, it is possible for older people to live independently till a higher age. The most obvious choice for such a robot is a bipedal humanoid robot since it can operate in a human environment. A humanoid robot has some key advantages compared to a robot on wheels with human-like arms. Where a wheeled robot can only operate on a flat surface, a humanoid robot is capable of operating on uneven surfaces, avoiding obstacles and even climb stairs.

Research in the field of humanoid robotics is currently very active. Researchers and engineers are already developing humanoid robots that show various human-like characteristics [1]. In the future, these robots should substitute people in a variety of tasks in industry, household, services, care, etc. Mimicking human-like walking behavior is very challenging since multiple degrees of freedom of the robot need to be controlled in a coordinated fashion to maintain the robot balanced during walking. At the Eindhoven University of Technology, this kind of research is done using the 1.34 [m] tall humanoid robot TUlip [2], see Figure 1.1. Each leg of TUlip has six degrees of freedom: three in the hip, one in the knee and two in the ankle, all driven by electric DC motors. Using a variety of sensors, the robot can operate fully autonomously. In the case of TUlip, twelve degrees of freedom must be controlled simultaneously.

In the past, many different control strategies have been developed in order to let a humanoid robot perform stable walking. The basic principles behind these strategies are different, but eventually they all rely on solving a nonlinear constrained optimization problem to find a suitable walking gait. These problems commonly depend on accurate knowledge of the system parameters, such as masses, moments of inertia and positions of the centers of mass of each link of the robot. For TUlip, offline identification experiments have been performed to identify these parameters using regressor methods [3, 4, 5]. This can be pretty time-consuming and moreover, changing the robot's hardware will highly affect the controller performance since dynamic parameters may be changed. Here, adaptive control technology may offer a solution. The idea is to create a closed-loop controller with parameters that can be updated to change the response of the system. The goal is for the parameters to converge to ideal values that cause the plant response to match the response of a certain reference model. The parameters to be updated concerned here can be the link masses, moments of inertia and locations of centers of mass of each link. Such a control mechanism is capable of handling changes in the robot's configuration. For example, when a link or motor is changed or when an extra battery is added to the robot's torso, dynamic parameters such as masses and locations of centers of mass necessary for the control laws, may be changed. The adaptive control mechanisms simply estimates the new parameters rather than perform a time-consuming static- or dynamic esti-

mation process. Adaptive control can thus contribute in the development of mimicking human-like walking behavior.

There is little literature available about online adaptive control of bipedal systems, although these robots have additional sensory information that may be beneficial, such as foot sensors, an inertial measurement unit and a vision systems. The goal of this project is to investigate the applicability of adaptive control mechanisms on bipedal humanoid robots. The results of this project will be beneficial for simulation, dynamical analysis, and adaptive control on humanoid robots and eventually help improvement of the balance control of humanoid robots such that, eventually, they will be capable of human-like walking in realistic environments, such homes, offices or hospitals.

In this report the first steps in research of the application of adaptive control on humanoid robots is described. The report is organised as follows. In Chapter 2, adaptive control theory is presented together with a modification which may be suitable to use on humanoid robots. In Chapter 3, some mathematical steps are described in rewriting the robot's equations of motion in regressor form so that it is suitable for the adaptive control mechanism. Then, in Chapter 4, the application of adaptive control to bipedal robots is discussed. In Chapter 5 the conclusions are discussed along with some recommendations.

Figure 1.1: Humanoid robot TUlip.

2 Adaptive Control Theory

In this chapter, the adaptive control architecture to be used is presented. Also a modification to this theory which can be useful for applying on humanoid robots is discussed, along with a proof-of-concept example. Adaptive control is defined as a control system where one or more parameters of the controller structure are modified to force the response of the resulting closed-loop system towards a desired one, despite uncertainties in the plant dynamics [6].

2.1 Model Reference Adaptive Control (MRAC)

Applying adaptive control to humanoid robots is not as straightforward as it is for serial robotic manipulators. The key difference between manipulators and bipedal humanoid robots is that the serial manipulator only consists of one open dynamic chain, while in the case of the bipedal robot there are two chains, since the robot has got two legs. When the humanoid robot stands with one foot on the ground, the robot can be considered as a robotic manipulator with several links and thus Model Reference Adaptive Control (MRAC) can be applied. Since the stance foot of the robot switches while walking, the control algorithm changes. The difficulties that are encountered with respect to this switching problem will be discussed later, here we introduce the theory behind MRAC

A Model Reference Adaptive Control Scheme consists of four parts: the system to be controlled (the plant), a reference model, a controller and an adaptation mechanism to estimate parameters, as depicted in Figure 2.1 [6].

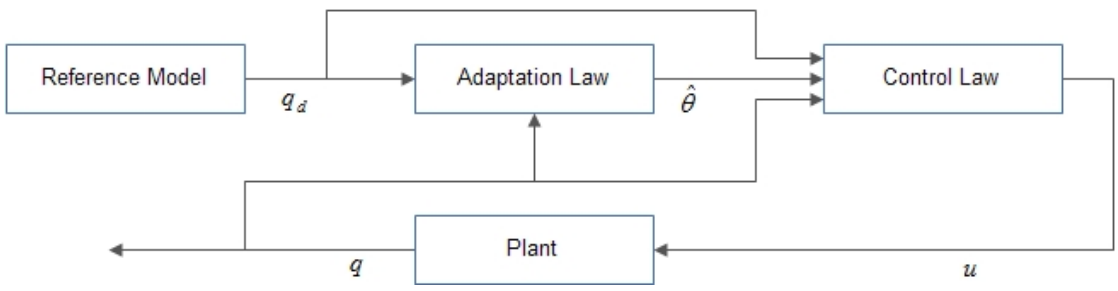

Figure 2.1: MRAC system. q_d is the desired position, q is the true position, u is the control input and $\hat{\theta}$ is the vector of estimated parameters.

Robot dynamics can be described by the equations of motion. The equations of motion of the robot are of the following form, where q is the vector of generalized coordinates, i.e. the joint angles:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = Hu, \tag{2.1}$$

where M is the mass matrix, C contains the centripetal and Coriolis terms, where terms containing $\dot{q}_i\dot{q}_j$ with $i = j$ belong to the centripetal terms and terms involving $\dot{q}_i\dot{q}_j$ with $i \neq j$ belong to the Coriolis terms. Moreover, g encompasses joint flexibility (which is not taken into account here) and gravitational effects. The matrix H represents the generalised directions of the forces or torques. The column u consists of the applied forces and torques.

The robot should follow a certain reference trajectory. Reference trajectories, based on the stability of the robot while walking using the Zero Moment Point, are amongst others presented in [7, 8]. By means of inverse kinematics these trajectories are translated to trajectories of all active joints. These joints are controlled by DC-motors and the controller should be able to minimize the position error, i.e. the error between the time-dependent angular reference position and the real angular position of the joint. Using radial encoders, the positions can be measured.

2.2 Control law proposition and error dynamics

In order to achieve asymptotic tracking of the reference signals, a feasible reference trajectory $q_d(t)$ for $t \geq 0$ is considered. The input $u(t)$ (the joint torques) must be computed such that the position errors should go to zero, that is

$$e_q(t) \to 0 \text{ for } t \to \infty;\ e_q = q - q_d. \tag{2.2}$$

A control law is designed such that $e_q(t) = 0$ is a globally asymptotically stable equilibrium point of the closed-loop error equation. The proposed control law is:

$$Hu = M(q)\,\ddot{q}_d + C(q,\dot{q})\,\dot{q}_d + g(q) - K_p e_q - K_d \dot{e}_q, \tag{2.3}$$

where $K_p = K_p^\top > 0$ and $K_d = K_d^\top > 0$. The choice of the control law is based on Lyapunov's second method [9], where only the stability is considered. The proposed control law results in the following error dynamics. Recall $e_q = q - q_d$, so that:

$$\begin{aligned} \ddot{e}_q &= \ddot{q} - \ddot{q}_d, \\ &= M^{-1}(q)\left[Hu - C(q,\dot{q})\,\dot{q} - g(q)\right] - \ddot{q}_d, \end{aligned} \tag{2.4}$$

which is realized by substituting (2.1). Then:

$$\begin{aligned} M(q)\,\ddot{e}_q &= Hu - C(q,\dot{q})\left[\dot{e}_q + \dot{q}_d\right] - g(q) - M(q)\,\ddot{q}_d, \\ &= M(q)\,\ddot{q}_d + C(q,\dot{q})\,\dot{q}_d + g(q) - K_p e_q - K_d \dot{e}_q - C(q,\dot{q})\,\dot{e}_q - C(q,\dot{q})\,\dot{q}_d \\ &\qquad \cdots - g(q) - M(q)\,\ddot{q}_d, \\ &= -K_p e_q - K_d \dot{e}_q - C(q,\dot{q})\,\dot{e}_q, \end{aligned} \tag{2.5}$$

which is realised by substituting the proposed control law (2.3). Rewriting (2.5) results in the following closed-loop error dynamics:

$$M(q)\,\ddot{e}_q + C(q,\dot{q})\,\dot{e}_q + K_d \dot{e}_q + K_p e_q = 0, \tag{2.6}$$

which should be asymptotically stable around $e_q(t) = 0$. This is proven in the next Section.

2.3 Stability proof

The control law used to control the joint positions should result in asymptotically stable tracking of the reference signals. To prove that this can be achieved, Lyapunov's second method is used. Define the system states to be the position (or tracking) error e_q and the error between the desired- and real velocities $\dot{e}_q$. A positive definite candidate Lyapunov function is the total energy associated with the closed loop:

$$V = \frac{1}{2}\dot{e}_q^\top M\left(q\right)\dot{e}_q + \frac{1}{2}e_q^\top K_p e_q, \tag{2.7}$$

where $K_p = K_p^\top > 0$. To prove stability, the time derivative of the Lyapunov function must be less than- or equal to zero. The time derivative of the Lyapunov function (2.7) is:

$$\dot{V} = \dot{e}_q^\top \left[M\left(q\right)\ddot{e}_q + \frac{1}{2}\dot{M}\left(q,\dot{q}\right)\dot{e}_q + K_p e_q\right]. \tag{2.8}$$

Note that, according to (2.5), the error dynamics can be rewritten as:

$$M\left(q\right)\ddot{e}_q = Hu - C\left(q,\dot{q}\right)\dot{q} - g\left(q\right) - M\left(q\right)\ddot{q}_d, \tag{2.9}$$

which is substituted into (2.8):

$$\dot{V} = \dot{e}_q^\top \left[Hu - M\left(q\right)\ddot{q}_d - C\left(q,\dot{q}\right)\dot{q} - g\left(q\right) + \frac{1}{2}\dot{M}\left(q,\dot{q}\right)e_q + K_p e_q\right]. \tag{2.10}$$

In the expression above, there exists a time derivative of the mass matrix: $\dot{M}$. This matrix must be removed from the equation since it is not necessarily positive definite and thus not desired in the stability analysis. If this is not done, (2.10) cannot be proven to be negative definite. The elimination of $\dot{M}$ is achieved as follows. The matrix consisting of the Coriolis and centripetal terms C, can be written as [6]:

$$C = \frac{1}{2}\dot{M} + \frac{1}{2}\left(G - G^\top\right), \tag{2.11}$$

where G is defined as the sum of the derivatives of the mass matrix with respect to the joint angles (i.e. the generalized coordinates) multiplied with the vector containing the joint velocities:

$$G = \Sigma_j \frac{\partial M}{\partial q_j}\dot{q}. \tag{2.12}$$

Any square matrix can be written as the sum of a symmetric- and skew symmetric part [10]:

$$A = \frac{1}{2}\left(A + A^\top\right) + \frac{1}{2}\left(A - A^\top\right) \tag{2.13}$$

By taking into account this property, the relation for C (2.11) implies that

$$\frac{1}{2}\dot{M} - C = -\frac{1}{2}\left(G - G^\top\right), \tag{2.14}$$

is skew-symmetric. By using this, a part of (2.10) can be rewritten:

$$
\begin{aligned}
\dot{e}_q^\top \left[\tfrac{1}{2}\dot{M}(q,\dot{q})\,\dot{e}_q - C(q,\dot{q})\,\dot{q}\right] &= \dot{e}_q^\top \left[\tfrac{1}{2}\dot{M}(q,\dot{q}) - C(q,\dot{q})\,\dot{e}_q - C(q,\dot{q})\,\dot{q}_d\right], \\
&= \tfrac{1}{2}\dot{e}_q^\top \left[\dot{M}(q,\dot{q}) - \dot{M}(q,\dot{q}) - \left(G - G^\top\right)\right]\dot{e}_q - \dot{e}_q^\top C(q,\dot{q})\,\dot{q}_d, \\
&= -\dot{e}_q^\top C(q,\dot{q})\,\dot{q}_d.
\end{aligned} \tag{2.15}
$$

This is applied to (2.10), so that the time derivative of the Lyapunov function (2.7) can be written as:

$$
\dot{V} = \dot{e}_q^\top \left[Hu - M(q)\,\ddot{q}_d - C(q,\dot{q})\,\dot{q}_d - g(q) + K_p e_q\right]. \tag{2.16}
$$

Stability is achieved if the time derivative of the Lyapunov function is less than- or equal to zero: $\dot{V} \leq 0$. To fulfill this, consider the proposed computed torque control law (2.3):

$$
Hu = M(q)\,\ddot{q}_d + C(q,\dot{q})\,\dot{q}_d + g(q) - K_p e_q - K_d \dot{e}_q, \tag{2.17}
$$

Then, substituting the above control law in the time-derivative of the Lyapunov function (2.16) results in:

$$
\dot{V} = -\dot{e}_q^\top K_d \dot{e}_q \leq 0, \tag{2.18}
$$

so that stability is proven. We now have:

$$
\begin{aligned}
V &= \tfrac{1}{2}\dot{e}_q^\top M(q)\,\dot{e}_q + \tfrac{1}{2}e_q^\top K_p e_q, \\
\dot{V} &= -\dot{e}_q^\top K_d \dot{e}_q,
\end{aligned} \tag{2.19}
$$

and thus $\dot{e}_q(t) \to 0$ for $t \to \infty$. The time derivative of the position error is thus proven to be stable, but this does not necessarily holds for the position error e_q. It remains to show that the system cannot get "stuck" at a position such that $\dot{V} = 0$ (i.e. $\dot{e}_q = 0$) if $q \neq q_d$. When $\dot{e}_q = 0$, it holds that (see (2.1) and (2.3)):

$$
\begin{aligned}
\ddot{q} \quad &= M(q)^{-1}\left[Hu - C(q,\dot{q})\,\dot{q} - g(q)\right], \\
&= M(q)^{-1}\left[M(q)\,\ddot{q}_d + C(q,\dot{q})\,\dot{q}_d + g(q) - K_p e_q - C(q,\dot{q})\,\dot{q} - g\right], \\
&= M(q)^{-1}(q)\left[M(q)\,\ddot{q}_d - C(q,\dot{q})\,\dot{e}_q - K_p e_q\right], \\
&= \ddot{q}_d - M(q)^{-1} K_p e_q, \\
\Rightarrow \quad \ddot{e}_q &= -M(q)^{-1} K_p e_q,
\end{aligned} \tag{2.20}
$$

which is nonzero for $e_q \neq 0$ if $K_p > 0$ and $M^{-1}K_p$ is regular. So if $e_q \neq 0$, then $\ddot{e}_q \neq 0$ and consequently $\dot{e}_q \neq 0$. The closed-loop is now proven to be asymptotically stable.

To summarize, recall that the closed loop error equation is:

$$
M(q)\,\ddot{e}_q + C(q,\dot{q})\,\dot{e}_q + K_d \dot{e}_q + K_p e_q = 0. \tag{2.21}
$$

We know that $\dot{e}_q(t) \to 0$ for $t \to \infty$ and we can prove for $t \to \infty$ that $\ddot{e}_q(t) \to 0$ and $e_q(t) \to 0$ if K_p is regular. So with the proposed control law, the equilibrium point $e_q(t) = 0$ is asymptotically stable.

There are however some restrictions on using this control law:

1. H must be invertible.
2. The number of inputs must be equal to the number of joints and each joint must have its own actuator.
3. No flexibility in the joints or links is allowed.
4. $M(q)$, $C(q, \dot{q})$, and $g(q)$ have to be computed online.

Condition 1, 2 and 4 are met and condition 3 is assumed to be met. Then, the above designed control law is applicable to the robot.

2.4 Handle unknown parameters: adaptation

The computed torque control law (2.3) cannot be used directly since some dynamic model parameters are unknown. The unknown parameters need to be estimated using an adaptation law. Let θ be the vector of unknown parameters and $\hat{\theta}$ the vector of the estimated parameters. The control law, including the estimated parameters, is now expressed as:

$$Hu = \hat{M}(q)\ddot{q}_d + \hat{C}(q, \dot{q})\dot{q}_d + \hat{g}(q) - K_d\dot{e}_q - K_p e_q. \tag{2.22}$$

The question that arises is if the closed loop is now still stable. The closed loop error equation can now be written as:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \hat{M}(q)\ddot{q}_d + \hat{C}(q, \dot{q})\dot{q}_d + \hat{g} - K_d\dot{e}_q - K_p e_q. \tag{2.23}$$

Taking into account that $q = e_q + q_d$, rewrite the error equation:

$$\begin{aligned} & M(q)\ddot{e}_q + M(q)\ddot{q}_d + C(q, \dot{q})\dot{e}_q + C(q, \dot{q})\dot{q}_d + g(q) \\ & \qquad \cdots = \hat{M}(q)\ddot{q}_d + \hat{C}(q, \dot{q})\dot{q}_d + \hat{g}(q) - K_d\dot{e}_q - K_p e_q. \\ \Rightarrow \quad & M(q)\ddot{e}_q + C(q, \dot{q})\dot{e}_q + K_d\dot{e}_q + K_p e_q \\ & \qquad \cdots = \left[\hat{M}(q) - M(q)\right]\ddot{q}_d + \left[\hat{C}(q, \dot{q}) - C(q, \dot{q})\right]\dot{q}_d + \hat{g}(q) - g(q). \end{aligned} \tag{2.24}$$

Next, the following important assumption is made:

It is assumed that the dynamic parameters θ [1] enter only linearly in the plant model, so that the plant can be written in the so-called Regressor form [10, 11]:

$$M(q, \theta)\ddot{q} + C(q, \dot{q}, \theta)\dot{q} + g(q, \theta) = W(q, \dot{q}, \ddot{q})\theta, \tag{2.25}$$

[1] i.e. masses, moments of inertia, the location of the center of mass of each link or a combination of them, i.e. base parameters [16].

where W is the Regressor matrix and θ is a vector containing the dynamic parameters to be estimated. The control law (2.22) can thus be rewritten as:

$$Hu = W_d\hat{\theta} - K_d\dot{e}_q - K_p e_q, \tag{2.26}$$

with $W_d = W\left(q, \dot{q}, \dot{q}_d, \ddot{q}_d\right)$. Now, the closed-loop error equation (2.24) is written in regressor form:

$$\begin{aligned} M\left(q\right)\ddot{e}_q + C\left(q,\dot{q}\right)\dot{e}_q + K_d\dot{e}_q + K_p e_q &= \hat{M}\left(q\right)\ddot{q}_d + \hat{C}\left(q,\dot{q}\right)\dot{q}_d + \hat{g}\left(q\right) - M\left(q\right)\ddot{q}_d \\ &\qquad \cdots - C\left(q,\dot{q}\right)\dot{q}_d - g\left(q\right) \\ &= W_d\hat{\theta} - W_d\theta \\ &= W_d\left(\hat{\theta} - \theta\right) \\ M\left(q\right)\ddot{e}_q + C\left(q,\dot{q}\right)\dot{e}_q + K_d\dot{e}_q + K_p e_q &= W_d e_\theta \end{aligned} \tag{2.27}$$

with $e_\theta = \hat{\theta} - \theta$; the difference between the estimated- and the actual numerical values of the unknown parameters of interest.

The goal is to adapt the parameters $\hat{\theta}$ so that $e_q(t) \to 0$ for $t \to \infty$ and that $e_q(t)$ is bounded for all $t \geq 0$. The adaptation mechanism is also based on Lyapunov's second method. Consider the candidate Lyapunov function:

$$V = \frac{1}{2}\dot{e}_q^\top M\left(q\right)\dot{e}_q + \frac{1}{2}e_q^\top K_p e_q + \frac{1}{2}e_\theta^\top \Gamma e_\theta,\ \Gamma = \Gamma^\top > 0. \tag{2.28}$$

The time derivative of this function is:

$$\begin{aligned} \dot{V} &= \dot{e}_q^\top\left[M\left(q\right)\ddot{e}_q + \tfrac{1}{2}\dot{M}\left(q,\dot{q}\right)\dot{e}_q + K_p e_q\right] + e_\theta^\top\Gamma\dot{e}_\theta, \\ &= \dot{e}_q^\top\left[\tfrac{1}{2}\dot{M}\left(q,\dot{q}\right) - C\left(q,\dot{q}\right)\right]\dot{e}_q - \dot{e}_q^\top K_d\dot{e}_q + e_\theta^\top\left[\Gamma\dot{e}_\theta + W_d^\top\dot{e}_q\right], \end{aligned} \tag{2.29}$$

where the first term of the right hand side is equal to zero (see equations (2.15)), the second term is smaller than- or equal to zero if $K_d > 0$ and thus in order for $\dot{V} \leq 0$ to hold, the third term is forced to zero. This is achieved using the following parameter update law:

$$\dot{e}_\theta = \dot{\hat{\theta}} - \dot{\theta} = \dot{\hat{\theta}} = -\Gamma^{-1}W_d^\top\dot{e}_q, \tag{2.30}$$

which holds if the parameters to be estimated are constant, i.e. $\dot{\theta} = 0$, which is assumed to be the case for dynamic parameters such as masses, moments of inertia and locations of centers of mass. The time derivative of the Lyapunov function (2.28) is then expressed as:

$$\dot{V} = -\dot{e}_q^\top K_d\dot{e}_q \leq 0 \text{ if } K_d > 0. \tag{2.31}$$

Based on Lyapunov's second method, it is concluded that an adaptive control mechanism of this form results in a stable system. However, it is only proven that $\dot{e}_q(t) \to 0$ for $t \to \infty$ and thus the position error $e_q(t)$ becomes constant, not necessarily zero. From the adaptation law (2.30) presented above, it follows that when $\dot{e}_q(t) \to 0$, $\dot{\hat{\theta}} \to 0$ and thus $\hat{\theta}(t) \to$ constant and from that it

follows that $e_\theta(t) \to$ constant and not necessarily zero. It must thus be proven in addition that this adaptive control mechanism results in asymptotical stability with respect to parameter convergence and position error.

Consider the error equation (2.27). It is just proven that when $t \to \infty$, $\dot{e}_q(t) \to 0$. From the error equation it then follows that $K_p e_q \to W_d e_\theta$, so in the limit the tracking error is coupled with the parameter error. Consider the regressor matrix containing the reference signals W_d, as introduced in (2.26). From the previous analysis it is concluded that $W_d e_\theta \to$ constant for $t \to \infty$ and $e_\theta(t) \to$ constant. Then:

$$\dot{W}_d e_\theta \to 0. \tag{2.32}$$

From this, the following can be concluded: if $\dot{W}_d(t)$ is *persistently exciting* [11], $\dot{W}_d$ cannot go to zero and thus $e_\theta(t) \to 0$. This is a sufficient condition. Then from the error equation and the above analysis it can thus be concluded that $e_q(t) \to 0$ if $\dot{W}_d$ is persistently exciting. It can be shown that if the reference signals are persistently exciting, $\dot{W}_d$ is persistently exciting [13] and consequently the dynamic parameters to be estimated will converge to their true values and the position error in the joint will converge asymptotically to zero.

Moreover, the parameter convergence can be accelerated by choosing Γ appropriately, e.g.:

$$\Gamma = \gamma \int_0^T W_d^\top W_d d\tau, \tag{2.33}$$

inspired by a least-squares-based adaptation, with γ a scaling factor and T a characteristic time (this can be the time of one forward step of the humanoid robot for example).

The plant model, reference model, control law and adaptation law can now be implemented in motion control software, according to the scheme presented in Figure 2.1.

2.5 Modification to guarantee asymptotic stability

In the previous section the basics of Model Reference Adaptive Control theory are presented. As discussed there, the stability of this method relies on the property that the reference trajectories are persistently exciting (i.e. $\dot{W}_d(t)$ is persistently exciting). Such trajectories can be generated relatively easy for serial robotic manipulators [12]. However when it comes to humanoid robots, this is a different story.

Since bipedal walking of humanoid robots is very challenging, the walking speeds are relatively low [21]. It can thus not be guaranteed that the reference signals are persistently exciting, resulting in unstable controllers and parameter adaptation. Therefore, a modification to the previous presented adaptive control theory must be made so that asymptotic convergence of the position error is guaranteed while the reference signals are not guaranteed to be persistently exciting. Such a method is proposed by Slotine & Li [14] and the idea is as follows:

- Replace $q_d(t)$ by another measure of tracking accuracy $q_r(t)$ (to be determined). Then: $\dot{e}_r(t) = \dot{q}(t) - \dot{q}_r(t)$ may converge to zero.
- Choose $q_r(t)$ so that $\dot{e}_r(t) = 0$ for all $t \geq t_n \geq 0$ implies $e_q(t) = q(t) - q_d(t) \to 0$ for $t \to \infty$, so require convergence to $e_q(t) = 0$.

If it is possible to find such a $q_r(t)$, then $e_q(t) \to 0$ for $t \to \infty$. The control law is then expressed as:

$$\begin{aligned} Hu &= \hat{M}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{g}(q) - K_d\dot{e}_r - K_p e_r, \\ &= W_r\hat{\theta} - K_d\dot{e}_r - K_p e_r, \end{aligned} \tag{2.34}$$

assuming the plant model is linear in its parameters. Furthermore, $W_r = W(q, \dot{q}, \dot{q}_r, \ddot{q}_r)$, is the regressor matrix depending on the joint positions and velocities, together with time derivatives of the new presented measure of tracking accuracy $q_r(t)$. Note that the stability analysis is still valid when $q_d(t)$ and $e_q(t)$ are replaced by $q_r(t)$ and $e_r(t)$ respectively, for proper choice of $q_r(t)$. A possible choice for $q_r(t)$ is [11]:

$$\dot{q}_r = \dot{q}_d - Le_q,\ L = L^\top > 0. \tag{2.35}$$

Then:

$$\dot{e}_r = \dot{e}_q + Le_q, \tag{2.36}$$

and consequently the new parameter update law is:

$$\dot{\hat{\theta}} = -\Gamma^{-1}W_r^\top \dot{e}_r. \tag{2.37}$$

Here, L, Γ, K_d and K_p can be tuned to best performance (all these matrices should however be positive). It can be shown that $\dot{e}_r(t) \to 0$, which results in $e_q(t) \to 0$ for $t \to \infty$ [11]. With this modification we choose an input $u(t)$ so that $\dot{e}_r(t) \to 0$ and do not try to reach $e_q(t) = q(t) - q_d(t) = 0$ directly. The key advantage of this method is that $e_q(t)$ will converge without the need for persistently exciting signals. However, with this method it is not guaranteed that the parameters converge to their true value, i.e. it is only guaranteed that $e_\theta(t) =$ constant. Therefore it can be concluded that adaptive control on humanoid robot might not be successful for parameter identification (i.e. for obtaining the correct values for the dynamic parameters).

2.6 Proof of concept

To check if the above modification can be used in humanoid robotics, a simulation is performed on a simple robot with three degrees of freedom as visualized in Figure 2.2. The geometry is kept simple for the sake of computation time. The robot is modeled using the DH-convention [10]. The equations of motion are derived and the unknown parameters are the masses of the three links (m_i), the moments of inertia ($I_{xx,i}$, $I_{yy,i}$, $I_{zz,i}$) and the locations of the centers of mass ($lc_{i,x}$, $lc_{i,y}$ and $lc_{i,z}$. From the equations of motion, the regressor matrix and vector of unknown base parameters [16] (this will be explained later) to be estimated are derived. This is done by using an algorithm created by van Zutven [4]. The parameter vector θ, where a_i is the length of link i and d_i the offset of link i corresponding to the DH-convention, is:

$$\theta = \begin{bmatrix} m_2 - (lc_{2,x}m_2)/a_2 + (lc_{3,x}m_3)/a_3 \\ (m_3(a_3 - lc_{3,x}))/a_3 \\ \left(lc_{2,x}m_2d_2^2\right)/a_2 + m_1lc_{1,z}^2 \\ -lc_{3,x}m_3(a_3 - lc_{3,x}) \\ -lc_{2,x}m_2(a_2 - lc_{2,x}) \end{bmatrix} \tag{2.38}$$

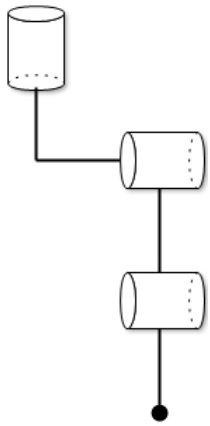

Figure 2.2: Three link robotic manipulator.

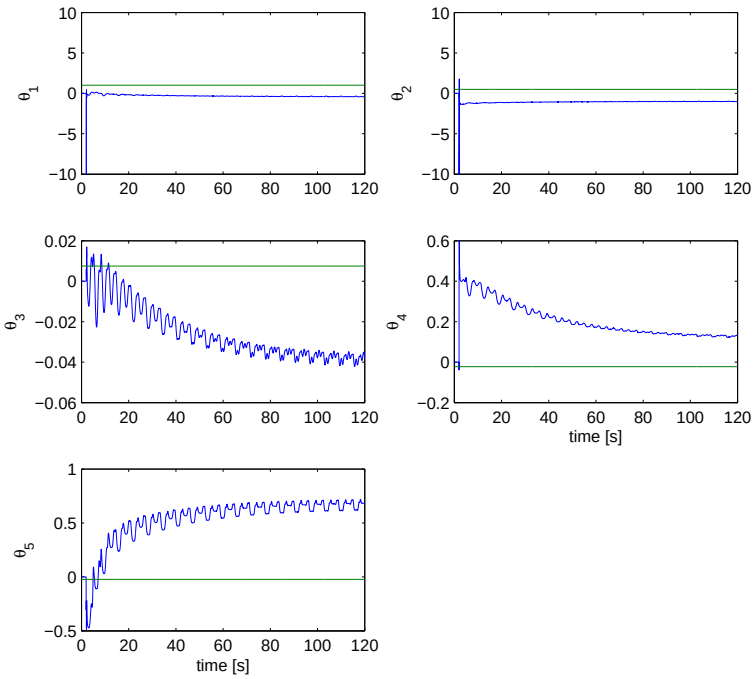

Figure 2.3: Parameter estimation of the 3-link robot leg using the standard adaptive control mechanism. Blue line: estimated parameter value. Green line: true parameter value.

A MATLAB/Simulink [23] model is created and run. First the standard MRAC controller is tested. Slowly varying reference signals are chosen (a second order reference model converts these signals in signals that can be tracked), resulting in relatively slow motions so that the trajectories might not be persistently exciting. Again, this might also hold for the trajectories of a true humanoid robot. In Figure 2.3 the parameter estimation is visualized. As seen, the parameters do converge, but not to their correct values (the green lines) and the estimation is oscillating. This is caused by the fact that the trajectories are not persistently exciting. This results in relatively large position errors of the links as seen in Figure 2.4. Moreover, convergence cannot be guaranteed and other problems such as parameter drift due to discretization may occur in practice.

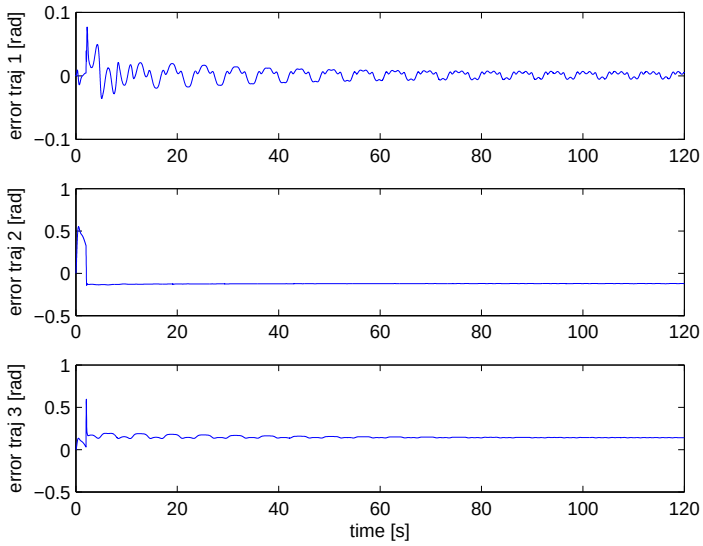

Figure 2.4: Joint position error of the 3-link robot leg using the standard adaptive control mechanism.

Next, the modified MRAC controller as discussed in this section is implemented in a MATLAB/Simulink model. Recap that in this case the reference signals are no longer required to be persistently exciting. Convergence of the parameters is guaranteed, but may not converge to their correct values. However, as long as convergence takes place, convergence of the joint position errors to zero is guaranteed. The parameter estimation process and position errors are shown in Figure 2.5 and 2.6. As can be seen, the parameters do converge, but not all to their correct value (the green line). However, this method guarantees tracking convergence if the parameters converge to a steady value, not necessarily the correct one. Since not all parameters converge to their true values, it can be concluded that the reference signals are not persistently exciting but the position errors still converge to zero. Therefore this method might be suitable to apply on humanoid robots.

2.7 Conclusions

In this Chapter an adaptive control law is presented. A modification is derived so that the reference trajectories of the robot's joints are not required to be persistently exciting. Due to this property, the proposed adaptive control architecture is suitable to use on a humanoid robot, where the motions of the links are relatively slow in general. A proof-of-concept simulation shows that the proposed controller works, as supported by the stability analysis. The application of this control architecture to bipedal robots will be discussed in the upcoming Chapters.

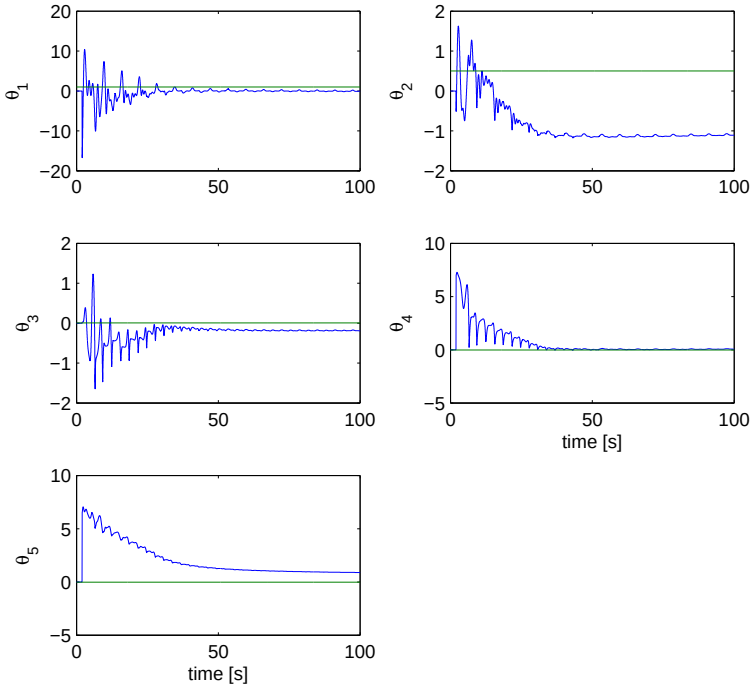

Figure 2.5: Parameter estimation of the 3-link robot leg using the modified adaptive control mechanism. Blue line: estimated parameter value. Green line: true parameter value.

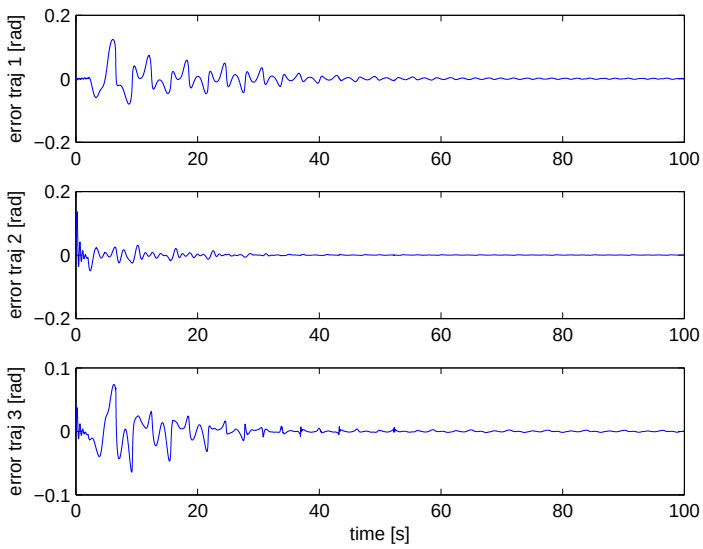

Figure 2.6: Joint position error of the 3-link robot leg using the modified adaptive control mechanism.

3 The regressor form

3.1 Regression

As discussed in the previous chapter, the equations of motion should be rewritten in regressor form before the adaptive control mechanism can be applied on the considered robot. Again, the equations of motion for robotic systems and the regressor form are expressed as

$$\begin{aligned} Hu &= M(q)\ddot{q} + C(q,\dot{q}) + g(q) \\ &= W\hat{\theta}, \end{aligned} \tag{3.1}$$

assuming the model is linear in its parameters. Furthermore, u contains the applied joint torques and the regressor matrix W contains functions of the measured joint angles and their time derivatives. The vector $\hat{\theta}$ contains the unknown parameters to be estimated. The above system can only be solved for $\hat{\theta}$ if the columns of W are linearly independent, which is rarely the case. Therefore, the standard parameter set should be transformed to a base parameter set [15]. The base parameters are the only identifiable parameters. Each base parameter consists of a combination of inertial parameters grouped together. In that case, the columns of the regressor matrix are linearly independent and thus the parameters can be identified.

There are several methods available for rewriting the equations of motion in regressor form and determining the base parameter set. First, there is the symbolic method developed by Khalil et al. [16], which is able to determine the base parameters for both series and some types of robots with closed kinematic chains, based on simple rules. Then, there is the symbolic method developed by Kawasaki et al. [17]. This method is capable of deriving the base parameters and rewrites the equations in the regressor form (3.1). This algorithm was also used by Van Zutven [4], who implemented the algorithm in a MATLAB function. Since this algorithm is relatively easy to use, this is used for trying to derive the regressor form of the humanoid robot TUlip.

3.2 Difficulties in deriving the regressor form for humanoid robots

Since the adaptive controller needs to be implemented in the control of humanoid robot TUlip, the regressor form of the equations of motion must be derived. It is important to note that there are two configurations possible: when the left foot is on the ground, the right foot is considered the end effector so that an open kinematic chain exists. The second configuration is the other way around. This means that in this case there are two sets of equations of motion.

Consider the right foot as stance foot and the left foot as end effector. First, the robot is modeled and coordinate frames are assigned to the links using the Denavit-Hartenberg convention [10]. Then, by deriving the Jacobian matrices and the potential energy of the robot, the mass matrix,

Christoffel symbols, Coriolis- and centripetal effects matrix and the matrix containing the gravitational forces are determined. A detailed description of this process can be found in [10].

Now that the equations of motion are determined, the regressor matrix and base parameter vector must be obtained. The MATLAB algorithm by Van Zutven et al. is used for this purpose. The algorithm expands the symbolic obtained equations from MATLAB and rewrites it to character format. Therefore, this algorithm needs a lot of computer memory to store the equations, let alone the CPU power to compute the regressor. Deriving the regressor form at a personal computer was unsuccessful, since the equations of motion are very large. Several potential solutions have been tried in order to get the regressor form using this algorithm:

- Assuming that the links are point masses to reduce the equations (this assumption is always made from now on).
- Substituting the sine- and cosine terms with variables to reduce memory usage. However, the substitution already took an unacceptable long time.
- Perform the calculations on an SE-rack computer [18] with 32GB of RAM memory. However, this amount of memory was also insufficient.

After many attempts, it is concluded that the regressor form of the equations of motion of the humanoid robot TUlip cannot be obtained by using this MATLAB algorithm. A promising method however, is presented in [19] and will be discussed in the next section.

3.3 Direct formulation of the regressor

A more promising method for calculating the regressor form is presented by Gabiccini et al. [19]. Based on the kinetic- and potential energy, the Lagrange equations of motion are derived per link and each term of these equations are then written linear in its dynamic parameters. The method in this paper contains however some errors with respect to matrix dimensions. Those errors are resolved and the correct method is presented here.

The Lagrange equations of motion *per link* are:

$$\left[\frac{d}{dt}\frac{\partial T^i}{\partial \dot{q}} - \frac{\partial T^i}{\partial q} + \frac{\partial U^i}{\partial q}\right]^\top \equiv W^i \theta^i \tag{3.2}$$

where T^i is the kinetic energy of link i and U is the potential energy of link i. The kinetic energy of link i can be written as:

$$\begin{aligned} T^i \quad &= \tfrac{1}{2} m_i \dot{q}^\top \left(J_{vi}^\top J_{vi}\right) \dot{q} - \tfrac{1}{2} m_i \dot{q}^\top \{J_{vi}^\top S\left(R_i^0 p_{iG_i}\right) J_{\omega i}\} \dot{q} \\ &+ \tfrac{1}{2} m_i \dot{q}^\top \{J_{\omega i} S\left(R_i^0 p_{iG_i}\right) J_{vi}\} \dot{q} \\ &+ \tfrac{1}{2} \dot{q}^\top \{J_{\omega i}^\top R_i^0 \left[I_{Gi}^i + m_i S^\top\left(p_{iG_i}\right) S\left(p_{iG_i}\right)\right] R_i^{0^\top} J_{\omega i}\} \dot{q}, \end{aligned} \tag{3.3}$$

where J_{vi} and $J_{\omega i}$ are the position and orientation DH Jacobians, R_i^0 the rotation matrix from the base frame to the ith frame, m_i the mass of link i, p_{iG_i} location of the center of mass of link i with respect to the origin of DH-coordinate frame i and I_{Gi}^i the moment of inertia tensor about the center of mass of link i. Furthermore, S is a skew-symmetric matrix such that $S\left(x\right) y = S^\top\left(y\right) x = x \times y$

$\forall\, x, y \in \mathbb{R}^3$.

According to Steiner's Theorem [20], $I_i^i =^i I_{Gi} + m_i S^\top (p_{iG_i})\, S(p_{iG_i})$ is substituted. Then, the derivative with respect to $\dot{q}$ is taken and after some rearrangement:

$$\begin{aligned}\left[\frac{\partial T^i}{\partial \dot{q}}\right]^\top &= \left(J_{vi}^\top J_{vi}\right)\dot{q} m_i + \{J_{vi}^\top S\left(J_{\omega i}\dot{q}\right) R_i^0 - J_{\omega i}^\top S\left(J_{vi}\dot{q}\right) R_i^0\} m_i p_{iG_i} \\ &+ J_{\omega i}^\top R_i^0 I_i^i R_i^{0^\top} J_{\omega i}\dot{q}.\end{aligned} \tag{3.4}$$

A clear structure becomes visible now. The first- and second term of the above equation are obtained by extracting the mass and the first order moment of inertia $m_i p_{iG_i}$. The elements of the inertia matrix I_i^i have to be extracted from the third term. For such purpose a third order tensor $E \in \mathbb{R}^{3\times3\times6}$ is introduced and the vector of parameters $\bar{J}_i = \begin{bmatrix}\bar{J}_{ixx} & \bar{J}_{ixy} & \bar{J}_{ixz} & \bar{J}_{iyy} & \bar{J}_{iyz} & \bar{J}_{izz}\end{bmatrix}^\top$, where

$$E = \begin{bmatrix}E_1 & E_2 & E_3 & E_4 & E_5 & E_6\end{bmatrix}, \qquad I_i^i = \begin{bmatrix}\bar{J}_{ixx} & \bar{J}_{ixy} & \bar{J}_{ixz} \\ \bar{J}_{ixy} & \bar{J}_{iyy} & \bar{J}_{iyz} \\ \bar{J}_{ixz} & \bar{J}_{iyz} & \bar{J}_{izz}\end{bmatrix}, \tag{3.5}$$

with

$$\begin{aligned} E_1 &= \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix}, \quad E_2 = \begin{bmatrix}0&1&0\\1&0&0\\0&0&0\end{bmatrix}, \quad E_3 = \begin{bmatrix}0&0&1\\0&0&0\\1&0&0\end{bmatrix}, \\ E_4 &= \begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}, \quad E_5 = \begin{bmatrix}0&0&0\\0&0&1\\0&1&0\end{bmatrix}, \quad E_6 = \begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix}. \end{aligned} \tag{3.6}$$

In this way we can write I_i^i as the inner product of E and $\bar{J}_i$: $I_i^i = E \cdot \bar{J}_i$. Then, the third term of equation (3.4) becomes:

$$J_{\omega i}^\top R_i^0 I_i^i R_i^{0^\top} J_{\omega i}\dot{q} = \left[J_{\omega i}^\top R_i^0 E_1 R_i^{0^\top} J_{\omega i}\dot{q} | \cdots | J_{\omega i}^\top R_i^0 E_6 R_i^{0^\top} J_{\omega i}\dot{q}\right] \bar{J}_i \tag{3.7}$$

We are now able to extract the mass m_i, the first moment of inertia $m_i p_{iG_i}$ and the components of the inertia tensor. Thus we get:

$$\frac{d}{dt}\left[\frac{\partial T^i}{\partial \dot{q}}\right]^\top = \dot{X}_0^i \theta_0^i + \dot{X}_1^i \theta_1^i + \dot{X}_2^i \theta_2^i, \tag{3.8}$$

where

$$
\begin{aligned}
X_0^i &= \left(J_{vi}^\top J_{vi}\right)\dot{q} \in \mathbb{R}^{n\times 1}, \\
X_1^i &= \{J_{vi}^\top S\left(J_{\omega i}\dot{q}\right) - J_{\omega i}^\top S\left(J_{vi}\dot{q}\right) R_i^0 \in \mathbb{R}^{n\times 3}, \\
X_2^i &= J_{\omega i}^\top R_i^0 \left[E_1|E_2|\cdots|E_6\right] R_i^{0^\top} J_{\omega i}\dot{q} \in \mathbb{R}^{n\times 6}, \\
\theta_0^i &= m_i \in \mathbb{R}, \\
\theta_1^i &= \begin{bmatrix} m_i p_{iG_{i,x}} & m_i p_{iG_{i,y}} & m_i p_{iG_{i,z}} \end{bmatrix}^\top \in \mathbb{R}^3, \\
\theta_2^i &= \bar{J}_i \in \mathbb{R}^6.
\end{aligned}
\tag{3.9}
$$

In a similar way, the derivative of the kinetic energy with respect to q is taken (the second term of the Lagrange equation (3.2)) and rewritten as:

$$
\left[\frac{\partial T^i}{\partial q}\right]^\top = Y_0^i\theta_0^i + Y_1^i\theta_1^i + Y_2^i\theta_2^i, \tag{3.10}
$$

where

$$
\begin{aligned}
Y_0^i &= \begin{bmatrix} \frac{1}{2}\dot{q}^\top \frac{\partial}{\partial q_1}\left(J_{vi}^\top J_{vi}\right)\dot{q} \\ \vdots \\ \frac{1}{2}\dot{q}^\top \frac{\partial}{\partial q_n}\left(J_{vi}^\top J_{vi}\right)\dot{q} \end{bmatrix}, \\
Y_1^i &= \begin{bmatrix} \frac{1}{2}\frac{\partial}{\partial q_1}\left[R_i^{0^\top} S^\top\left(J_{\omega i}\dot{q}\right) J_{vi}\dot{q} - R_i^{0^\top} S^\top\left(J_{vi}\dot{q}\right) J_{\omega i}\dot{q}\right]^\top \\ \vdots \\ \frac{1}{2}\frac{\partial}{\partial q_n}\left[R_i^{0^\top} S^\top\left(J_{\omega i}\dot{q}\right) J_{vi}\dot{q} - R_i^{0^\top} S^\top\left(J_{vi}\dot{q}\right) J_{\omega i}\dot{q}\right]^\top \end{bmatrix}, \\
Y_2^i &= \begin{bmatrix} \frac{1}{2}\dot{q}^\top \frac{\partial}{\partial q_1}\left(J_{\omega i}^\top R_i^0 E R_i^{0^\top} J_{\omega i}\right)\dot{q} \\ \vdots \\ \frac{1}{2}\dot{q}^\top \frac{\partial}{\partial q_n}\left(J_{\omega i}^\top R_i^0 E R_i^{0^\top} J_{\omega i}\right)\dot{q} \end{bmatrix}.
\end{aligned}
\tag{3.11}
$$

The last term of the Lagrange equations comes from the potential energy, which can be written for link i as:

$$
U^i = -m_i g^\top \left(p_i^0 + R_i^0 p_{iG_i}\right) \tag{3.12}
$$

where g is the gravitational acceleration vector with respect to the base frame and p_i^0 is the position vector from the base to the ith DH-frame. Differentiating with respect to q and rearranging the terms, so that the mass m_i and the first moment of inertia $m_i p_{iG_i}$ are isolated, we get:

$$
\left[\frac{\partial U^i}{\partial q}\right]^\top = Z_0^i m_i + Z_1^i m_i p_{iG_i}, \tag{3.13}
$$

with

$$Z_0^i = -J_{vi}^\top g, \qquad Z_1^i = -\begin{bmatrix} \left(\frac{\partial\left(R_i^{0\top} g\right)}{\partial q_1}\right)^\top \\ \vdots \\ \left(\frac{\partial\left(R_i^{0\top} g\right)}{\partial q_1}\right)^\top \end{bmatrix}. \tag{3.14}$$

The regressor and parameter vector for link i can now be written as:

$$W^i\theta^i = \begin{bmatrix} W_0^i & W_1^i & W_2^i \end{bmatrix} \begin{bmatrix} \theta_0^i \\ \theta_1^i \\ \theta_2^i \end{bmatrix}, \tag{3.15}$$

with

$$\begin{aligned} W_0^i &= \dot{X}_0^i - Y_0^i + Z_0^i \in \mathbb{R}^{n\times 1}, \\ W_1^i &= \dot{X}_1^i - Y_1^i + Z_1^i \in \mathbb{R}^{n\times 3}, \\ W_2^i &= \dot{X}_2^i - Y_2^i \in \mathbb{R}^{n\times 6}. \end{aligned} \tag{3.16}$$

Then, the regressor matrix and parameter vector of the entire robot can be written as:

$$W\left(q, \dot{q}, \ddot{q}\right) = \begin{bmatrix} W^1 & \cdots & W^n \end{bmatrix}, \qquad \theta = \begin{bmatrix} \theta^{1\top} & \cdots & \theta^{n\top} \end{bmatrix}. \tag{3.17}$$

The above presented method is implemented in a MATLAB algorithm, which determines the robot's equations of motion based on the DH-convention. It then calculates the regressor and parameter vector. The algorithm is presented in Appendix A. Using this algorithm, the regressor of humanoid robot TUlip is determined from the equations of motion within the acceptable computing time of approximately 45 minutes (on an i7 3610QM CPU).

3.4 The Slotine-Li regressor

Since the adaptive control mechanism to be used is the method introduced by Slotine and Li, see Section 3.4, the above regression method must be modified so that it is compatible with the Slotine and Li version of the regressor. Gabiccini et. al [19] have introduced such a modification, however, also in this part of the paper there are errors and unclear sections. Also here, the errors are successful resolved and the correct method is presented below.

Following the theory presented in Gabiccini et al. and considering the equations of motion (2.1) and the MRAC theory from Chapter 2, the robot regressor defined as $\hat{M}\left(q\right)\ddot{q}_d + \hat{C}\left(q, \dot{q}\right)\dot{q}_d + g\left(q\right) = W_d\hat{\theta}$, can be *partially* expressed as:

$$\hat{M}\left(q\right)\ddot{q}_d + \hat{C}\left(q, \dot{q}\right)\dot{q}_d = X - Y, \tag{3.18}$$

and let X_h and Y_h be the h-th components of vectors X and Y, where

$$
\begin{aligned}
X_h &= \sum_{j=1}^{n} m_{hj}\ddot{q}_{d_j} + \sum_{j=1}^{n}\sum_{k=1}^{n} \tfrac{1}{2}\left(\frac{\partial m_{hj}}{\partial q_k} + \frac{\partial m_{hk}}{\partial q_j}\right)\dot{q}_k\dot{q}_{d_j}, \\
Y_h &= \tfrac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial m_{jk}}{\partial q_h}\dot{q}_k\dot{q}_{r_j},
\end{aligned}
\tag{3.19}
$$

where m are the elements of the mass matrix M. Expressions for the above equations are determined so that they are expressed linear in the dynamic parameters. Due to the fact that M is linkwise additive it follows that X and Y are as well, and after systematic calculations, we get:

$$
X^i = \dot{X}^i_{0d}\theta^i_0 + \dot{X}^i_{1d}\theta^i_1 + \dot{X}^i_{2d}\theta^i_2, \tag{3.20}
$$

with θ^i_j as in (3.9) and where

$$
\begin{aligned}
\dot{X}^i_{0d} &= \left(J^\top_{vi}J_{vi}\right)\ddot{q}_d + \tfrac{1}{2}\left\{\frac{\partial\left(J^\top_{vi}J_{vi}\right)}{\partial q} + \left[\frac{\partial\left(J^\top_{vi}J_{vi}\right)}{\partial q}\right]^{\top_{1,3,2}}\right\}\dot{q}\dot{q}_d, \\
\dot{X}^i_{1d} &= X^i_1\ddot{q}_r + \tfrac{1}{2}\left\{\frac{\partial X^i_1}{\partial q} + \left[\frac{\partial X^i_1}{\partial q}\right]^{\top_{1,3,2,4}}\right\}\dot{q}\dot{q}_d, \\
\dot{X}^i_{2d} &= X^i_2\ddot{q}_r + \tfrac{1}{2}\left\{\frac{\partial X^i_2}{\partial q} + \left[\frac{\partial X^i_2}{\partial q}\right]^{\top_{1,3,2,4}}\right\}\dot{q}\dot{q}_d,
\end{aligned}
\tag{3.21}
$$

with

$$
\begin{aligned}
X^i_1 &= J^\top_{\omega i}R^0_i\left[Q_1|Q_2|Q_3\right]R^{0^\top}_i J_{vi} - J^\top_{vi}R^0_i\left[Q_1|Q_2|Q_3\right]R^{0^\top}_i J_{\omega i} \in \mathbb{R}^{n\times n\times 3}, \\
X^i_2 &= J^\top_{\omega i}R^0_i\left[E_1|\cdots|E_6\right]R^{0^\top}_i J_{\omega i} \in \mathbb{R}^{n\times n\times 6}.
\end{aligned}
\tag{3.22}
$$

Similar to (3.6), a third order tensor $Q \in \mathbb{R}^{3\times3\times3}$ is used to reshape $S\left(p_{iG_i}\right)$, getting $S\left(p_{iG_i}\right) = Q \cdot p_{iG_i}$ where

$$
\begin{gathered}
Q = \begin{bmatrix} Q_1 & Q_2 & Q_3 \end{bmatrix}, \\
Q_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad Q_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.
\end{gathered}
\tag{3.23}
$$

In (3.22), a generalized transpose operator is introduced, working on objects of three- and four dimensions (tensors). This operator switches the dimensions according to the subscript of the transpose operator, e.g. $\left(a_{hjk}\right)^{\top_{1,3,2}} = a_{hkj}$. The tensors appearing in (3.22) are:

$$
\frac{\partial\left(J^\top_{vi}J_{vi}\right)}{\partial q} \in \mathbb{R}^{n\times n\times n}, \qquad \frac{\partial X^i_1}{\partial q} \in \mathbb{R}^{n\times n\times n\times 3}, \qquad \frac{\partial X^i_2}{\partial q} \in \mathbb{R}^{n\times n\times n\times 6}. \tag{3.24}
$$

For Y holds that

$$Y^i = Y^i_{0d}\theta^i_0 + Y^i_{1d}\theta^i_1 + Y^i_{2d}\theta^i_2, \tag{3.25}$$

where, very similar to (3.11):

$$\begin{aligned}
Y^i_{0d} &= \begin{bmatrix} \frac{1}{2}\dot{q}_d^\top \frac{\partial}{\partial q_1}\left(J_{vi}^\top J_{vi}\right)\dot{q} \\ \vdots \\ \frac{1}{2}\dot{q}_d^\top \frac{\partial}{\partial q_n}\left(J_{vi}^\top J_{vi}\right)\dot{q} \end{bmatrix}, \\
Y^i_{1d} &= \begin{bmatrix} \frac{1}{2}\frac{\partial}{\partial q_1}\left[R_i^{0^\top} S^\top\left(J_{\omega i}\dot{q}\right) J_{vi}\dot{q}_d - R_i^{0^\top} S^\top\left(J_{vi}\dot{q}\right) J_{\omega i}\dot{q}_d\right]^\top \\ \vdots \\ \frac{1}{2}\frac{\partial}{\partial q_n}\left[R_i^{0^\top} S^\top\left(J_{\omega i}\dot{q}\right) J_{vi}\dot{q}_d - R_i^{0^\top} S^\top\left(J_{vi}\dot{q}\right) J_{\omega i}\dot{q}_d\right]^\top \end{bmatrix}, \\
Y^i_{2d} &= \begin{bmatrix} \frac{1}{2}\dot{q}_d^\top \frac{\partial}{\partial q_1}\left(J_{\omega i}^\top R_i^0 E R_i^{0^\top} J_{\omega i}\right)\dot{q} \\ \vdots \\ \frac{1}{2}\dot{q}_d^\top \frac{\partial}{\partial q_n}\left(J_{\omega i}^\top R_i^0 E R_i^{0^\top} J_{\omega i}\right)\dot{q} \end{bmatrix}.
\end{aligned} \tag{3.26}$$

The gravitational terms can be written as presented in the previous section, since $g(q)$ is not a function of the reference velocity $\dot{q}_d$. The regressor can now be written according to (3.15), where now

$$\begin{aligned}
W^i_0 &= \dot{X}^i_{0d} - Y^i_{0d} + Z^i_0 \in \mathbb{R}^{n\times 1}, \\
W^i_1 &= \dot{X}^i_{1d} - Y^i_{1d} + Z^i_1 \in \mathbb{R}^{n\times 3}, \\
W^i_2 &= \dot{X}^i_{2d} - Y^i_{2d} \in \mathbb{R}^{n\times 6}.
\end{aligned} \tag{3.27}$$

Then, the Slotine & Li-regressor matrix and parameter vector of the entire robot can be written as:

$$W\left(q, \dot{q}, \dot{q}_d, \ddot{q}_d\right) = \begin{bmatrix} W^1 & \cdots & W^n \end{bmatrix}, \qquad \theta = \begin{bmatrix} \theta^{1^\top} & \cdots & \theta^{n^\top} \end{bmatrix}. \tag{3.28}$$

The above presented regression method is implemented in MATLAB and presented in Appendix B. With this algorithm, the regressor matrix and parameter vector, necessary for the adaptive control mechanism presented in Chapter 2, is determined for humanoid robot TUlip. In the code in the Appendix is the part where the equations of motion are determined left out, since this is similar to what is presented in Appendix A.

Now that there is an efficient method for regression for robots with a high number of degrees of freedom and thus large equations of motion, the adaptive control mechanism presented in Chapter 2 can be applied.

4 Application to humanoid robots

4.1 Difficulties in application to humanoid robots

In the previous sections the adaptive control theory is presented, along with an efficient method to derive the Slotine & Li-regressor for robots with large equations of motion. The application to bipedal humanoid robots, however, is not as straightforward as it is for serial robotic manipulators. As stated before, a bipedal robot can be considered as a serial manipulator as one foot stands on the ground: this foot can then be considered the base of the robot. The base switches to the other foot when the robot takes a step with the other leg. It is this switching behavior that introduces difficulties so that classic adaptive control might not be applicable.

When the humanoid robot takes a step, the estimation of unknown parameters is started. Then, the robot takes a step with his other leg, which means that there is another set of equations of motion, with the other foot as its base, that is evaluated. However, the adaptation must be continued, notwithstanding a different set of equations of motion is used. This can lead to problems, since information about the adaptation must be exchanged between the two sets of equations of motion. It is not per definition guaranteed that this will lead to a stable control system.

Two possible solutions are discussed next. The first one is based on a "hybrid" adaptive control mechanism, where two sets of equations of motion are evaluated and information about the parameters to be estimated is exchanged between the two sets. The second one is based on a dynamic model with constraints, so that only one set of equations of motion is evaluated and the constraints are switching.

The methods below are illustrated by means of a two-link symmetric walking robot in two dimensions, as visualized in Figure 4.1. This robot is very simple, but is ideal to show the concept since it got two "legs", and thus switching must occur in the adaptive control mechanism. For the sake of simplicity, the links of the robot are considered point masses. This is often done for humanoid robots to keep the equations of motion simpler. Since humanoid robots move relatively slow, this approximation is fair.

4.2 Hybrid adaptive control

A possible solution to the switching problem might be an adaptive controller with a hybrid mechanism, i.e. two controllers, each for every set of equations of motion, where the information about the adaptation is exchanged.

First, a mathematical model of the robot is determined. This model can be found in Appendix C. As stated before, the links are considered point masses for simplicity. Using the regression algo-

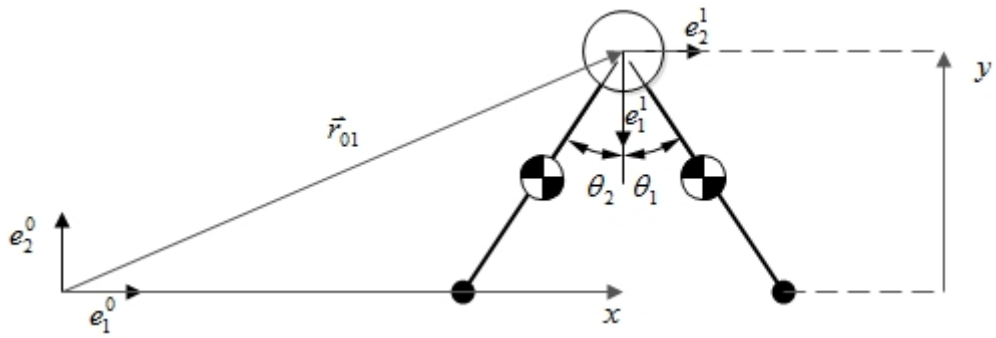

Figure 4.1: Two-link biped robot.

rithm by van Zutven [4], the regressor and base parameter vector are determined and implemented in an adaptive control scheme in Simulink (see Figure 2.1 and Section 3.4). The Simulink model can be run for both a step with the left foot and a step with the right foot, depending on the conditions provided to the model. The time it takes to perform one step is taken 1 [s].

The parameter estimation begins at the first step and ends when the first step finishes. The numerical values for the parameters obtained in this step are used as an initial condition for the estimation during the second step, and thus the other model. Also information about the position and velocity are exchanged and by means of a coordinate transformation the initial position of the robot at step $k+1$ is exactly the final position of the robot at step k. Since the parameter update is in this way a process without 'jumps' in the estimation, this type of controller may be stable.

First, the robot is considered symmetric with the masses of both links equal. This results in one unknown parameter to be estimated: the mass of both links. The simulation is run and the parameter estimation process and position error are presented in Figure 4.2. As seen, the controlled system is stable and the position error converges. The robot is modeled as two serial manipulators, with both one foot fixed to the ground (i.e. the base of the manipulator). The adaptive controller is applied on these manipulators. As described before, during the switching of controllers and models to be evaluated, information about position, velocity and estimated parameter values are exchanged so that the robot's configuration at the end of step k is exactly the configuration at the start of step $k+1$. The parameter update continues since the initial condition of the estimation process is the last known value obtained in step k. In this way, a continuous process is realised. This is supported by evaluating the Lyapunov function and its time derivative (2.19) during the simulation. The functions are presented in Figure 4.3. As seen, the Lyapunov function and its time derivative are continuous. The fact that there are no peaks during the switching proves that this hybrid control mechanism is a continuous process and accordingly the theory presented in Chapter 2 is still valid.

Next, some more simulations are performed with different settings to test the method further. It is now assumed that the masses of each link are not equal to each other, so there are two parameters to be estimated (i.e. the robot is asymmetric). The parameter vector is, however, equal for both the model with foot 1 fixed as the model with foot 2 fixed to the ground: $\theta_i = \begin{bmatrix} m_1 & m_2 \end{bmatrix}^\top$. This means that also here a continuous parameter update can take place. The parameter estimation process and time derivative of the Lyapunov function are visualized in Figure 4.4. As seen, the results are similar to the results of the previous simulation so it can be concluded that the method may also work with more parameters to be estimated.

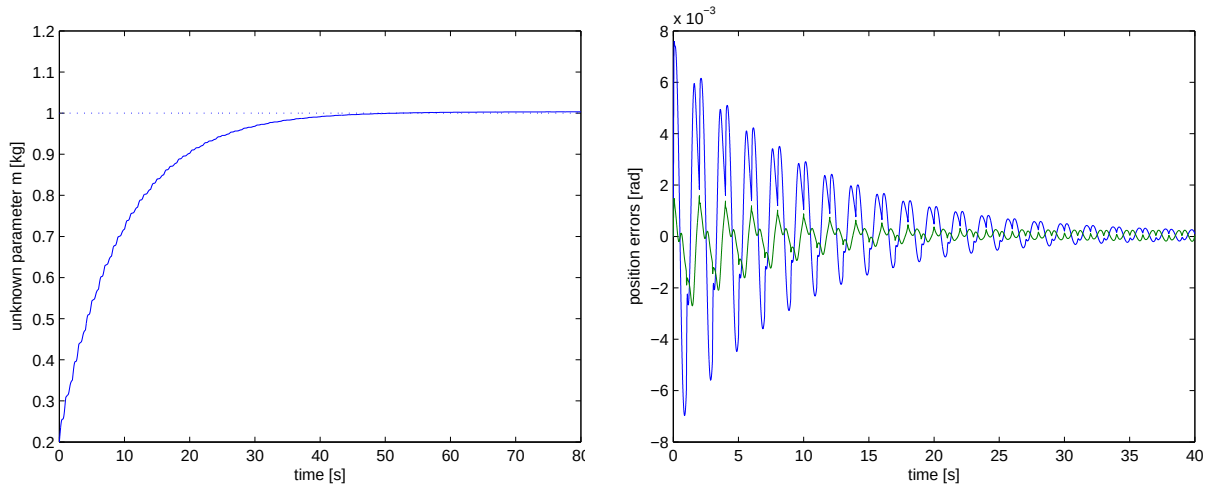

Figure 4.2: Parameter estimation and position error. One unknown mass parameter, $m_1 = m_2$.

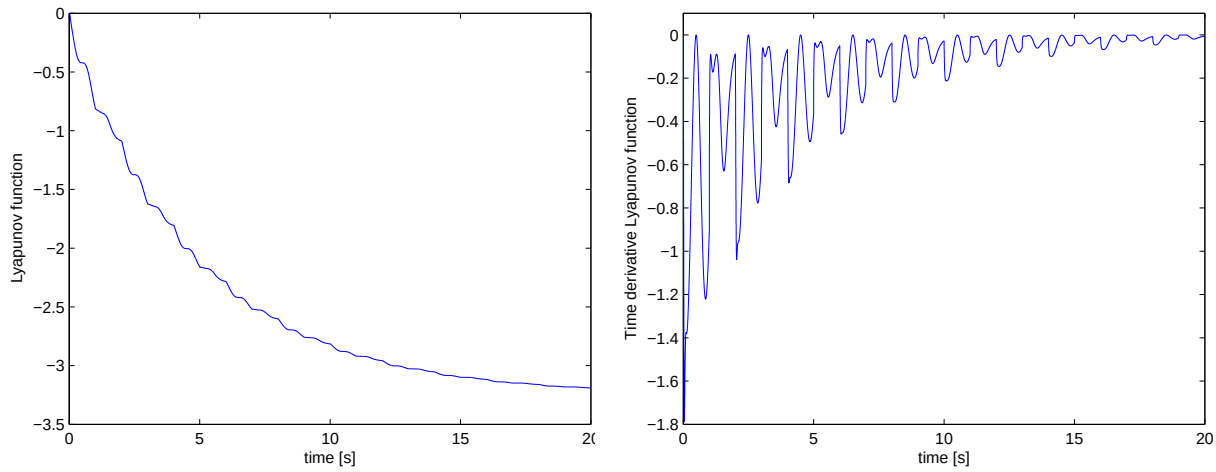

Figure 4.3: Lyapunov function and its time derivative evaluated. One unknown parameter. Symmetric robot.

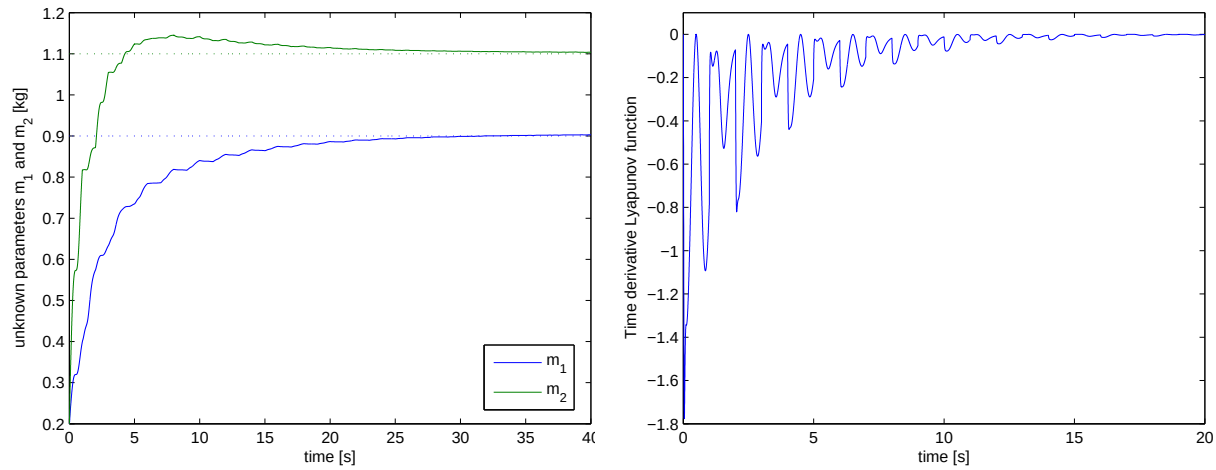

Figure 4.4: Parameter update process (left) and time derivative of the Lyapunov function (right). Two unknown parameters.

The previous simulations indicate that the hybrid adaptive controller may work as expected in the case of symmetric- and asymmetric robots when the parameter vectors θ_i are equal. Next, the case of asymmetric robots is investigated with base parameter sets [3] to be estimated. Consider the same simple bipedal robot. Now the masses are assumed not to be equal and also the locations of the centers of mass are different for both links. This results in different base parameter sets after regression:

$$\theta_1 = \begin{bmatrix} m_1 c_1^2 - a_1 m_1 c_1 \\ m_2 c_2^2 - a_2 m_2 c_2 \\ m_1 - c_1 m_1/a_1 + c_2 m_2/a_2 \\ m_2 - c_2 m_2/a_2 \end{bmatrix}, \quad \theta_2 = \begin{bmatrix} m_1 c_1^2 - a_1 m_1 c_1 \\ m_2 c_2^2 - a_2 m_2 c_2 \\ m_1 - c_1 m_1/a_1 \\ m_2 + c_1 m_1/a_1 - c_2 m_2/a_2 \end{bmatrix}, \tag{4.1}$$

where a_i is the length of link i and c_i the location of center of mass of link i over de length of the link.

Note that the first two base parameters of both sets are equal, but the last two are not. In this case the adaptation is not a continuous process for four parameters which might cause instability. Using the same controller settings as in the simulations before, a simulation is performed on the asymmetric robot. Parameters θ_1 are estimated during a step with foot 1 fixed and parameters θ_2 are estimated during a step with foot 2 fixed. The adaptation of the parameters that cannot be estimated during a step are set to hold. In Figure 4.6 the estimation process of the six base parameters is presented. All parameters converge and the position error decreases over time to a steady error signal, see Figure 4.5.

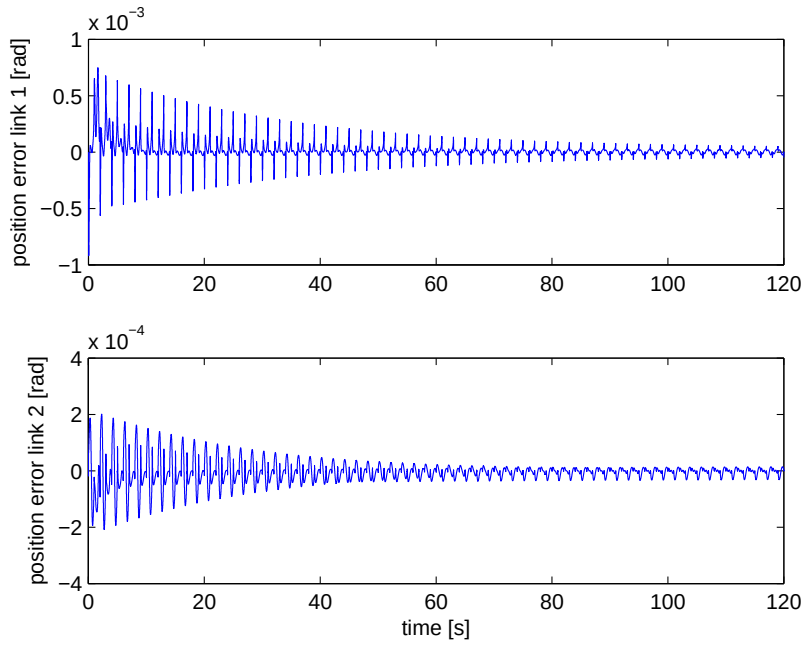

Figure 4.5: Position error course of the two-link walk robot using the hybrid adaptive controller, six unknown parameters, asymmetric robot.

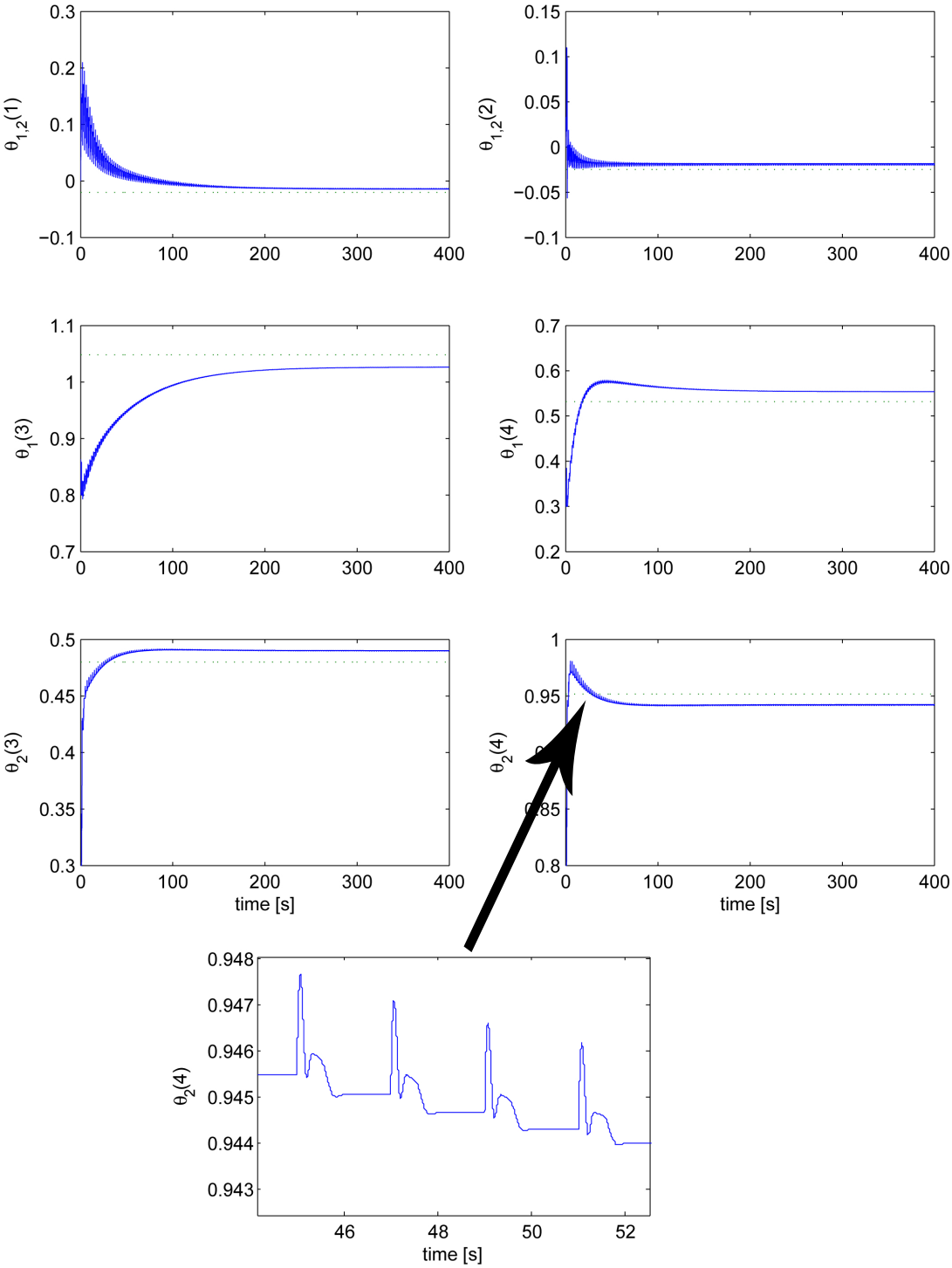

Figure 4.6: Parameter estimation of the two-link walk robot using the hybrid adaptive controller, six unknown parameters, asymmetric robot.

Again, the Lyapunov function is evaluated and presented in Figure 4.7. Notice that there are peaks present in the evaluation. Due to the fact that some parameters are not updated continuously, but are held to the last known value during the time the model is evaluated where these parameters

do not occur in. The update continues when the model is evaluated again where the parameters do occur in. As discussed in Chapter 2, the update law depends on the velocity error, or in this case a combination of velocity- and position error $\dot{e}_r$. When the update is on hold, this error signal is not, resulting in a jump in the adaptation when the adaptation continues. In this case, the stability analysis presented before is no longer valid and thus it is not guaranteed that the controlled system is stable. The simulated system as presented here, however, seems stable, but as seen in Figure 4.5 the position errors are oscillating and do not converge entirely to zero. Some more simulations are performed with different initial conditions and different values of the unknown parameters. The results are similar. This indicates that the method may also work on asymmetric robots, but this cannot be proven by the theory at hand.

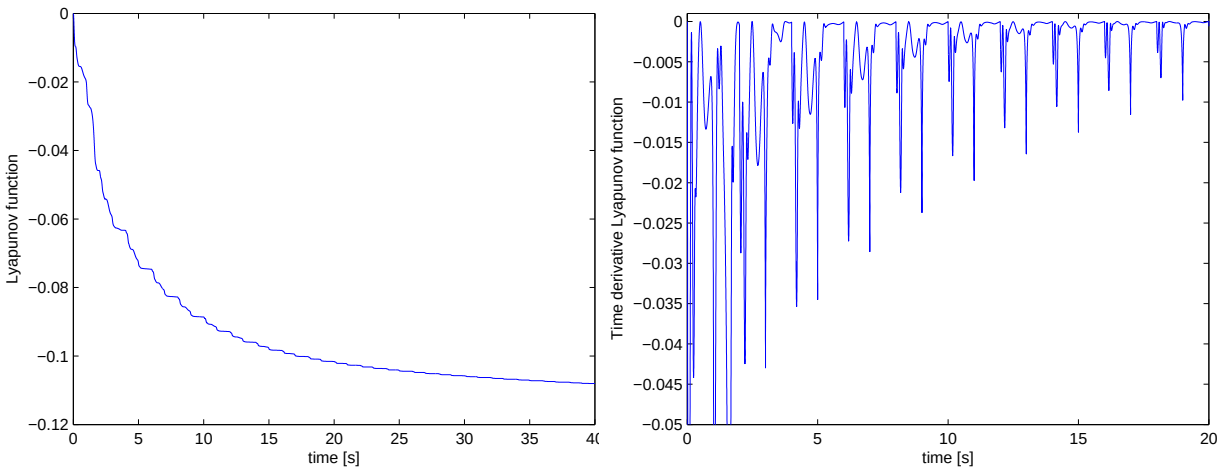

Figure 4.7: Lyapunov function and its time derivative evaluated for the asymmetric robot.

The presented hybrid adaptive control mechanism works on symmetric- and simple asymmetric robots. The bipedal robot is modeled as two serial manipulators: one with foot 1- and one with foot 2 as its stance foot (i.e. the base of the manipulator). The adaptive control mechanism as presented in Chapter 2 is applied and by means of exchange of data a continuous parameter estimation is realised, resulting in a stable closed loop. Humanoid robots are often assumed to be symmetric, so the above presented controller is applicable to humanoid robots. For asymmetric robots however, it cannot be proven that the above method works since the parameter update is not continuous which may cause drift of parameters and thus instability.

4.3 Constrained adaptive control

Another adaptive control method uses a constrained dynamic model of the robot. This constrained model can be found in Appendix C. Constraints that keep one foot of the robot fixed to the ground are added to the equations of motion. The constraints can be expressed in constraint Jacobians P_i and Lagrange multipliers λ_i ($i = 1, 2$, constraints for the left resp. the right foot fixed), i.e. the forces that keep the robot's foot on the ground. Since a humanoid robot often has force sensors in its feet, the justifiable assumption is made that the Lagrange multipliers can be measured and thus be added to the control law. There is just one set of equations of motion in contrast to the hybrid control architecture of the previous Section, while the constraints are switching. In this case the adaptation can be run continuously since the constraints do not enter in the adaptation mechanism.

The constrained equations of motion are expressed as:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = Hu + P_i\lambda_i. \tag{4.2}$$

Take as system states again the position error e_q and velocity error $\dot{e}_q$ so that a positive definite candidate Lyapunov function is still (2.7), with time derivative (2.8). The constrained equations of motion are substituted in this time derivative so that:

$$\begin{aligned} V &= \tfrac{1}{2}\dot{e}_q^\top M(q)\dot{e}_q + \tfrac{1}{2}e_q^\top K_p e_q, \\ \dot{V} &= \dot{e}_q^\top \left[Hu + P_i\lambda_i - M(q)\ddot{q}_d - C(q,\dot{q})\dot{q} - g(q) + \tfrac{1}{2}\dot{M}(q,\dot{q})e_q + K_p e_q\right], \\ &= \dot{e}_q^\top \left[Hu + P_i\lambda_i - M(q)\ddot{q}_d - C(q,\dot{q})\dot{q}_d - g(q) + K_p e_q\right]. \end{aligned} \tag{4.3}$$

Since it is assumed that the Lagrange multipliers λ_i, can be measured, these forces can be added to the control law. The constraint Jacobian P_i depends only on the joint angles, which can be measured. The control law then becomes:

$$Hu = M(q)\ddot{q}_d + C(q,\dot{q})\dot{q}_d + g(q) - P_i\lambda_i - K_p e_q - K_d\dot{e}_q, \tag{4.4}$$

resulting in

$$\dot{V} = -\dot{e}_q^\top K_d \dot{e}_q \leq 0. \tag{4.5}$$

Consider the stability proof in Section 2.3. This proves that if the Lagrange multipliers can be measured, the adaptive control mechanism based on constrained dynamics is stable. The constraints are not part of the regressor matrix, so the constraints are not present in the adaptation law, so the parameter estimation can be done continuously.

The constrained system is implemented in a MATLAB/Simulink model together with an adaptive controller. The unknown parameters are the link masses, which are not equal to each other. The model is run continuously and the constraints are switching. Note that the constraint equations are a function of the unknown parameters, so the estimated values of the masses, obtained by the adaptation law, are forwarded to the part where the constraints are calculated in the simulation. In Figure 4.8 the parameter estimation process and the position error of the two links are presented. As seen, there is convergence of the parameters and also the error decreases. This indicates that such an adaptive control mechanism may be suitable for use on bipedal robots. In Figure 4.9 the Lyapunov function (2.7) and its time derivative are visualized. It can be seen that both functions are continuous, indicating that the stability theory presented in Chapter 2 is still valid. This example is simple, but it shows that switching of constraint equations to not cause a problem for adaptive control.

This method, using constrained equations of motion, might also be suitable to use on asymmetric robots and base parameter sets, in contrast to the hybrid adaptive control mechanism. Since there is no switching in the regressor and parameter update law, there are no discontinuities in the evaluation of the Lyapunov function and its time derivative, indicating a stable control mechanism.

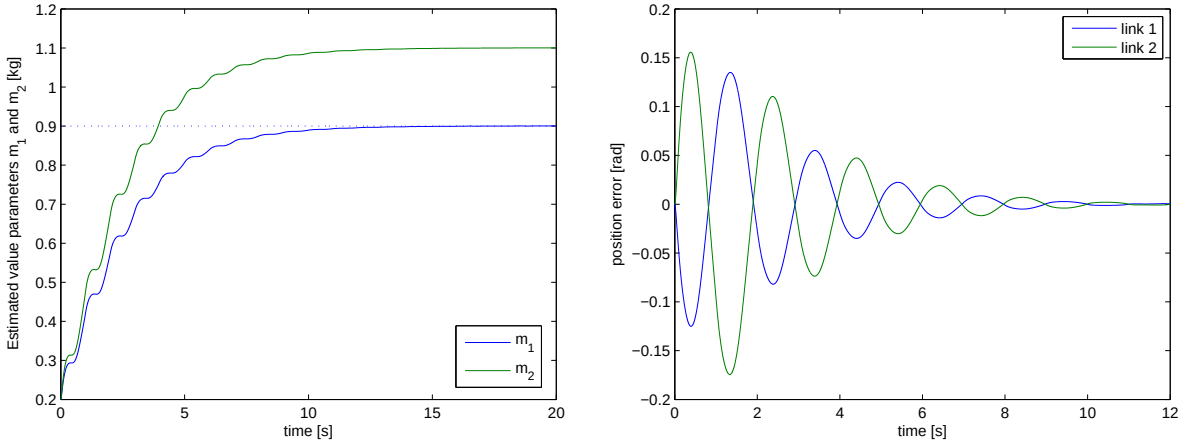

Figure 4.8: Parameter estimation and position error using the constrained dynamics adaptive controller.

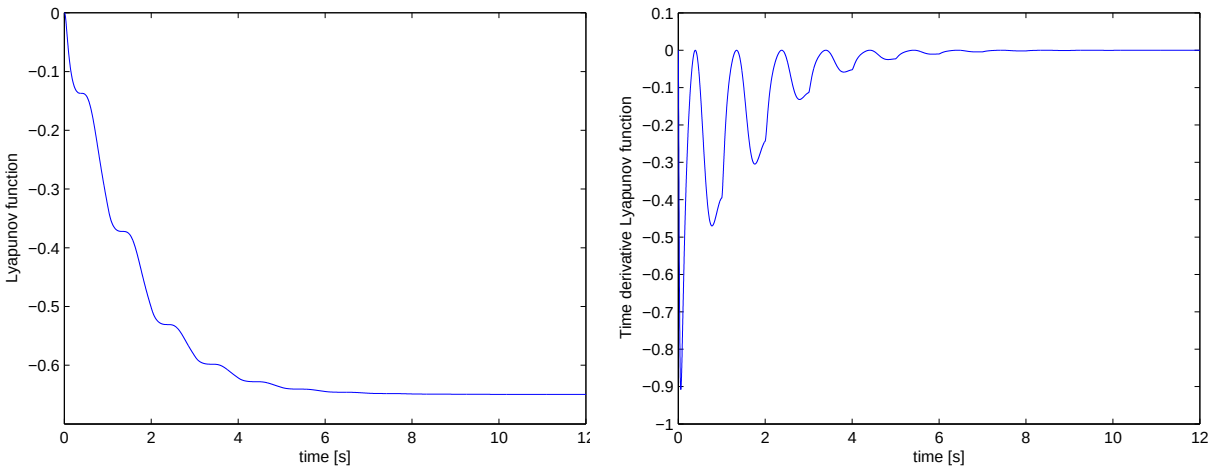

Figure 4.9: Lyapunov function and its time derivative using the constrained dynamics adaptive controller.

5 Conclusions and recommendations

5.1 Conclusions

Conventional Model Reference Adaptive Control may be applicable to bipedal robots by means of a hybrid adaptive controller. Two controllers, each based on a set of equations of motions with one link fixed to the ground (this can be interpreted as the base of a serial robot) are designed and evaluated for the corresponding step of the robot. The two controllers exchange information regarding the parameter estimation, position and velocity of the links so that a continuous adaptation mechanism and tracking controller is achieved. This is supported by simulations.

Another method of adaptive control on bipedal robots is based on a constrained dynamics model. Here, one set of equations of motion is evaluated and the constraint equations (each set of constraint equations keeps one foot on the ground) are switching while walking so that the correct foot is fixed to the world while walking. Simulations on a relatively simple example shows that this method may also work. This method is also promising for asymmetric robots and it can also be applied when using base parameters.

The regression MATLAB algorithm based on the method presented in Chapter 3 is capable of calculating the equations of motion based on the input of DH-parameters and determining the regressor form within acceptable computing times, in contrast to known regression methods.

5.2 Recommendations

There is enough work left to do regarding the topic adaptive control on humanoid robots. Due to the time limits of this open-space project, there are several things that could not be done. It may be usefull to see what happens when the hybrid adaptive controller is applied on the TUlip simulator in Gazebo [22]. A mathematical proof for this method should be derived.

Also the constrained dynamics model could be derived for TUlip. If there is a reliable method for measuring the forces that keep the robot on the ground (i.e. the Lagrange multipliers as discussed in chapter 4, this may be done using the pressure sensors in the feet of TUlip), this method could also be implemented in TUlip's motion control software. When simulations are promising, experiments can be done.

To improve the tracking performance even better, friction in the joints can be added to the adaptive controller. The controller then estimates the friction parameters so that friction in the joints can be compensated.

Bibliography

[1] M. Hackel (editor), *Humanoid Robots: Human-like Machines*. I-Tech Education and Publishing, Vienna, 2007

[2] *The TUlip humanoid robot*. `http://www.techunited.nl/nl/tulip` Accessed: 29/06/2013

[3] J.A.J. Baelemans, *Parameter estimation of humanoid robots using the center of pressure*. Master's thesis, Eindhoven University of Technology, Eindhoven, 2013.

[4] P.W.M. van Zutven, *Modeling, identification and stability of humanoid robots*. Master's thesis, Eindhoven University of Technology, Eindhoven 2009

[5] P.W.M. van Zutven, D. Kostic, H. Nijmeijer, "Parameter identification of robotic systems with series elastic actuators", in *8th IFAC Symposium on Nonlinear Control Systems*, pp. 350-355, 2010

[6] A.G. de Jager, *Applied Nonlinear Control*. Lecture slides 2013.

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

[8] P.W.M. van Zutven, D. Kostic, H. Nijmeijer, "On the stability of bipedal walking", in *LNAI series: Simulation, Modeling and Programming for Autonomous Robots*, pp. 521-532, 2010

[9] H.K. Khalil, *Nonlinear Systems*. Prentice-Hall, 3rd edition, 1996

[10] M.W. Spong, S. Hutchinson, M. Vidyasagar, *Robot modeling and control*. Wiley, 2006

[11] J-J.E. Slotine, W. Li, *Applied Nonlinear Control*. Prentice-Hall, 1st edition, 1991.

[12] M. Gautier, W. Khalil, "Exciting trajectories for the identification of base inertial parameters of robots", in *Proceedings of the 30th IEEE Conference on Decision and Control*, vol. 1, pp. 494-499, 1991

[13] S. Boyd, S.S. Sastry, "Necessary and sufficient conditions for parameter convergence in adaptive control", in *Automatica*, vol. 22, no. 6, pp. 629-639, 1986

[14] J-J.E. Slotine, W. Li, "On the adaptive control of robot manipulators", in *The International Journal of Robotics*, vol. 6 no. 3, pp. 49-59, 1987

[15] W. Khalil, E. Dombre, *Modeling, Identification and control of Robots*. Hermes Penton Ltd, 2002

[16] M. Gautier, W. Khalil, "Direct calculation of minimum set of inertial parameters of serial robots", *IEEE Transactions on Robotics and Automation*, vol. 6, pp. 368-373, 1990

[17] H. Kawasaki, T. Shimizu, K. Kanzaki, "Symbolic analysys of the base parameters for closed-chain robots based on the completion procedure", in *IEEE International Conference on Robotics and Automation, 1996. Proceedings.*, vol. 2, pp. 1781-1786, 1996

[18] *SE Computer Systems*. http://se.wtb.tue.nl/sewiki/sesystems/start Accessed: 10/06/2013

[19] M. Gabiccini, A. Bracci, D. De Carli, M. Fredianelli, A. Bicchi, *Explicit Lagrangian Formulation of the Dynamic Regressors for Serial Manipulators*, http://www.dipmat.univpm.it/aimeta2009/Atti%20Congresso/MECCANICA_MACCHINE/Gabiccini_paper07.pdf, 2007, Accessed: 31/05/2013.

[20] N. van de Wouw, *Multibody Dynamics* reader, Lecture Notes associated with the course "Multibody Dynamics" at Eindhoven University of Technology, Eindhoven University of Technology, 2012

[21] *Dutch Robotics*. http://www.dutchrobotics.net/node/5 Accessed: 29/06/2013

[22] *Gazebo Simulator*. http://www.gazebosim.org Accessed: 14/08/2013

[23] The Mathworks, Inc., *Matlab R2013a documentation*, 2013

A | MATLAB algorithm for direct regression

In this section the MATLAB algorithm that determines the robot's equations of motion and rewrites it to regressor form, is presented. Note that in the code below a three-link robot is considered. However, the algorithm can handle any serial configuration, simply by changing the DH parameters in the code.

```
1  % This script calculates the robot's equations of motion and rewrites it in
2  % regressor form. The script can handle any serial configuration with
3  % revolute joints. Entries to be changed:
4  %       - DH parameters (line 48-53)
5  %       - mass- and inertia parameters if known (line 143-155)
6  %       - gravitational vector; depends on base coordinate frame (line 193)
7  %       - symbolic variable declarations (line 26-39)
8  %       - locations of centers of mass (line 109-111)
9  %         note that extra entries can be added if a robot of more links is
10 %         considered.
11
12 % preprocessing
13 % -------------
14 close all; clear all; clc;
15
16 % ========================================================================
17 %      EQUATIONS OF MOTION
18 % ========================================================================
19
20 % display
21 % -------
22 disp('Start deriving equations of motion');
23
24 % declare symbolic variables
25 % --------------------------
26 syms pi g real;
27 syms a2 a3 real;
28 syms d1 d2 real;
29 syms q1 q2 q3 real;
30 syms q1_dot q2_dot q3_dot real;
31 syms q1d_dot q2d_dot q3d_dot real;
32 syms m1 m2 m3 real;
33 syms Ixx_1 Iyy_1 Izz_1 real;
34 syms Ixx_2 Iyy_2 Izz_2 real;
35 syms Ixx_3 Iyy_3 Izz_3 real;
36 syms lc1x lc1y lc1z real;
37 syms lc2x lc2y lc2z real;
38 syms lc3x lc3y lc3z real;
39 syms q1_ddot q2_ddot q3_ddot;
40
41
42 % Generate vectors between origins of the circumjacent coordinate frames
43 % attached to the robot joints. Generate rotation matrices between the
44 % circumjacent coordinate frames attached to the robot joints. Generate
45 % matrices of homogenous transformations between the circumjacent coordinate
```

```
% frames attached to the robot joints.
% ---------------------------------------------------------------------------
a=[0; a2; a3];
d=[d1; d2; 0];
alf=[0.5*pi; 0; 0];
th=[q1; q2; q3];
thd=[q1_dot; q2_dot; q3_dot];
thdd=[q1_ddot; q2_ddot; q3_ddot];

q=[]; % Vector of generalized coordinates
qd=[]; % Vector of generalized velocities
qdd=[]; % Vector of generalized accelerations
for ii=1:length(a),
    q=[q; th(ii)];
    qd=[qd; thd(ii)];
    qdd=[qdd; thdd(ii)];
end

Null_covector=[0 0 0];

x=[]; R=[]; A=[];
for ii=1:length(a),
    x{ii}=[a(ii)*cos(th(ii)); a(ii)*sin(th(ii)); d(ii)];

    R{ii}=[cos(th(ii)), -cos(alf(ii))*sin(th(ii)),  sin(alf(ii))*sin(th(ii))
           sin(th(ii)),  cos(alf(ii))*cos(th(ii)), -sin(alf(ii))*cos(th(ii))
                     0,              sin(alf(ii)),              cos(alf(ii))];

    A{ii}=[        R{ii},  x{ii}
           Null_covector,      1];
end

% Generate matrices of homogenous transformations representing positions
% x^0_i and orientations R^0_i of the link coordinate frames oixiyizi (i=1,2,3)
% relative to the coordinate frame o0x0y0z0 of the base.
% ------------------------------------------------------------------------------
T_0_{1}=A{1};
for ii=2:length(a),
    T_0_{ii} = simple( T_0_{ii-1} * A{ii} );
end

% Create linear velocity Jacobian Jv and angular velocity Jacobian Jomega
% ---------------------------------------------------------------------
o_0_0=[0;0;0];
z_0_0=[0;0;1];

for kk = 1:length(a),
    J_v_{kk}=[]; J_omega_{kk}=[];
    J_v_{kk}=cross(z_0_0,T_0_{kk}(1:3,4)-o_0_0);
    J_omega_{kk}=z_0_0;
        for ii=1:kk-1,
            J_v_{kk}=[J_v_{kk}, cross(T_0_{ii}(1:3,3),T_0_{kk}(1:3,4)-T_0_{ii}(1:3,4))];
            J_omega_{kk}=[J_omega_{kk}, T_0_{ii}(1:3,3)];
        end
    J_v_{kk}=[J_v_{kk} zeros(3,length(a)-kk)];
    J_v_{kk}=simple(J_v_{kk});
    J_omega_{kk}=[sym(J_omega_{kk}) sym(zeros(3,length(a)-kk))];
    J_omega_{kk}=simple(J_omega_{kk});
end

% Coordinates of the i-th link center of mass expressed relative to the
% i-th coordinate frame
% -------------------------------------------------------------------
o_c_{1}=[0; -lc1y; lc1z];
o_c_{2}=[-lc2x; 0; 0];
o_c_{3}=[-lc3x; 0; 0];

% Origins o_0_{i} (i=1,...,n) of the link coordinate frames expressed
% relative to the coordinate frame of the base and coordinates o_0_c_{i}
% of the i-th link center of mass expressed relative to the coordinate
```

```
% frame of the base
% ----------------------------------------------------------------------
o_0_=[];
o_0_c_=[];
for ii=1:length(a),
    o_0_{ii} = T_0_{ii}(1:3,4) ;
    o_0_c_{ii} = o_0_{ii} + T_0_{ii}(1:3,1:3)*o_c_{ii} ;
end


% Jacobians of link centers of mass
% ---------------------------------
for ii=1:length(a),
    Jv_c_{ii}=sym(0*ones(3,length(a)));
    Jomega_c_{ii}=sym(0*ones(3,length(a)));

    Jv_c_{ii}(:,1) = simple(cross ( z_0_0, o_0_c_{ii}-o_0_0 ));
    Jomega_c_{ii}(:,1)=z_0_0;

    for jj=2:ii,
        Jv_c_{ii}(:,jj) = simple(cross( T_0_{jj-1}(1:3,3), o_0_c_{ii}-o_0_{jj-1} ));
        Jomega_c_{ii}(:,jj)= T_0_{jj-1}(1:3,3);
    end
end

% Link inertial parameters
% ------------------------
m=[m1; m2; m3];

I_{1}=[Ixx_1,    0,    0
           0,Iyy_1,    0
           0,    0,Izz_1];

I_{2}=[Ixx_2,    0,    0
           0,Iyy_2,    0
           0,    0,Izz_2];

I_{3}=[Ixx_3,    0,    0
           0,Iyy_3,    0
           0,    0,Izz_3];


% Elements of the inertia matrix
% ------------------------------
D_q=0;
for ii=1:length(a),
    D_q=simple(D_q + m(ii)*transpose(Jv_c_{ii})*Jv_c_{ii} ...
        + transpose(Jomega_c_{ii})*T_0_{ii}(1:3,1:3)*I_{ii}...
        *transpose(T_0_{ii}(1:3,1:3))*Jomega_c_{ii});
end

% Christoffel symbols
% -------------------
for kk=1:length(a),
    for ii=1:length(a),
        for jj=1:length(a),
            c_q{ii,jj,kk}=simple(0.5*( diff(D_q(kk,jj),q(ii)) ...
                + diff(D_q(kk,ii),q(jj)) - diff(D_q(ii,jj),q(kk)) ));
        end
    end
end

% Matrix C_q appears in the term representing Coriolis and centripetal effects
% ----------------------------------------------------------------------------
C_q=0*D_q;
for kk=1:length(a),
    for jj=1:length(a),
        [kk,jj];
        for ii=1:length(a),
            C_q(kk,jj)=C_q(kk,jj)+c_q{ii,jj,kk}*qd(ii);
```

```
        end
    end
end

% gravity relative to the base frame, this is the vector OPPOSITE to the
% gravity vector!
% ---------------------------------------------------------------------
g_vec=[0; g; 0];

% Potential energy
% ----------------
P=0;
for ii=1:length(a),
    P=simple(P+m(ii)*transpose(g_vec)*o_0_c_{ii});
end

% Elements of the gravity vector
% ------------------------------
g_q=[];
for ii=1:length(a),
    g_q=[g_q; simple(diff(P,q(ii)))];
end

% display
% -------
disp('Equations of motion derived');



% ======================================================================
%      REGRESSION
% ======================================================================

% Display start regression and start timer
disp('Start regression');
tic;

% The regressor Yreg and the parameter vector pi_param are initially empty
Wreg = [];
pi_param = [];

% Express inertia tensor i in a 3x3x6 tensor E and a vector J. These are
% the components of the tensor E
E_{1} = [1 0 0; 0 0 0; 0 0 0];
E_{2} = [0 1 0; 1 0 0; 0 0 0];
E_{3} = [0 0 1; 0 0 0; 1 0 0];
E_{4} = [0 0 0; 0 1 0; 0 0 0];
E_{5} = [0 0 0; 0 0 1; 0 1 0];
E_{6} = [0 0 0; 0 0 0; 0 0 1];
E = [E_{1} E_{2} E_{3} E_{4} E_{5} E_{6}];

% start loop over each link
for ii = 1:length(a),

    % Rewrite the first part of the lagrange equation in X and pi
    % -----------------------------------------------------------

    % calculate X
    X_0_{ii} = simple((J_v_{ii}.'*J_v_{ii})*qd);

    s1 = J_omega_{ii}*qd; S1 = [0 -s1(3) s1(2); s1(3) 0 -s1(1); -s1(2) s1(1) 0];
    s2 = J_v_{ii}*qd; S2 = [0 -s2(3) s2(2); s2(3) 0 -s2(1); -s2(2) s2(1) 0];
    X_1_{ii} = simple((J_v_{ii}.'*S1-J_omega_{ii}.'*S2)*T_0_{ii}(1:3,1:3));

    for jj = 1:6
        X_2_{ii}(:,jj) = J_omega_{ii}.'*T_0_{ii}(1:3,1:3)*E_{jj}*T_0_{ii}(1:3,1:3).'...
            *J_omega_{ii}*qd;
    end

    % pi
```

```
    pi_0_{ii} = m(ii);
    pi_1_{ii} = [m(ii)*o_c_{ii}(1) m(ii)*o_c_{ii}(2) m(ii)*o_c_{ii}(3)].';

    iI_{ii} = I_{ii} + m(ii)*[0 -o_c_{ii}(3) o_c_{ii}(2); o_c_{ii}(3) 0 -...
        o_c_{ii}(1); -o_c_{ii}(2) o_c_{ii}(1) 0].'*[0 -o_c_{ii}(3) o_c_{ii}(2); ...
        o_c_{ii}(3) 0 -o_c_{ii}(1); -o_c_{ii}(2) o_c_{ii}(1) 0];
    Ji = [iI_{ii}(1,1) iI_{ii}(1,2) iI_{ii}(1,3) iI_{ii}(2,2) iI_{ii}(2,3) iI_{ii}(3,3)];
    pi_2_{ii} = Ji.';



    % Rewrite the second part of the lagrange equation in Y and pi
    % (pi already determined above)
    % ------------------------------------------------------------

    % calculate Y
    Y_0_{ii} = [];
    for jj = 1:length(a),
       if q(jj) ~= pi/2
           line_{jj} = 0.5*qd.'*(diff((J_v_{ii}.'*J_v_{ii}),q(jj)))*qd;
       else
           line_{jj} = 0;
       end
       Y_0_{ii} = [Y_0_{ii}; line_{jj}];
    end
    clear line_

    Y_1_{ii} = [];
    for jj = 1:length(a),
        if q(jj) ~= pi/2
            line_{jj} = 0.5*(diff((T_0_{ii}(1:3,1:3).'*S1.'*J_v_{ii}*qd...
                -T_0_{ii}(1:3,1:3).'*S2.'*J_omega_{ii}*qd),q(jj))).';
        else
            line_{jj} = zeros(1,3);
        end
        Y_1_{ii} = [Y_1_{ii}; line_{jj}];
    end
    clear line_

    Y_2_{ii} = [];
    for jj = 1:length(a),
        for kk = 1:6
            if q(jj) ~= pi/2
                line_{jj}(:,kk) = 0.5*qd.'*(diff((J_omega_{ii}.'*T_0_{ii}(1:3,1:3)...
                    *E_{kk}*T_0_{ii}(1:3,1:3).'*J_omega_{ii}),q(jj)))*qd;
            else
                line_{jj} = zeros(1,6);
            end
        end
        Y_2_{ii} = [Y_2_{ii}; line_{jj}];
    end
    clear line_


    % Rewrite the third part of the lagrange equation in Z
    % ----------------------------------------------------

    % calculate Z
    Z_0_{ii} = J_v_{ii}.'*g_vec; % note that the minus sign is dropped, since
    %                              here the vector g is opposite to the true
    %                              gravitational vector

    Z_1_{ii} = [];
    for jj = 1:length(a),
        if q(jj) ~= pi/2
            line_{jj} = diff((T_0_{ii}(1:3,1:3).'*g_vec),q(jj));
        else
            line_{jj} = zeros(3,1);
        end
        Z_1_{ii} = [Z_1_{ii}; line_{jj}.'];
```

```
    end
    clear line_


    % Calculate Xdot from X
    % ---------------------

    X_0_d_{ii} = 0;
    for jj = 1:length(a),
        if q(jj) ~= pi/2 && qd(jj) ~= 0
            X_0_d_{ii} = X_0_d_{ii} + diff(X_0_{ii},q(jj))*qd(jj) ...
                + diff(X_0_{ii},qd(jj))*qdd(jj);
        else
            X_0_d_{ii} = zeros(length(a),1);
        end
    end
    X_1_d_{ii} = 0;
    for jj = 1:length(a),
        if q(jj) ~= pi/2 && qd(jj) ~= 0
            X_1_d_{ii} = X_1_d_{ii} + diff(X_1_{ii},q(jj))*qd(jj) ...
                + diff(X_1_{ii},qd(jj))*qdd(jj);
        else
            X_1_d_{ii} = zeros(length(a),3);
        end
    end
    X_2_d_{ii} = 0;
    for jj = 1:length(a),
        if q(jj) ~= pi/2 && qd(jj) ~= 0
            X_2_d_{ii} = X_2_d_{ii} + diff(X_2_{ii},q(jj))*qd(jj) ...
                + diff(X_2_{ii},qd(jj))*qdd(jj);
        else
            X_2_d_{ii} = zeros(length(a),6);
        end
    end


    % Determine the regressor blocks
    % ------------------------------

    W_0_{ii} = X_0_d_{ii} - Y_0_{ii} + Z_0_{ii};
    W_1_{ii} = X_1_d_{ii} - Y_1_{ii} + Z_1_{ii};
    W_2_{ii} = X_2_d_{ii} - Y_2_{ii};



    % Determine the regressor and parameter vector for link i
    % -------------------------------------------------------

    W_{ii} = [W_0_{ii} W_1_{ii} W_2_{ii}];
    pi_v_{ii} = [pi_0_{ii}; pi_1_{ii}; pi_2_{ii}];



    % Fill total regressor and parameter vector
    % -----------------------------------------

    Wreg = [Wreg W_{ii}];
    pi_param = [pi_param; pi_v_{ii}];


    % Display that one loop is done
    % -----------------------------
    fprintf('loop %i of %i is done\n',ii,length(a));

% end loop
end


% Remove columns that result in zero entries
disp('removing columns from regressor that will result in zero and clean up parameter vector');
```

```
ind = find(pi_param);
pi_param = pi_param(ind);
Wreg = Wreg(:,ind);

% Display complete and stop timer
disp('Regression complete, type "Wreg" and "pi_param" to displays regressor and parameter vector');
toc;
```

B MATLAB algorithm for Slotine & Li regression

In this section the MATLAB algorithm that calculates the Slotine & Li regressor and parameter vector, as explained in section 3.4, is presented. The equations of motion can be determined using the code of appendix A.

```
% This script calculates the robot's Slotine & Li regressor and parameter
% vector. The equations of motion can be obtained using the script in
% appendix A.

% Display start regression and start timer
disp('Start regression');
tic;

% The regressor Yreg and the parameter vector pi_param are initially empty
Wreg = [];
pi_param = [];

% Express inertia tensor i in a 3x3x6 tensor E and a vector J. These are
% the components of the tensor E
E_{1} = [1 0 0; 0 0 0; 0 0 0];
E_{2} = [0 1 0; 1 0 0; 0 0 0];
E_{3} = [0 0 1; 0 0 0; 1 0 0];
E_{4} = [0 0 0; 0 1 0; 0 0 0];
E_{5} = [0 0 0; 0 0 1; 0 1 0];
E_{6} = [0 0 0; 0 0 0; 0 0 1];
E = [E_{1} E_{2} E_{3} E_{4} E_{5} E_{6}];

% Skew symmetric matrix S(p_iG_i) is reshaped as an inner product of the
% third order tensor Q with vector p_iG_i. These are the components of Q
Q_{1} = [0 0 0; 0 0 -1; 0 1 0];
Q_{2} = [0 0 1; 0 0 0; -1 0 0];
Q_{3} = [0 -1 0; 1 0 0; 0 0 0];
Q = [Q_{1} Q_{2} Q_{3}];

% start loop over each link
for ii = 1:length(a),

    % Rewrite the first part of the lagrange equation in X and pi
    % ------------------------------------------------------------

    % calculate X0dot
    dJdq = sym(zeros(length(a),length(a),length(a)));
    for jj = 1:length(a),
        dJdq(:,:,jj) = diff((J_v_{ii}.'*J_v_{ii}),q(jj));
    end
    dJdqT = permute(dJdq,[1 3 2]);
    dJdq_tot = dJdq + dJdqT;
    for jj = 1:length(a),
        dJdq_times_qd(:,jj) = dJdq_tot(:,:,jj)*qd;
    end
    X_d_0r_{ii} = (J_v_{ii}.'*J_v_{ii})*qrdd+0.5*dJdq_times_qd*qrd;
```

```
    % calculate X1dot
    for jj = 1:3
        X_1_{ii}(:,:,jj) = J_omega_{ii}.'*T_0_{ii}(1:3,1:3)*Q_{jj}*T_0_{ii}(1:3,1:3).'*J_v_{ii}...
            -J_v_{ii}.'*T_0_{ii}(1:3,1:3)*Q_{jj}*T_0_{ii}(1:3,1:3).'*J_omega_{ii};
    end
    for jj = 1:3
        X_1_times_qrdd(:,jj) = X_1_{ii}(:,:,jj)*qrdd;
    end
    dX1dq = sym(zeros(length(a),length(a),length(a),3));
    for jj = 1:length(a),
        for kk = 1:3
            dX1dq(:,:,jj,kk) = diff((X_1_{ii}(:,:,kk)),q(jj));
        end
    end
    dX1dqT = permute(dX1dq,[1 3 2 4]);
    dX1dq_tot = dX1dq + dX1dqT;
    dX1dq_times_qd = sym(zeros(length(a),length(a),3));
    for jj = 1:length(a),
        for kk = 1:3
            dX1dq_times_qd(:,jj,kk) = dX1dq_tot(:,:,jj,kk)*qd;
        end
    end
    for jj = 1:3
        X_d_1r_{ii}(:,jj) = X_1_times_qrdd(:,jj) + 0.5*dX1dq_times_qd(:,:,jj)*qrd;
    end

    % calculate X2dot
    for jj = 1:6
        X_2_{ii}(:,:,jj) = J_omega_{ii}.'*T_0_{ii}(1:3,1:3)*E_{jj}*T_0_{ii}(1:3,1:3).'*J_omega_{ii};
    end
    for jj = 1:6
        X_2_times_qrdd(:,jj) = X_2_{ii}(:,:,jj)*qrdd;
    end
    dX2dq = sym(zeros(length(a),length(a),length(a),6));
    for jj = 1:length(a),
        for kk = 1:6
            dX2dq(:,:,jj,kk) = diff((X_2_{ii}(:,:,kk)),q(jj));
        end
    end
    dX2dqT = permute(dX2dq,[1 3 2 4]);
    dX2dq_tot = dX2dq + dX2dqT;
    dX2dq_times_qd = sym(zeros(length(a),length(a),6));
    for jj = 1:length(a),
        for kk = 1:6
            dX2dq_times_qd(:,jj,kk) = dX2dq_tot(:,:,jj,kk)*qd;
        end
    end
    for jj = 1:6
        X_d_2r_{ii}(:,jj) = X_2_times_qrdd(:,jj) + 0.5*dX2dq_times_qd(:,:,jj)*qrd;
    end

    % pi
    pi_0_{ii} = m(ii);
    pi_1_{ii} = [m(ii)*o_c_{ii}(1) m(ii)*o_c_{ii}(2) m(ii)*o_c_{ii}(3)].';

    iI_{ii} = I_{ii} + m(ii)*[0 -o_c_{ii}(3) o_c_{ii}(2); o_c_{ii}(3) 0 -o_c_{ii}(1);...
        -o_c_{ii}(2) o_c_{ii}(1) 0].'*[0 -o_c_{ii}(3) o_c_{ii}(2); o_c_{ii}(3) 0 -o_c_{ii}(1);...
        -o_c_{ii}(2) o_c_{ii}(1) 0];
    Ji = [iI_{ii}(1,1) iI_{ii}(1,2) iI_{ii}(1,3) iI_{ii}(2,2) iI_{ii}(2,3) iI_{ii}(3,3)];
    pi_2_{ii} = Ji.';



    % Rewrite the second part of the lagrange equation in W and pi
    % (pi already determined above)
    % ------------------------------------------------------------

    % skew symmetric matrices S_i
    s1 = J_omega_{ii}*qd; S1 = [0 -s1(3) s1(2); s1(3) 0 -s1(1); -s1(2) s1(1) 0];
```

```
    s2 = J_v_{ii}*qd; S2 = [0 -s2(3) s2(2); s2(3) 0 -s2(1); -s2(2) s2(1) 0];

    % calculate Y
    Y_0_{ii} = [];
    for jj = 1:length(a),
       line_{jj} = 0.5*qrd.'*(diff((J_v_{ii}.'*J_v_{ii}),q(jj)))*qd;
       Y_0_{ii} = [Y_0_{ii}; line_{jj}];
    end
    clear line_

    Y_1_{ii} = [];
    for jj = 1:length(a),
        line_{jj} = 0.5*(diff((T_0_{ii}(1:3,1:3).'*S1.'*J_v_{ii}*qrd-T_0_{ii}(1:3,1:3).'...
            *S2.'*J_omega_{ii}*qrd),q(jj))).';
        Y_1_{ii} = [Y_1_{ii}; line_{jj}];
    end
    clear line_

    Y_2_{ii} = [];
    for jj = 1:length(a),
        for kk = 1:6
            line_{jj}(:,kk) = 0.5*qrd.'*(diff((J_omega_{ii}.'*T_0_{ii}(1:3,1:3)*E_{kk}...
                *T_0_{ii}(1:3,1:3).'*J_omega_{ii}),q(jj)))*qd;
         end
        Y_2_{ii} = [Y_2_{ii}; line_{jj}];
    end
    clear line_



    % Rewrite the third part of the lagrange equation in Z
    % ----------------------------------------------------

    % calculate Z
    Z_0_{ii} = J_v_{ii}.'*g_vec; % note that the minus sign is dropped,
    %                              since here the vector g is opposite to
    %                              the true gravitational vector

    Z_1_{ii} = [];
    for jj = 1:length(a),
        line_{jj} = diff((T_0_{ii}(1:3,1:3).'*g_vec),q(jj));
        Z_1_{ii} = [Z_1_{ii}; line_{jj}.'];
    end
    clear line_



    % Determine the regressor blocks
    % ------------------------------

    W_0_{ii} = X_d_0r_{ii} - Y_0_{ii} + Z_0_{ii};
    W_1_{ii} = X_d_1r_{ii} - Y_1_{ii} + Z_1_{ii};
    W_2_{ii} = X_d_2r_{ii} - Y_2_{ii};



    % Determine the regressor and parameter vector for link i
    % -------------------------------------------------------

    W_{ii} = [W_0_{ii} W_1_{ii} W_2_{ii}];
    pi_v_{ii} = [pi_0_{ii}; pi_1_{ii}; pi_2_{ii}];

    % Fill total regressor and parameter vector
    % -----------------------------------------

    Wreg = [Wreg W_{ii}];
    pi_param = [pi_param; pi_v_{ii}];


% end loop
```

```
end

% Remove columns that result in zero entries
disp('removing columns from regressor that will result in zero and clean up parameter vector');
ind = find(pi_param);
pi_param = pi_param(ind);
Wreg = Wreg(:,ind);

% Display complete and stop timer
disp('Regression complete, type "Wreg" and "pi_param" to displays regressor and parameter vector');
toc;
```

C | Model of the two-link walking robot

C.1 Equations of motion

In this section, the mathematical model and constraints of the two-link walk robot, introduced in chapter 4, is presented. Consider the robot, presented in Figure 4.1 and for clarity, again in Figure C.1. The robot's position can be uniquely described by the generalized coordinates $q = \begin{bmatrix} x(\theta_1, \theta_2)) & y(\theta_1, \theta_2) & \theta_1 & \theta_2 \end{bmatrix}^\top$, where the coordinates x and y can be expressed as a function of the joint angles, since the robot is walking.

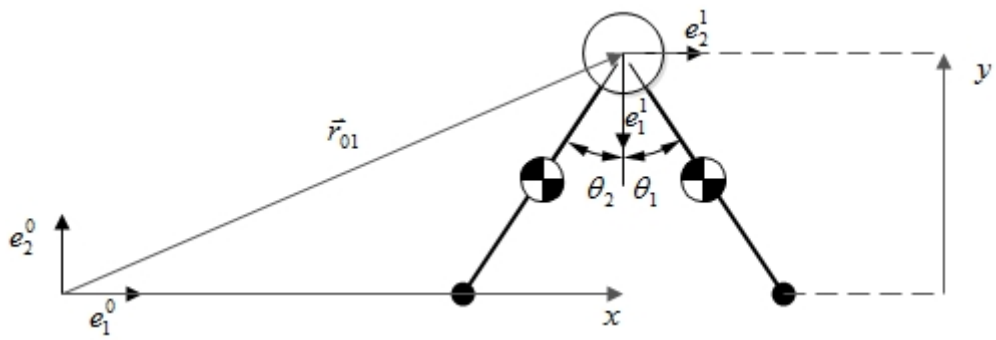

Figure C.1: Two-link biped robot.

Frame $\underline{e}^0 = \begin{bmatrix} e_1^0 & e_2^0 \end{bmatrix}^\top$ is fixed to the world, while frame $\underline{e}^1 = \begin{bmatrix} e_1^1 & e_2^1 \end{bmatrix}^\top$ is fixed to the joints of the robot. The position vector $\vec{r}_{01}$ represents the location of the robot's fixed frame relative to the world frame. This position vector is expressed as:

$$\vec{r}_{01} = \begin{bmatrix} x & y \end{bmatrix}. \tag{C.1}$$

Next, the position vectors $\vec{r}_{CM_1}$ and $\vec{r}_{CM_2}$ are introduced, as the position vectors of the centers of mass relative to the world frame $\underline{e}^0$:

$$\begin{aligned} \vec{r}_{CM_1} &= \begin{bmatrix} x - \frac{L}{2}\sin(\theta_1) & y - \frac{L}{2}\cos(\theta_1) \end{bmatrix} \underline{e}^0, \\ \vec{r}_{CM_2} &= \begin{bmatrix} x + \frac{L}{2}\sin(\theta_2) & y - \frac{L}{2}\cos(\theta_2) \end{bmatrix} \underline{e}^0. \end{aligned} \tag{C.2}$$

Now the Lagrange equations of motion are determined using the kinetic- and potential energy. The kinetic energy for each link is:

$$
\begin{aligned}
T_i &= \tfrac{1}{2} m_i \dot{\vec{r}}_{CMi} \cdot \dot{\vec{r}}_{CMi}, \\
T_1 &= \tfrac{1}{2} m_1 \left(\left(\dot{x} - \tfrac{L}{2}\dot{\theta}_1 \cos(\theta_1) \right)^2 + \left(\dot{y} + \tfrac{L}{2}\dot{\theta}_1 \sin(\theta_1) \right)^2 \right), \\
T_2 &= \tfrac{1}{2} m_2 \left(\left(\dot{x} + \tfrac{L}{2}\dot{\theta}_2 \cos(\theta_2) \right)^2 + \left(\dot{y} + \tfrac{L}{2}\dot{\theta}_2 \sin(\theta_2) \right)^2 \right),
\end{aligned}
\tag{C.3}
$$

and the potential energy can be expressed as:

$$
V = g m_1 \left(y - \frac{L}{2} \cos(\theta_1) \right) + g m_2 \left(y - L \cos(\theta_2) \right). \tag{C.4}
$$

Using the above expressions, the Lagrange equations of motion are derived:

$$
\frac{\partial}{\partial t} \frac{\partial T}{\partial \dot{q}} - \frac{\partial T}{\partial q} + \frac{\partial V}{\partial q} = H u
$$

$$
\Longleftrightarrow
$$

$$
\begin{bmatrix}
(m_1 + m_2)\,\ddot{x} + \frac{L}{2} m_1 \sin(\theta_1)\dot{\theta}_1^2 - \frac{L}{2} m_1 \ddot{\theta}_1 \cos(\theta_1) + \frac{L}{2} m_2 \ddot{\theta}_2 \cos(\theta_2) - \frac{L}{2} m_2 \ddot{\theta}_2 \sin(\theta_2) \\
(m_1 + m_2)\,\ddot{y} + \frac{L}{2} m_1 \cos(\theta_1)\dot{\theta}_1^2 + \frac{L}{2} m_2 \cos(\theta_2)\dot{\theta}_2^2 + g m_1 + g m_2 + \frac{L}{2} m_1 \ddot{\theta}_1 + \frac{L}{2} m_2 \ddot{\theta}_2 \sin(\theta_2) \\
\frac{L^2}{4} m_1 \ddot{\theta}_1 + \frac{L}{2} g m_1 \sin(\theta_1) - \frac{L}{2} m_1 \ddot{x} \cos(\theta_1) + \frac{L}{2} m_1 \ddot{y} \sin(\theta_1) \\
\frac{L^2}{4} m_2 \ddot{\theta}_2 + \frac{L}{2} g m_2 \sin(\theta_2) + \frac{L}{2} m_2 \ddot{x} \cos(\theta_2) + \frac{L}{2} m_2 \ddot{y} \sin(\theta_2)
\end{bmatrix}
$$

$$
= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix},
\tag{C.5}
$$

which can be written in the form (2.1), with

$$
M = \begin{bmatrix}
m_1 + m_2 & 0 & -\frac{L}{2} m_1 \cos(\theta_1) & \frac{L}{2} m_2 \cos(\theta_2) \\
0 & m_1 + m_2 & \frac{L}{2} m_1 \sin(\theta_1) & \frac{L}{2} m_2 \sin(\theta_2) \\
-\frac{L}{2} m_1 \cos(\theta_1) & \frac{L}{2} m_1 \sin(\theta_1) & \frac{L^2}{4} m_1 & 0 \\
\frac{L}{2} m_2 \cos(\theta_2) & \frac{L}{2} m_2 \sin(\theta_2) & 0 & \frac{L^2}{4} m_2
\end{bmatrix}, \tag{C.6}
$$

$$
C = \begin{bmatrix}
0 & 0 & \frac{L}{2} m_1 \dot{\theta}_1 \sin(\theta_1) & -\frac{L}{2} m_2 \dot{\theta}_2 \sin(\theta_2) \\
0 & 0 & \frac{L}{2} m_1 \dot{\theta}_1 \cos(\theta_1) & \frac{L}{2} m_2 \dot{\theta}_2 \cos(\theta_2) \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}, \tag{C.7}
$$

$$g = \begin{bmatrix} 0 \\ g\left(m_1 + m_2\right) \\ \frac{L}{2} g m_1 \sin(\theta_1) \\ \frac{L}{2} g m_2 \sin(\theta_2) \end{bmatrix}, \tag{C.8}$$

$$Hu = \begin{bmatrix} 0 \\ 0 \\ \tau_1 \\ \tau_2 \end{bmatrix}. \tag{C.9}$$

Note that only the third and fourth coordinate, θ_1 and θ_2, are actuated.

C.2 Constraints

To keep the robot with one link on the ground, the considered link is fixed by means of constraints which are added to the equations of motion. The constraints are chosen at velocity level, so that a "no-slip" condition in e_1^0-direction is achieved. A constraint in e_2^0 is superfluous since gravity pulls the robot towards the ground. Taking into account the geometry of the robot and the definition of the angles, consider the angels θ_1 and θ_2 as defined in Figure C.1.

To keep foot i on the ground, first the velocity vector of the foot with respect to the base frame is determined:

$$\begin{aligned} \dot{\vec{r}}_{foot_1} &= \dot{\vec{r}}_{01} + \dot{\vec{r}}_{1/foot_1} = \begin{bmatrix} \dot{x} - L\dot{\theta}_1 \cos(\theta_1) & \dot{y} + L\dot{\theta}_1) \sin(\theta_1) \end{bmatrix} \underline{e}^0, \\ \dot{\vec{r}}_{foot_2} &= \dot{\vec{r}}_{01} + \dot{\vec{r}}_{1/foot_2} = \begin{bmatrix} \dot{x} + L\dot{\theta}_2 \cos(\theta_2) & \dot{y} + L\dot{\theta}_2 \sin(\theta_2) \end{bmatrix} \underline{e}^0. \end{aligned} \tag{C.10}$$

The constraints can be expressed as:

$$\begin{aligned} &\dot{\vec{r}}_{foot_i} \cdot \vec{e}_1^0 = 0, \\ &\theta_1 = u_1 \quad \text{(driving constraint)}, \\ &\theta_2 = u_2 \quad \text{(driving constraint)}, \end{aligned} \tag{C.11}$$

so that for the situation with foot 1 fixed:

$$\begin{aligned} &\dot{x} - L\dot{\theta}_1 \cos(\theta_1) = 0, \\ &\theta_1 - u_1 = 0, \\ &\theta_2 - u_2 = 0, \end{aligned} \tag{C.12}$$

and for the situation with foot 2 fixed:

$$
\begin{aligned}
&\dot{x} + L\dot{\theta}_2 \cos(\theta_2) = 0, \\
&\theta_1 - u_1 = 0, \\
&\theta_2 - u_2 = 0.
\end{aligned}
\tag{C.13}
$$

Then, the constraint Jacobians are:

$$
\begin{aligned}
P_1^\top &= \begin{bmatrix} 1 & 0 & -L\cos(\theta_1) & 0 \end{bmatrix} \\
P_2^\top &= \begin{bmatrix} 1 & 0 & 0 & L\cos(\theta_2) \end{bmatrix}
\end{aligned}
\tag{C.14}
$$

The constrained equations of motion are:

$$
M(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + g(q) = Hu + P\lambda \tag{C.15}
$$

where λ is the column of Lagrange multipliers, i.e. the forces that keep the robot with one foot on the ground. The Lagrange multipliers can be expressed as:

$$
\lambda_i = \left(P_i^\top M^{-1} P_i\right)^{-1} \left(P_i^\top M^{-1}\left((C\dot{q} + g) - Hu\right) - \bar{w}_i\right), \tag{C.16}
$$

with

$$
\bar{w}_i = \left(\frac{\partial P_i^\top}{\partial t} + \frac{\partial P_i^\top \dot{q}}{\partial q}\right)\dot{q} \tag{C.17}
$$